Tight Lower Bound for the Channel Assignment Problem

ARKADIUSZ SOCALA, University of Warsaw, Poland

We study the complexity of the Channel Assignment problem. An open problem asks whether Channel Assignment admits an $O(c^n)$ (times a polynomial in the bit size) time algorithm, where $n$ is a number of the vertices, for a constant $c$ independent of the weights on the edges. We answer this question in the negative. Indeed, we show that in the standard Word RAM model, there is no $2^{o(n \log n)}$ (times a polynomial in the bit size) time algorithm solving Channel Assignment unless the exponential time hypothesis fails. Note that the currently best known algorithm works in time $O^*(n) = 2^{O(n \log n)}$, so our lower bound is tight (where the $O^*$ notation suppresses polynomial factors).

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems
General Terms: Algorithms, Theory
Additional Key Words and Phrases: Channel assignment, lower bounds, exponential time hypothesis

ACM Reference Format:
Arkadiusz Socala. 2016. Tight lower bound for the channel assignment problem. ACM Trans. Algorithms 12, 4, Article 48 (September 2016), 19 pages.
DOI: http://dx.doi.org/10.1145/2876505

1. INTRODUCTION

In the Channel Assignment problem, we are given a symmetric weight function $w : V^2 \rightarrow \mathbb{N}$ (we assume that $0 \in \mathbb{N}$). The elements of $V$ will be called vertices (as $w$ induces a graph on the vertex set $V$ with edges corresponding to positive values of $w$). We say that $w$ is $\ell$-bounded when for every $x, y \in V$ we have $w(x,y) \leq \ell$. An assignment $c : V \rightarrow \mathbb{Z}$ is called proper when for each pair of vertices $x, y$ we have $|c(x) - c(y)| \geq w(x,y)$. The number $(\max_{v \in V} c(v) - \min_{v \in V} c(v) + 1)$ is called the span of $c$. The goal is to find a proper assignment of minimum span. Note that the special case when $w$ is 1-bounded corresponds to the classical graph coloring problem. It is therefore natural to associate the instance of the channel assignment problem with an edge-weighted graph $G = (V,E)$, where $E = \{uw : w(u,v) > 0\}$ with edge weights $w_E : E \rightarrow \mathbb{N}$ such that $w_E(xy) = w(x,y)$ for every $xy \in E$ (in what follows, we abuse the notation slightly and use the same letter $w$ for both the function defined on $V^2$ and $E$). The minimum span is also referred to as the span of $(G,w)$ and denoted by $\text{span}(G,w)$.

It is interesting to realize the place of Channel Assignment in a hierarchy of constraint satisfaction problems (CSPs). We have already seen that it is a generalization of the classical graph coloring. It is also a special case of the CSP. In CSP, we are given a vertex set $V$, a constraint set $C$, and a number of colors $d$. Each constraint is a set of pairs of the form $(v,t)$, where $v \in V$ and $t \in \{1,\ldots,d\}$. An assignment $c : V \rightarrow \{1,\ldots,d\}$ is proper
if every constraint \( A \in \mathcal{C} \) is satisfied—that is, there exists \((v, t) \in A\) such that \( c(v) \neq t \). The goal is to determine whether there is a proper assignment. Note that \textsc{Channel Assignment} corresponds to \textsc{CSP} where \( d \) is equal to the maximum allowed span and every edge \( uv \) of weight \( w(uv) \) in the instance of \textsc{Channel Assignment} corresponds to the set of constraints of the form \([(u, t_1), (v, t_2)]\), where \(|t_1 - t_2| < w(uv)|

In the general case, the best known algorithm runs in \( O^*(n!) \) time, where \( n \) is the number of vertices (see McDiarmid [2003]). However, there has been some progress on the \( \ell \)-bounded variant. McDiarmid [2003] came up with an \( O^*((2\ell + 1)^n) \)-time algorithm, which was improved by Král [2005] to \( O^*((\ell + 2)^n) \), further to \( O^*((\ell + 1)^n) \) by Cygan and Kowalik [2011], and to \( O^*((2\sqrt{\ell + 1})^n) \) by Kowalik and Socała [2014]. These are all dynamic programming (and hence exponential space) algorithms. The last but one applies the fast zeta transform to get a minor speedup and the last one uses the meet-in-the-middle approach. Interestingly, all of these works show also algorithms that count all proper assignments of span at most \( s \) within the same running time (up to polynomial factors) as the decision algorithm.

Since graph coloring is solvable in time \( O^*(2^n) \) [Björklund et al. 2009], it is natural to ask whether \textsc{Channel Assignment} is solvable in time \( O^*(c^n) \) for some constant \( c \). It is a major open problem (see Král [2005], Cygan and Kowalik [2011], and Husfeldt et al. [2013]) to find such a \( O(c^n) \)-time algorithm for \( c \) independent of \( \ell \) or prove that it does not exist under a reasonable complexity assumption. A complexity assumption commonly used in such cases is the exponential time hypothesis (ETH), introduced by Impagliazzo et al. [2001]. It states that 3-CNF-SAT cannot be computed in time \( 2^{o(n)} \), where \( n \) is the number of variables in the input formula. The open problem mentioned earlier becomes even more interesting when we realize that under ETH, CSP does not have a \( O^*(c^n) \)-time algorithm for a constant \( c \) independent of \( d \), as proved by Traxler [2008].

Our results. Our main result is a proof that \textsc{Channel Assignment} does not admit a \( O(c^n) \)-time algorithm for a constant \( c \) under the ETH assumption. By applying a sequence of reductions (Figure 1) starting in 3-CNF-SAT and ending in \textsc{Channel Assignment}, we were able to solve this open problem and show that there is no \( 2^{o(n \log n)} \)-time algorithm solving \textsc{Channel Assignment} unless the ETH fails. Note that the currently best known algorithm works in time \( O^*(n!) = 2^{O(n \log n)} \), so our lower bound is tight.

\textsc{Equal Weight Matchings} as a generic problem without the \( 2^{o(n \log n)} \)-time algorithm.
To prove that there is no \( 2^{o(n \log n)} \)-time algorithm for some problem, we may want to use a reduction from some better studied problem, say from 3-CNF-SAT, for which we know that there is no \( 2^{o(n)} \)-time algorithm unless the ETH fails. Therefore, in this case, we need to be able to transform an instance of 3-CNF-SAT of size \( n \) into an instance of our target problem of size \( O\left(\frac{n}{\log n}\right) \). Then a \( 2^{o(n \log n)} \)-time algorithm for our target problem would imply a \( 2^{o(n)} \)-time algorithm for 3-CNF-SAT, which contradicts the ETH. However, such reductions that compress the size of the instance from \( O(n) \) to, for example, \( O\left(\frac{n}{\log n}\right) \) are rare (for more examples of linear to sublinear reductions, see

---

**Fig. 1.** Sequence of the used reductions and the size of the instance. The compression follows between \textsc{Family Intersection} and \textsc{Equal Weight Matchings}. Although the definition of \textsc{Family Intersection} is rather technical, the \textsc{Equal Weight Matchings} problem is quite natural and can be used as a generic problem without the \( 2^{o(n \log n)} \)-time algorithm.
Cygan et al. [2013] and Lokshtanov et al. [2011a, 2011b]). As shown in the Figure 1, we do this for the problem **Equal Weight Matchings** defined as follows:

**Equal Weight Matchings**

**Input:** Two complete weighted bipartite graphs $G_1 = (V_1 \cup W_1, E, w_1)$ and $G_2 = (V_2 \cup W_2, E, w_2)$ such that $|V_1| = |W_1|$ and $|V_2| = |W_2|$. The weight functions $w_1, w_2$ have nonnegative integer values.

**Question:** Are there two perfect matchings $M_1$ in $G_1$ and $M_2$ in $G_2$ such that $w_1(M_1) = w_2(M_2)$?

Note that to show that a new problem $P$ does not admit a $2^{o(n \log n)}$-time algorithm, it suffices to give a linear reduction from **Equal Weight Matchings** to $P$. We show such reduction for **Channel Assignment** and hope that the same thing can be done for other problems.

**Organization of the article.** In Section 2, we describe a sequence of reductions starting in 3-CNF-SAT and ending in **Equal Weight Matchings** and the conclusions from the existence of these reductions leading to the theorem on the hardness of **Equal Weight Matchings**. In Section 3, we present a reduction from **Equal Weight Matchings** to **Channel Assignment** and prove the hardness of **Channel Assignment**.

**Notation.** Throughout the article, $n$ denotes the number of vertices of the graph under consideration. For an integer $k$, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. We denote an instance of (decisional) **Channel Assignment** as $I = (G, w, s)$ where the instance is satisfied when $\text{span}(G, w) \leq s$.

### 2. Hardness of Equal Weight Matchings

In this section, we describe a sequence of reductions starting in 3-CNF-SAT and ending in **Equal Weight Matchings** and the consequences of these reductions on the complexity of **Equal Weight Matchings**. In the second of these two reductions, we compress the instance from the size $O(n)$ to the size $O(n \log n)$, which is an important part of our result.

#### 2.1. From 3-CNF-SAT to Family Intersection

The intuition is that for a given instance of 3-CNF-SAT, we consider a set of the occurrences of the variables in the formula—that is, we treat any two different occurrences of the same variable as if they were two different variables. Note that in a 3-CNF-SAT instance with $n$ variables and $m$ clauses, we have $3m$ occurrences of the $n$ variables so that there are $2^{3m}$ assignments of the occurrences.

We would like to represent two useful subsets of the set of all $2^{3m}$ assignments of the occurrences. The first is the set of the consistent assignments—that is, such assignments of the occurrences that all occurrences of the same variable have the same value. The second is the set of the assignments of the occurrences such that every clause is satisfied (although they are allowed to have different values for different occurrences of the same variable, i.e., they do not need to be consistent). Note that the instance of 3-CNF-SAT is a YES-instance if and only if the intersection of these two sets is nonempty.

To represent those two sets, we would like to use the following concept. For a function $f : [a] \times [b] \to \mathbb{N}$, we define $X_f = \{ \sum_{i=1}^{a} f(i, \sigma(i)) | \sigma : [a] \to [b] \}$. We call this set an $f$-family.

We will define a function $f$ such that the elements of the $f$-family $X_f$ correspond to the assignments of the occurrences such that every two occurrences of the same variable have the same value. Then we define another function $g$ such that the
Thus, in our example, we have $f(1,1) = 10102$, $f(2,1) = 01002$, and $f(3,1) = 00012$. Therefore, $X_f = \{00002, 00012, 01002, 01012, 10102, 10112, 11102, 11112\}$.

To represent the set of the consistent assignments of the occurrences, we can use a function $g : [m] \times [3] \to \mathbb{N}$ such that $g(i,j)$ is a $j$-th assignment (in some fixed order) of the occurrences of the variables in the $i$-th clause that satisfies this clause. Note that every clause in 2-CNF-SAT has three assignments of the occurrences that satisfy this clause. Thus, in our example, we have $g(1,1) = 10002$, $g(1,2) = 01002$, $g(1,3) = 11002$, $g(2,1) = 00002$, $g(2,2) = 00012$ and $g(2,3) = 01112$. Therefore, $X_g = \{01002, 01012, 01112, 10002, 10012, 10112, 11002, 11012, 11112\}$.

The set $X_f \cap X_g = \{01002, 01012, 10112, 11112\}$ is the set of all consistent assignments of the occurrences such that each clause is satisfied.

We can formalize our observation as follows.

**Lemma 2.2.** There is a polynomial time reduction from a given instance of 3-CNF-SAT with $n$ variables and $m$ clauses into an instance of FAMILY INTERSECTION with $f : [n] \times [2] \to \mathbb{N}$ and $g : [m] \times [7] \to \mathbb{N}$ such that $\max X_f < 2^{3m}$ and $\max X_g < 2^{3m}$.

**Proof.** Let $V = \{v_1, v_2, \ldots, v_n\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be the sets of variables and clauses of the input formula, respectively. Let $D = \{d_1, d_2, \ldots, d_{3m}\}$ be the set of all $3m$ occurrences of our $n$ variables in our $m$ clauses. We will treat these occurrences as separate variables. For every variable $v_i \in V$, we define a set $I_i \subseteq [3m]$ such that $j \in I_i$ if and only if $d_j$ is an occurrence of the variable $v_i$. Similarly, for every clause $c_i \in C$, we define a set $J_i \subseteq [3m]$ such that $j \in J_i$ if and only if $d_j$ is an occurrence (of any variable) belonging to the clause $c_i$. For every $i \in [m]$, we have $|J_i| = 3$.

For every clause $c_i$, we can treat the subsets of $J_i$ as the assignments of the occurrences $d_j$ belonging to the clause $c_i$. We treat the subset $K \subseteq J_i$ as the assignment of the occurrences in the clause $c_i$ such that the occurrence $d_j$ is set to 1 if and only if $j \in K$, and otherwise it is set to 0. We say that $K \subseteq J_i$ satisfies the clause $c_i$ if the corresponding assignment of the occurrences satisfies this clause.

For every clause $c_i \in C$, let us define the set $P_i = \{K \subseteq J_i : K$ satisfies the clause $c_i\}$. Again note that here we treat all occurrences as the different variables. Note that $|P_i| = 7$ for every $i$, so we can denote $P_i = \{P_{i1}, P_{i2}, \ldots, P_{i7}\}$.

A number from $0, 1, \ldots, 2^{3m} - 1$ can be interpreted in the binary system as the characteristic vector of length $3m$ of a subset of the indices of the occurrences—that is, that the $i$-th bit represents if the occurrence $d_i$ belongs to this subset or not. We define

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
**Family Intersection** & \\
\hline
**Input:** A function $f : [a] \times [b] \to \mathbb{N}$ and a function $g : [c] \times [d] \to \mathbb{N}$. & \\
**Question:** Is $X_f \cap X_g$ nonempty? & \\
\hline
\end{tabular}
\end{table}
a function \( f : [n] \times [2] \rightarrow \mathbb{N} \) such that for every \( i \in [n] \), we set \( f(i, 1) = \sum_{j \in I(i)} 2^{j-1} \) and \( f(i, 2) = 0 \). In other words, the number \( f(i, 1) \) represents the characteristic vector of all occurrences of the variable \( u_i \). Note that \( \sigma : [n] \rightarrow [2] \) corresponds to the set of all assignments of variables. Therefore, \( X_f \) is the set of all characteristic vectors that represent all assignments of the occurrences such that all occurrences of the same variable have the same value.

We define a function \( g : [m] \times [7] \rightarrow \mathbb{N} \) such that for every \( i \in [m] \) and for every \( j \in [7] \), we can set \( g(i, j) = \sum_{c_i \in \mathcal{P}_i} 2^{k-1} \). Then for every \( i \in [m] \), the numbers \( g(i, 1), g(i, 2), \ldots, g(i, 7) \) represent the characteristic vectors of all assignments of the occurrences in the clause \( c_i \) that satisfy this clause. Therefore, the set \( X_g \) is the set of all characteristic vectors that represents the assignments of all \( 3m \) occurrences such that all clauses are satisfied.

It follows that the set \( X_f \cap X_g \) is the set of all characteristic vectors that represent the assignments of the occurrences such that all occurrences of the same variable have the same value and all clauses are satisfied. In other words, elements of \( X_f \cap X_g \) correspond to satisfying assignments. \( \square \)

### 2.2. From Family Intersection to Equal Weight Matchings

Consider an \( f \)-family \( X_f \) and a \( g \)-family \( X_g \) for some functions \( f : [n] \times [2] \rightarrow \mathbb{N} \) and \( g : [m] \times [7] \rightarrow \mathbb{N} \). In this section, we show how to encode \( X_f \) in some weighted bipartite graph \( G_1 \) so that the set of the weights of the perfect matchings in \( G_1 \) will be equal to \( X_f \). Similarly, we will encode \( X_g \) in some bipartite graph \( G_2 \) such that the set of the weights of the perfect matchings in \( G_2 \) will be equal to \( X_g \). Thus, the set \( X_f \cap X_g \) is nonempty if and only if \( G_1 \) and \( G_2 \) contain perfect matchings with the same weight. Moreover, the number of vertices of the graph \( G_1 \) will be \( O(\frac{n}{\log n}) \), and the number of vertices of the graph \( G_2 \) will be \( O(\frac{m}{\log m}) \). This is a crucial step of our construction, as the instance size decreases (by a logarithmic factor).

Before we describe the reduction, we need the two following technical lemmas that describe a construction of permutations with certain properties. The permutations correspond naturally to perfect matchings in bipartite graphs. Elements of \([k]^b\) will be treated as \( b \)-character words over alphabet \([k]\)—that is, for \( x \in [k] \) and \( w \in [k]^b \) by \( xw \), we mean the word of length \( b + 1 \) obtained by concatenating \( x \) and \( w \). For convenience, we define a set \( \hat{\mathbb{N}} = \{0, 1, 2, \ldots\} \) as a copy of the natural numbers \( \mathbb{N} \), and for every \( n \in \mathbb{N} \), we define \( [n] = \{1, 2, \ldots, n\} \). Every set \([k]^b\) is just a copy of \([k]^b\), so we refer to bijections between \([k]^b\) and \([\hat{k}]^b\) as to permutations.

The first lemma provides a way of merging \( k \) permutations \( \phi_1, \phi_2, \ldots, \phi_k : [k]^b \rightarrow [k]^b \) into one permutation \( \psi : [\hat{k}]^{b+1} \rightarrow [\hat{k}]^{b+1} \) in a way specified by a function \( \rho : [\hat{k}]^{b+1} \rightarrow [k] \).

The second lemma is the first one to provide a way of encoding a function with one argument and \( k \) values as a permutation of the number of elements that is sublinear with respect to the size of the domain of the function. This leads us later in the reduction to the way of encoding an \( f \)-family in a full weighted bipartite graph with a sublinear number of vertices.

**Lemma 2.3.** For every \( b \in \mathbb{N} \) and for a given sequence of permutations \( \phi_1, \phi_2, \ldots, \phi_k : [k]^b \rightarrow [k]^b \) and for every function \( \rho : [k] \rightarrow [k] \), there is a permutation \( \psi : [\hat{k}]^{b+1} \rightarrow [\hat{k}]^{b+1} \) such that

(i) for every \( x \in [k] \) and for every \( \hat{w} \in [\hat{k}]^b \), there exists \( y \in [k] \) such that \( \psi(\hat{x} \hat{w}) = y \psi_x(\hat{w}) \), and moreover,
(ii) for every \( \hat{w} \in [\hat{k}]^b \), we have \( \psi(\hat{1} \hat{w}) = \rho(\hat{w}) \phi_1(\hat{w}) \).
Before we proceed to the proof, we suggest that the reader takes a look at an example in Figure 2 ($b = 1, k = 3$).

**Proof.** We start with the permutation $\hat{x} \hat{w} \mapsto x\hat{\varphi}_x(\hat{w})$, which already satisfies the condition $\exists_\varphi(x)\hat{\varphi}(\hat{x}\hat{w}) = y\hat{\varphi}_x(\hat{w})$. Next, we are going to swap the values for some (disjoint) pairs of the arguments to fulfill the condition $\hat{\varphi}(\hat{x}\hat{w}) = \rho(\hat{w})\phi_1(\hat{w})$. Such swaps are preserving the condition of being a permutation. Moreover, we perform only such swaps that also preserve the $\exists_\varphi(x)\hat{\varphi}(\hat{x}\hat{w}) = y\hat{\varphi}_x(\hat{w})$ condition.

For every $\hat{w} \in [k]^b$, we need to put $\phi_1(\hat{w}) = \rho(\hat{w})\phi(\hat{w})$. Let us assign $x = \rho(\hat{w})$ and $\hat{u} = \phi^{-1}_x(\phi(\hat{w}))$. Note that $\phi_x(\hat{u}) = \phi(\hat{w})$. If $x \neq 1$, then we can put $\phi(\hat{x}\hat{u}) = 1\phi(\hat{w})$. Thus, we have swapped the values for the arguments $\hat{w}$ and $\hat{x}$. Our function is still a permutation. Note that the condition $\exists_\varphi(x)\hat{\varphi}(\hat{x}\hat{w}) = y\hat{\varphi}_x(\hat{w})$ is still preserved because $\phi_\varphi(\hat{w}) = \phi_1(\hat{w})$. We just need to show that the swaps can be performed independently.

For every $i \in [k]$, a function $\phi^{-1}_i \circ \phi_1$ is a permutation, and so for every $\hat{w} \in [k]^b$, the values of $\rho(\hat{w})\phi^{-1}_i(\phi_1(\hat{w}))$ are pairwise different. Indeed, for two different $\hat{u}, \hat{w} \in [k]^b$, either the values $\rho(\hat{u})$ and $\rho(\hat{w})$ are different or $\rho(\hat{u}) = \rho(\hat{w}) = x$ for some $x \in [k]$ and then $(\phi^{-1}_i \circ \phi_1)(\hat{u}) \neq (\phi^{-1}_i \circ \phi_1)(\hat{w})$, so the values $\phi^{-1}_i(\phi_1(\hat{w}))$ and $\phi^{-1}_i(\phi_1(\hat{u}))$ are different. Therefore, our pairs of the arguments to swap are pairwise disjoint. Thus, all swaps can be performed independently.

Therefore, for every $x \in [k]$ and $\hat{w} \in [k]^b$, we have

$$\phi(\hat{x}\hat{w}) = \begin{cases} \rho(\hat{w})\phi_1(\hat{w}) & \text{for } \hat{x} = \hat{1} \\ 1\phi_\varphi(\hat{w}) & \text{for } \hat{x} \neq \hat{1} \land \rho(\phi^{-1}_i(\phi_x(\hat{w}))) = x \\ x\phi_\varphi(\hat{w}) & \text{in other cases.} \end{cases}$$

The intuition of the following lemma is that for every sequence $\hat{w} \in [k]^b$ and for every $i \in [b]$ such that $\hat{w}_i = \hat{1}$, we can pick a value $\alpha(\hat{w}, i) \in [k]$, and there is a permutation $\phi : [k]^b \rightarrow [k]^b$ such that for every sequence $\hat{w} \in [k]^b$ and for every $i \in [b]$ such that $\hat{w}_i = \hat{1}$, we have $\phi(\hat{w})_i = \alpha(\hat{w}, i)$.
Lemma 2.4. Let $b \in \mathbb{N}$ and $\alpha : [k]^b \times [b] \rightarrow [k] \cup \{\perp\}$ such that for every $\hat{w} \in [k]^b$ and for every $i \in [b]$ holds $\alpha (\hat{w}, i) \neq \perp$ if and only if $\hat{w}_i = \hat{1}$. There is a permutation $\phi : [k]^b \rightarrow [k]^b$ such that for every $\hat{w} \in [k]^b$ and for every $i \in [b]$ if $\hat{w}_i = \hat{1}$, then $\phi (\hat{w})_i = \alpha (\hat{w}, i)$.

Proof. We will use an induction on $b$.

For $b = 0$, we have $\phi (\varepsilon) = \varepsilon$.

For $b > 0$, we can define functions $\alpha_1, \alpha_2, \ldots, \alpha_k : [k]^{b-1} \times [b-1] \rightarrow [k] \cup \{\perp\}$ such that for every $x \in [k]$, every $\hat{w} \in [k]^{b-1}$, and every $i \in [b-1]$, we put $\alpha_k (\hat{w}, i) = \alpha (\hat{w}, i + 1)$. From the inductive hypothesis for $b - 1$ used for every function of $\alpha_1, \alpha_2, \ldots, \alpha_k$, we get the permutations $\phi_1, \phi_2, \ldots, \phi_k : [k]^{b-1} \rightarrow [k]^{b-1}$ such that for every $x \in [k]$, for every $\hat{w} \in [k]^{b-1}$, and for every $i \in [b-1]$, we have that if $\hat{w}_i = \hat{1}$, then $\phi_k (\hat{w})_i = \alpha_k (\hat{w}, i) = \alpha (\hat{w}, i + 1)$.

Now we can use Lemma 2.3 to merge the permutations $\phi_1, \phi_2, \ldots, \phi_k$ using a function $\rho : [k]^{b-1} \rightarrow [k]$ such that $\rho (\hat{w}) = \alpha (\hat{1} \hat{w}, 1)$ for every $\hat{w} \in [k]^{b-1}$. We obtain one permutation $\phi : [k]^b \rightarrow [k]^b$ such that by Lemma 2.3 (i), for every $\hat{x} \in [k]$, for every $\hat{w} \in [k]^{b-1}$, and for every $i \in [b-1]$, we have that $\phi (\hat{x} \hat{w})_i = \phi_k (\hat{w})_i$.

Now we can describe the reduction.

Lemma 2.5. For $k \geq 2$ and a function $f : [n] \times [k] \rightarrow \mathbb{N}$, there is a complete bipartite graph $G = (V_1 \cup V_2, E, w)$ such that

— for every $x \in X_f$, there exists a perfect matching $M$ of $G$ such that $w (M) = x$,
— for every perfect matching $M$ of $G$ we have $w (M) \in X_f$, and
— $|V_1| = |V_2| = O (\frac{nk^b \log k}{\log n + \log k})$.

Before the proof, we present an informal idea. For some number $b$, we define two sets of vertices of our bipartite graph as $V_1 = [k]^b$ and $V_2 = [k]^b$—that is, the sets of words of length $b$. The number $b$ is chosen in such a way that the sets $V_1$ and $V_2$ are small enough, and at the same time, $b$ is large enough so that the total number of occurrences of the character $\hat{1}$ in all words in $V_1$ is at least $n$. Each such occurrence of $\hat{1}$ can be used to encode the values of $f$ for one fixed first argument, and therefore we need at least $n$ of them. Then if $\hat{1}$ is the $i$-th character of some word $v_1 \in V_1$ and $c$ is the $i$-th character of some word $v_2 \in V_2$, then if this occurrence of $\hat{1}$ encodes the values of $f$ for some fixed first argument $j$, we add $f (j, c)$ to the weight of the corresponding edge in our bipartite graph. Now we can proceed to the formal proof of the lemma.

Proof. Let us consider the smallest $b \in \mathbb{N}_+$ such that $c = b \cdot k^{b-1} \geq n$. Later, we will show that $|V_1| = |V_2| = k^b$ is sufficient. Throughout the proof, $|V_1| = |V_2| = k^b$.

For convenience, we extend our chosen function $f : [n] \times [k] \rightarrow \mathbb{N}$ to $f : [c] \times [k] \rightarrow \mathbb{N}$ in such a way that for every $i = n + 1, n + 2, \ldots, c$ and for every $j \in [k]$, we put $f (i, j) = 0$. Note that the $f$-family $X_f$ does not change after this extension.

Let $V_1 = [k]^b$ and $V_2 = [k]^b$ be the sets of words of length $b$ over the alphabets $[k]$ and $[k]$, respectively. Note that $|V_1| = |V_2| = k^b$. Let $\beta : V_1 \times [b] \rightarrow [c] \cup \{\perp\}$ be any function such that if $\hat{w}_j \neq \hat{1}$, then $\beta (\hat{w}, j) = \perp$, and every value from the set $[c]$ is used exactly once—that is, for every $x \in [c]$, there is exactly one argument $(\hat{w}, j) \in V_1 \times [b]$ such
that $\beta(\hat{w}, j) = x$. Note that such a function always exists because the total number of the occurrences of $\hat{1}$ in all words in $V_1$ is exactly $c = b \cdot k^{b-1}$.

Now we define our weight function $w : V_1 \times V_2 \rightarrow \mathbb{N}$ as follows:

$$w(\hat{t}, u) = \sum_{i \in [b]} f(\beta(\hat{t}, i), u_i).$$

An example of such a weight function can be found in Figure 3 (or in Figure 4 as a picture of a bipartite graph).

Note that because $\beta$ picks every value from the set $[c]$ exactly once, then $\beta(\hat{t}, i) \rightarrow \phi(\hat{t}_i)$ defines a function, say $\sigma$, from $[c]$ to $[k]$. Then for every permutation $\phi : V_1 \rightarrow V_2$, we have

$$\sum_{t \in V_1} w(\hat{t}, \phi(\hat{t})) = \sum_{t \in V_1} \sum_{i \in [b]} f(\beta(\hat{t}, i), \phi(\hat{t}_i)) \in X_f.$$ 

In other words, the set of the weights of all perfect matchings in $G$ is a subset of $X_f$ as required.

We also need to show that for every $x \in X_f$, there exists some permutation $\phi : V_1 \rightarrow V_2$ such that $\sum_{t \in V_1} w(\hat{t}, \phi(\hat{t})) = x$. This permutation gives us a corresponding perfect matching of weight $x$ in $G$. Let us take a function $\sigma : [c] \rightarrow [k]$ such that $x = \sum_{i \in [c]} f(i, \sigma(\hat{t}_i))$, which exists by the definition of $X_f$. Define $\alpha : [k]^b \times [b] \rightarrow [k] \cup \{\perp\}$ as follows:

$$\alpha(\hat{u}, i) = \begin{cases} \sigma(\beta(\hat{u}, i)) & \text{for } \hat{u}_i = \hat{1} \\ \perp & \text{for } \hat{u}_i \neq \hat{1}. \end{cases}$$

Now we can use Lemma 2.4 with function $\alpha$ to obtain a permutation $\phi : [k]^b \rightarrow [k]^b$ such that for every $\hat{u} \in [k]^b$ and for every $i \in [b]$, if $\hat{u}_i = \hat{1}$, then $\phi(\hat{u})_i = \sigma(\beta(\hat{u}, i))$. Thus, we have that

$$\sum_{\hat{u} \in [k]^b} w(\hat{u}, \phi(\hat{u})) = \sum_{\hat{u} \in [k]^b} \sum_{i \in [b]} f(\beta(\hat{u}, i), \sigma(\beta(\hat{u}, i))) = \sum_{i \in [b]} f(i, \sigma(i)) = x.$$
Fig. 4. Graph encoding an $f$-family for $f : [4] \times [2] \to \mathbb{N}$. The lower indices $(1), (2), (3),$ and $(4)$ are added to indicate the correspondence between the occurrences of $\hat{1}$ and the elements of $[n]$ (the first argument of the function $f$). The lower indices $(i)$ and $(ii)$ are added to indicate the correspondence between the second argument of the function $f$ and the position in the (two element) sequence $(\cdot, \cdot)$.

Hence, we have shown that $X_f$ is the set of weights of all perfect matchings in graph $G$. The last thing is to show that the number of vertices is sufficiently small. Since $bk^b \geq nk$, we must have $b > \log \frac{nk}{\log(nk)} \sim \log nk$. Then, since $(b - 1)k^{b-2} < n$, we must have $k^b < nk^2/(b - 1) \leq (1 + o(1))nk^2/\log(nk) = O\left(\frac{nk^2}{\log(nk)}\right) = O\left(\frac{nk^2 \log k}{\log n + \log k}\right)$. Therefore, $|V_1| = |V_2| = k^b = O\left(\frac{nk^2 \log k}{\log n + \log k}\right)$, as required. □

Lemma 2.5 immediately implies the following result.

**Lemma 2.6.** There is a polynomial time reduction that for an instance $I = (f, g)$ of FAMILY INTERSECTION with $f : [a] \times [b] \to \mathbb{N}$ and $g : [c] \times [d] \to \mathbb{N}$ reduces it into an instance of EQUAL WEIGHT MATCHINGS $J = (G_1, G_2)$ with $|V(G_1)| = O\left(\frac{ab^2 \log b}{\log a + \log b}\right)$ and $|V(G_2)| = O\left(\frac{cd^2 \log d}{\log c + \log d}\right)$ vertices. The sets of the weights of all perfect matchings in $G_1$ and in $G_2$ are respectively equal to $X_f$ and $X_g$.

Together with Lemma 2.2, we obtain the following theorem.

**Theorem 2.7.** There is a polynomial time reduction from a given instance of 3-CNF-SAT with $n$ variables and $m$ clauses into an instance of EQUAL WEIGHT MATCHINGS with $|V(G_1)| = O\left(\frac{n}{\log n}\right)$, $|V(G_2)| = O\left(\frac{m}{\log m}\right)$ and the maximum matching weights bounded by $2^{3m}$. 

ACM Transactions on Algorithms, Vol. 12, No. 4, Article 48, Publication date: September 2016.
A commonly know corollary of the sparsification lemma of Impagliazzo et al. [2001] is as follows.

**Corollary 2.8.** There is no algorithm solving 3-CNF-SAT in $2^{o(n+m)}$-time where $n$ is the number of variables and $m$ is the number of clauses unless ETH fails.

Using Corollary 2.8, we can prove the following lower bound.

**Corollary 2.9.** There is no algorithm solving Equal Weight Matchings in $2^{o(n \log n)} \text{poly}(r)$-time where $n$ is the total number of vertices and $r$ is the bit size of the input unless ETH fails.

**Proof.** For a given instance of 3-CNF-SAT with $n$ variables and $m$ clauses, we can use the reduction from Theorem 2.7 to obtain an instance of Equal Weight Matchings. The total number of vertices in the new instance is

$$|V(G_1)| + |V(G_2)| = O\left(\frac{n}{\log n} + \frac{m}{\log m}\right)$$

because the function $\frac{n}{\log n}$ is nondecreasing for the sufficiently big values of $n$. Weights of the matchings are bounded by $2^{3m}$ and the bit size of the instance

$$r = O\left(\left(\frac{n + m}{\log(n + m)}\right)^2 \log 2^{3m}\right) = \text{poly}(nm).$$

Next, let us assume that there is an algorithm solving Equal Weight Matchings in $2^{o(n \log n)} \text{poly}(r)$-time. Then we could solve our instance of 3-CNF-SAT in time

$$2^{o\left(\frac{n + m}{\log(n + m)}\right) \log(\frac{n + m}{\log(n + m)}) \text{poly}(nm)}$$

$$= 2^{o\left(\frac{n + m}{\log(n + m)} \log(n + m)\right) \text{poly}(nm)} = 2^{o(n + m)},$$

which contradicts the ETH by Corollary 2.8. $lacksquare$

3. HARDNESS OF CHANNEL ASSIGNMENT

Consider two weighted complete bipartite graphs $G_1$ and $G_2$. We would like to encode them in a Channel Assignment instance in such a way that this Channel Assignment instance is a YES-instance if and only if there are two perfect matchings, one in $G_1$ and the other in $G_2$, of the same weight.

Because Channel Assignment is a natural generalization of a classical graph coloring problem, we will refer to the assignments as colorings. It is convenient because we then can refer to the value assigned to a vertex $v$ as to the color of $v$.

Consider an instance $I = (V, d, s)$ of Channel Assignment, where $d$ is our weight function and $s$ is a maximum allowed span. We say that $c : V \rightarrow \mathbb{Z}$ is a YES-coloring if $c$ is a proper coloring and has span at most $s$. Note that an instance of Channel Assignment is a YES-instance if and only if it has a YES-coloring.

Our approach is that we encode those graphs $G_1$ and $G_2$ separately in such a way that we have a special vertex $v_M$ whose color in every YES-coloring represents a weight of some perfect matching in $G_1$, and on the other hand, in every YES-coloring its color represents (in a similar way) a weight of some perfect matching in $G_2$. Thus, a YES-coloring would be possible if and only if the graphs $G_1$ and $G_2$ have two perfect matchings, one in $G_1$ and the other in $G_2$, with equal weights.
Before we present a way to encode a weighted complete bipartite graph in a Channel Assignment instance, we would like to present the two lemmas to merge those two encoded graphs into a one instance of Channel Assignment. To do this, we use the following concepts.

We say that instance \( I \) is \((x, y)\)-spanned for some vertices \( x, y \in V \) if for every YES-coloring \( c \) of \( I \), we have \( |c(x) - c(y)| = s - 1 \). We say that an instance \( I = (V, d, s) \) of Channel Assignment is \((X, Y)\)-spanned for some nonempty subsets of vertices \( \emptyset \neq X, Y \subseteq V \) if it is \((x, y)\)-spanned for every two vertices \( x \in X \) and \( y \in Y \).

**Lemma 3.1.** For every \((u, v)\)-spanned instance \( I_1 = (V_1, d_1, s) \) and \((w, z)\)-spanned instance \( I_2 = (V_2, d_2, s) \) of Channel Assignment, there is a \((\{u, w\}, \{v, z\})\)-spanned instance \( I = (V_1 \cup V_2, d, s) \) of Channel Assignment such that

(i) for every YES-coloring \( c \) of \( I \), the coloring \( c \mid_{V_1} \) is a YES-coloring of \( I_1 \) and the coloring \( c \mid_{V_2} \) is a YES-coloring of \( I_2 \), and

(ii) for every YES-coloring \( c_1 \) of \( I_1 \) and every YES-coloring \( c_2 \) of \( I_2 \) such that \( c_1(u) = c_2(w) \), \( c_1(v) = c_2(z) \) and for every \( x \in V_1 \cap V_2 \), we have \( c_1(x) = c_2(x) \) and there exists a YES-coloring \( c \) of \( I \) such that \( c \mid_{V_1} = c_1 \) and \( c \mid_{V_2} = c_2 \).

**Proof.** Let \( B = \{u, w\} \times \{v, z\} \cup \{v, z\} \times \{u, w\} \), and let

\[
d(x, y) = \begin{cases} 
\frac{s - 1}{2} & \text{if } (x, y) \in B \\
\max\{d_1(x, y), d_2(x, y)\} & \text{if } x, y \in V_1 \cap V_2 \\
d_1(x, y) & \text{if } x, y \in V_1 \text{ and } (x, y) \not\in V_1 \cap V_2 \\
d_2(x, y) & \text{if } x, y \in V_2 \text{ and } (x, y) \not\in V_1 \cap V_2 \\
0 & \text{otherwise.}
\end{cases}
\]

Our instance \((\{u, w\}, \{v, z\})\)-spanned because for all pairs in \( B \), we set the minimum allowed distance to at least \( s - 1 \). Note that for \( i = 1, 2 \), for every \( x, y \in V_i \) we have \( d(x, y) \geq d_i(x, y) \). Hence, every proper coloring \( c \) of \( I \) has the property that \( c \mid_{V_i} \) is a proper coloring of \( I_1 \) and \( c \mid_{V_2} \) is a proper coloring of \( I_2 \). In addition, the maximum allowed spans of \( I, I_1, I_2 \) are the same, and thus for every YES-coloring \( c \) of \( I \) coloring \( c \mid_{V_1} \) is a YES-coloring of \( I_1 \) and \( c \mid_{V_2} \) is a YES-coloring of \( I_2 \). Hence, (i) is clear.

For (ii), consider a YES-coloring \( c_1 \) of \( I_1 \) and a YES-coloring \( c_2 \) of \( I_2 \) such that \( c_1(u) = c_2(w) \) and \( c_1(v) = c_2(z) \) and such that for every \( x \in V_1 \cap V_2 \), we have \( c_1(x) = c_2(x) \). Then we define a coloring \( c \):

\[
c(x) = \begin{cases} 
c_1(x) & \text{if } x \in V_1 \\
c_2(x) & \text{if } x \in V_2.
\end{cases}
\]

We know that \( c_1(u) = c_2(w) \) and \( c_1(v) = c_2(z) \), so all vertices of \( V_1 \cup V_2 \) have colors between \( c_1(u) \) and \( c_1(v) \)—that is, the span of \( c \) is at most \( s \), as required. It is straightforward to check that \( c \) is a proper coloring.

**Lemma 3.2.** For every \((u_L, v_R)\)-spanned instance \( I = (V, d, s) \) of Channel Assignment and for every numbers \( l, r \in \mathbb{N} \), there exists a \((w_L, w_R)\)-spanned instance \( I' = (V \cup \{w_L, w_R\}, d', l + s + r) \) such that

(i) for every YES-coloring \( c \) of \( I \), there is a YES-coloring \( c' \) of \( I' \) such that \( c' \mid_{V} = c \), and

(ii) for every YES-coloring \( c' \) of \( I' \) such that \( c'(w_L) \leq c'(w_R) \), we have that

— a coloring \( c' \mid_{V} \) is a YES-coloring of \( I \), and

— \( c'(u_L) = c'(w_L) + l \) and \( c'(v_R) = c'(w_R) - r \).
there exists a perfect matching \( w \) with \( |V_1| = |V_2| = 2 \) encoded in a \text{Channel Assignment} form. The color of the vertex \( v_M = w_2 \) corresponds to the weight of the perfect matching in \( G \) given by the permutation \( \pi \) and is equal to \( c(v_M) = c(v_L) + 7M + w(M_i) \). The picture is simplified. Some of the edges and corresponding minimum distances are omitted in the picture.

**Proof.** We assume that \( w_L, w_R \notin V \). We put

\[
d'(x, y) = \begin{cases} 
  l + s - 1 + r & \text{for } \{x, y\} = \{w_L, w_R\} \\
  l & \text{for } \{x, y\} \cap \{w_L, w_R\} = \{w_L\} \\
  r & \text{for } \{x, y\} \cap \{w_L, w_R\} = \{w_R\} \\
  d(x, y) & \text{for } x, y \in V.
\end{cases}
\]

It is straightforward to check that \( d' \) satisfies (i) and (ii).

**Lemma 3.3.** Let \( G = (V_1 \cup V_2, E, w) \) be a weighted complete bipartite graph with nonnegative weights and such that \( |V_1| = |V_2| \). Let \( n = |V_1|, m = \max_{e \in E} w(e), M = n \cdot m + 1, l = (4n - 1) \cdot M, \) and \( s = (8n - 1) \cdot M \). There exists a \((g_1, v_R)\)-spanned instance \( I = (V, d, s) \) of \text{Channel Assignment} with \( |V| = O(n) \), and such that for some vertex \( v_M \in V \),

(i) for every YES-coloring \( c \) of \( I \) such that \( c(v_L) \leq c(v_R) \), there exists a perfect matching \( M_G \) in \( G \) such that \( c(v_M) = c(v_L) + l + w(M_G) \), and

(ii) for every perfect matching \( M_G \) in \( G \), there exists a YES-coloring \( c \) of \( I \) such that \( c(v_L) \leq c(v_R) \) and \( c(v_M) = c(v_L) + l + w(M_G) \).

**Proof.** Let \( V_1 = \{v_1^{(1)}, v_2^{(1)}, \ldots, v_n^{(1)}\} \) and \( V_2 = \{v_1^{(2)}, v_2^{(2)}, \ldots, v_n^{(2)}\} \). We will build our \text{Channel Assignment} instance step by step. A simplified picture of the instance can be found in Figure 5. The outline of the proof as follows:

—We will define the vertices of our \text{Channel Assignment} instance and will force some properties of its colorings by putting appropriate weights on the edges. We specify those properties as Claims.

—We start with defining the vertices \( v_L = v_1, v_2, \ldots, v_{4n} = v_R \). These vertices will be used as a backbone of our \text{Channel Assignment} instance—that is, we force them to be colored in a strictly ascending way (or strictly descending, but we assume the ascending order without loss on generality). All of the following groups of vertices that we will add later will be colored in relation to the colors of \( v_1, v_2, \ldots, v_{4n} \).

**Claim 1.** For every YES-coloring \( c \) and for every \( i < j \), we have that \( c(v_i) < c(v_j) \).

—Then we add the vertices \( w_1, w_2, \ldots, w_{2n-1} \) and interleave them with the vertices \( v_1, v_2, \ldots, v_{4n} \).

**Claim 2.** For every YES-coloring \( c \), the colors of vertices in the sequence

\[
v_1, v_2, w_1, v_3, v_4, w_2, v_5, \ldots, v_{4n-2}, w_{2n-1}, v_{4n-1}, v_{4n}.
\]

are increasing.
Next we add the vertices \( a_1, a_2, \ldots, a_n \) and interleave them with the previous vertices, but in an arbitrary order \( \pi_c \).

**Claim 3.** For every YES-coloring \( c \), there is a permutation \( \pi_c \) such that the colors of vertices of the sequence
\[
v_1, a_{\pi_c(1)}, v_2, w_1, v_3, a_{\pi_c(2)}, v_4, w_2, v_5, \ldots, v_{2n-1}, a_{\pi_c(n)}, v_{2n}
\]
are increasing.

—Similarly, we add vertices \( b_1, b_2, \ldots, b_n \) in an arbitrary order \( \rho_c \).

**Claim 4.** For every YES-coloring \( c \), there is a permutation \( \rho_c \) such that the colors of vertices in the sequence
\[
v_{2n+1}, b_{\rho_c(1)}, v_{2n+2}, w_{n+1}, v_{2n+3}, b_{\rho_c(2)}, v_{2n+4}, w_{n+2},
\]
\[
v_{2n+5} \ldots v_{4n-1}, b_{\rho_c(n)}, v_{4n}
\]
are increasing.

—Then we force that the vertices \( a_1, a_2, \ldots, a_n \), and the vertices \( b_1, b_2, \ldots, b_n \) are colored in exactly the same (arbitrary) order—that is, the order \( \pi_c \) is exactly the same as the order \( \rho_c \).

**Claim 5.** For every YES-coloring \( c \), there is a permutation \( \pi_c \) such that the colors of vertices in the sequence
\[
v_1, a_{\pi_c(1)}, v_2, w_1, v_3, \ldots, v_{2n-1}, a_{\pi_c(n)}
\]
\[
v_{2n}, w_n, v_{2n+1}, b_{\pi_c(1)}, v_{2n+2}, w_{n+1}, v_{2n+3}, b_{\pi_c(2)}, v_{2n+4},
\]
\[
w_{n+2}, v_{2n+5} \ldots v_{4n-1}, b_{\pi_c(n)}, v_{4n}
\]
are increasing.

—Then, with all of the preceding constructions specified, we state the dependency between the set of all possible colors of the vertex \( v_M \) and the set of the possible weights of the matchings in our bipartite graph.

**Claim 6.** Let \( \pi : [n] \to [n] \) be any permutation and \( M_\pi = \{ v_i^{(1)} v_i^{(2)} : i \in [n] \} \) be the corresponding perfect matching in \( G \). There is exactly one YES-coloring \( c \) such that \( \pi_c = \pi \). Moreover, \( c(v_M) = c(v_L) + l + w(M_\pi) \).

Now we can present the proof in a formal way. Let us introduce the vertices \( v_L = v_1, v_2, \ldots, v_{4n} = v_R \) to set \( V \). Because of the symmetry of colorings, we can assume that for every coloring \( c \) of our instance, we have \( c(v_L) \leq c(v_R) \). Indeed, we know that for every coloring \( c \), a symmetric coloring \( c'(v) = 1 + \text{span}(c) - c(v) \) has the same span as \( c \) and is proper if and only if \( c \) is proper.

We set the minimum distance \( d(v_L, v_R) = s - 1 \). Then for every YES-coloring \( c \), we have that \( |c(v_L) - c(v_R)| = s - 1 \), so our instance is \( (v_L, v_R) \)-spanned. For every \( i, j \in [4n] \), such that \( i \neq j \) and \( |i - j| \neq 1, 4n \), we set the minimum distance \( d(v_i, v_j) = |i - j| \cdot 2M \). Then we can prove the following claim.

**Claim 1.** For every YES-coloring \( c \) and for every \( i < j \), we have that \( c(v_i) < c(v_j) \).

**Proof of the Claim:** We have assumed without loss of generality that for all colorings, we have \( c(v_L) \leq c(v_R) \). Note that for \( i \neq j \), we have \( c(v_i) \neq c(v_j) \). If there are \( i < j \) such that \( c(v_j) < c(v_i) \), then \( c(v_R) - c(v_L) = c(v_R) - c(v_j) + c(v_j) - c(v_i) = c(v_j) - c(v_L) \geq 2M \times ((4n - i) + (j - i) + (j - i)) \geq 2M \times (4n - 1 + 2(j - i)) \geq 2M \times (4n - 1 + 2M) > s \), a contradiction. This proves the claim.

Note that for every YES-coloring and for every \( i \in [4n, -1] \), we have
\[
2M \leq c(v_{i+1}) - c(v_i) \leq 2M + n \cdot m < 3M, \tag{1}
\]
and otherwise \( c(v_R) - c(v_L) \geq (4n - 2) \cdot 2M + 2M + n \cdot m + 1 = s \), so \( c \) has span at least \( s + 1 \), a contradiction.
Let us introduce new vertices \( w_1, w_2, \ldots, w_{2n-1} \) to set \( V \). For every \( i \in [2n-1] \), and \( j \in [4n] \), we set the minimum distances \( d(w_i, v_j) = |4i + 1 - 2j| \cdot M \). For every YES-coloring \( c \) and for every \( i \in [2n-1] \), we have \( c(v_{2i}) + M \leq c(v_{2i+1}) - M \) by (1), and otherwise we have that \( c(v_j) \leq c(w_i) \leq c(v_{j+1}) \) for some \( j \neq 2i \) (because \( c(v_{2i}) - c(v_1) = s - 1 \), so every YES-coloring uses only colors from the interval \([c(v_1), c(v_{2i+1}])\)), and then \( c(v_{j+1}) - c(v_j) \geq d(v_j, w_i) + d(w_i, v_{j+1}) \) and \( \{v_j, v_{j+1}\} \neq \{v_{2i}, v_{2i+1}\} \), so at least one of these two distances is at least \( 3M \), and therefore \( c(v_{j+1}) - c(v_j) \geq 3M + M > 3M \), a contradiction with (1). Thus, infer the following claim.

**Claim 2.** For every YES-coloring \( c \), the colors of vertices in the sequence

\[
v_1, v_2, w_1, v_3, v_4, w_2, v_5, \ldots, v_{4n} - 2, w_{2n-1}, v_{4n-1}, v_{4n}.
\]

are increasing.

We introduce new vertices \( a_1, a_2, \ldots, a_n \), and for every \( i \in [n] \), and \( j \in [4n] \), we set the minimum distances

\[
d(a_i, v_j) = \begin{cases} M + w(v^{(1)}_i, v^{(2)}_j) & \text{when } j \leq 2n \text{ and } 2 \nmid j \\ M & \text{when } j \leq 2n \text{ and } 2 \mid j \\ (j - 2n) \cdot 2M + M & \text{when } j > 2n. \end{cases}
\]

Then for every YES-coloring \( c \) and for every \( i \in [n] \), we have \( c(a_i) \leq c(v_{2n}) \) because in the other case, we have \( c(v_j) \leq c(a_i) \leq c(v_{j+1}) \) for some \( j \geq 2n \) and then \( c(v_{j+1}) - c(v_j) \geq d(v_j, a_i) + d(a_i, v_{j+1}) \geq M + 3M > 3M \), a contradiction with (1).

Moreover, for every \( i \in [n] \), and every \( j \in [2n-1] \), we set the minimum distance \( d(a_i, w_j) = 2M \). Therefore, by (1) and (2) for every YES-coloring \( c \) and every \( i \in [n] \), the vertex \( a_i \) is colored with the color from one of the intervals \((c(v_{2i-1}), c(v_{2i}))\) for some \( j \in [n] \).

Finally, for every \( i, j \in [n] \), such that \( i \neq j \), we set the minimum distance \( d(a_i, a_j) = 4M \), so by (1) we know that for every YES-coloring \( c \) and every \( i \in [n] \), exactly one vertex \( a_j \) of vertices \( a_1, a_2, \ldots, a_n \) is colored with the color from the interval \((c(v_{2i-1}), c(v_{2i}))\). The assignment of vertices \( a_1, a_2, \ldots, a_n \) to intervals \((c(v_1), c(v_2)), (c(v_3), c(v_4)), \ldots, (c(v_{2n-1}), c(v_{2n}))\) determines a permutation \( \pi_c : [n] \rightarrow [n] \)—that is, \( \pi_c(i) = j \) if \( a_j \) gets a color from \((c(v_{2i-1}), c(v_{2i}))\). Hence, we get the following claim.

**Claim 3.** For every YES-coloring \( c \), there is a permutation \( \pi_c \) such that the colors of vertices of the sequence

\[
v_1, a_{\pi_c(1)}, v_2, w_1, v_3, a_{\pi_c(2)}, v_4, w_2, v_5, \ldots, v_{2n-1}, a_{\pi_c(n)}, v_{2n}
\]

are increasing.

Similarly, we introduce new vertices \( b_1, b_2, \ldots, b_n \), and for every \( i \in [n] \) and \( j \in [4n] \), we set the minimum distances

\[
d(b_i, v_j) = \begin{cases} (2n - j + 1) \cdot 2M + M & \text{when } j \leq 2n \\ M + m - w(v^{(1)}_i, v^{(2)}_j/2) & \text{when } j > 2n \text{ and } 2 \nmid j \\ M & \text{when } j > 2n \text{ and } 2 \mid j. \end{cases}
\]

In addition, for every \( i \in [n] \) and every \( j \in [2n-1] \), we set the minimum distance \( d(b_i, v_j) = 2M \), and for every \( i, j \in [n] \) such that \( i \neq j \), we set the minimum distance \( d(b_i, b_j) = 4M \). Hence, similarly as before, the colors of vertices \( b_1, b_2, \ldots, b_n \) determine a permutation \( \rho_c : [n] \rightarrow [n] \). Thus, we have the following claim.
Claim 4. For every YES-coloring $c$, there is a permutation $\pi_c$ such that the colors of vertices in the sequence
\[
v_{2n+1}, b_{\pi(1)}, v_{2n+2}, w_{n+1}, v_{2n+3}, b_{\pi(2)}, v_{2n+4}, w_{n+2},
v_{2n+5} \ldots v_{4n-1}, b_{\pi(n)}, v_{4n}
\]
are increasing.

For every $i \in [n]$, we set the minimum distance $d(a_i, b_i) = n \cdot 4M$. Then we know that for every YES-coloring $c$, we have $\pi_c^{-1}(i) \leq \rho_c^{-1}(i)$, and otherwise we can take $j = 2\pi_c^{-1}(i) - 1$ and $k = 2n + 2\pi_c^{-1}(i)$ and then $(c(b_i) - c(a_i)) + 2M \leq c(v_j) - c(v_k)$ and $k - j \leq 2n$, so the sequence $v_1, v_2, \ldots, v_j, v_{k}, \ldots, v_{4n}$, has at least $4n, 2n + 1 = 2n + 1$ elements so $c(v_{4n}) - c(v_1) \geq (2n - 1) \cdot 2M + (c(v_k) - c(v_j)) \geq (2n - 1) \cdot 2M + (c(b_i) - c(a_i)) + 2M \geq (2n - 1) \cdot 2M + n \cdot 4M + 2M - 8M > (n - 1) \cdot 8M - 1 - s$, a contradiction.

Since $\pi_c$ and $\rho_c$ are permutations, we further infer that for every YES-coloring $c$, we have $\pi_c = \rho_c$. Hence, we have the following claim.

Claim 5. For every YES-coloring $c$, there is a permutation $\pi_c$ such that the colors of vertices in the sequence
\[
v_1, a_{\pi(1)}, v_2, w_1, v_3, a_{\pi(2)}, v_4, w_2, v_5 \ldots, v_{n-1}, a_{\pi(n)},
v_1, a_{\pi(1)}, v_2, w_1, v_3, a_{\pi(2)}, v_4, w_2, v_5 \ldots, v_{n-1}, a_{\pi(n)},
\]
are increasing.

This ends the description of the instance $I$. Note that $I$ is $(v_L, v_R)$-spanned because $d(v_L, v_R) = s - 1$. Let us put $v_M = w_n$. We are going to show the following claim.

Claim 6. Let $\pi : [n] \rightarrow [n]$ be any permutation and $M \pi = \{v^{(1)}_{\pi(i)} : i \in [n]\}$ be the corresponding perfect matching in $G$. There is exactly one YES-coloring $c$ such that $\pi_c = \pi$. Moreover, $c(\pi M) = c(v_L) + 1 + w(M \pi)$.

Proof of the Claim: Let us consider a sequence of vertices
\[
v_1, a_{\pi(1)}, v_2, w_1, v_3, a_{\pi(2)}, v_4, w_2, v_5 \ldots, v_{n-1}, a_{\pi(n)},
v_1, a_{\pi(1)}, v_2, w_1, v_3, a_{\pi(2)}, v_4, w_2, v_5 \ldots, v_{n-1}, a_{\pi(n)},
\]
and the coloring $c$ implied by the minimum distances of pairs of consecutive elements in this sequence—that is, $c(v_1) = 1, c(a_{\pi(1)}) = c(v_1) + d(v_1, a_{\pi(1)}), c(v_2) = c(a_{\pi(2)}) + d(a_{\pi(2)}, v_2), c(w_1) = c(v_2) + d(v_2, w_1), c(v_3) = c(w_1) + d(w_1, v_3), \ldots, c(v_{4n}) = c(b_{\pi(n)}) + d(b_{\pi(n)}, v_{4n})$. We need to check that all minimum distance constraints $d$ are satisfied and that the span of this coloring is not greater than $s$.

Note that for every $i \in [4n]$, and for every vertex $x \in V$ such that $v_i \neq x$, we have $d(x, v_i) \geq M$. Therefore, for every $i \in [4n - 1]$, we have $c(v_{i+1}) - c(v_i) = (c(v_{i+1}) - c(x)) + (c(x) - c(v_i)) = d(x, v_{i+1}) + d(v_i, x) \geq 2M$, where $x$ is the vertex separating $v_i$ and $v_{i+1}$ in the sequence. Thus, for every $i, j \in [4n]$, we have $|c(v_i) - c(v_j)| \geq |i - j| \cdot 2M$, so if $i, j \neq 1, 4n$, then $|c(v_i) - c(v_j)| \geq d(v_i, v_j)$. Hence, also for every $i \in [2n - 1]$, and $j \in [4n]$, we have $|c(w_i) - c(v_j)| = |c(w_i) - c(v_k)| + |c(v_k) - c(v_j)| \geq M + |k - j| \cdot 2M$, where in case of $j \leq 2i$, we have $k = 2i + 1$ and in this case $M + |k - j| \cdot 2M = |4i - 2j + 1| \cdot M$, and in case of $j > 2i$, we have $k = 2i + 1$ and in this case $M + |k - j| \cdot 2M = |2j - 4i - 1| \cdot M = |4i - 2j + 1| \cdot M$. Thus, in both cases, $|c(w_i) - c(v_j)| \geq |4i - 2j + 1| \cdot M = d(w_i, v_j)$. We will check the distance between $v_L = v_1$ and $v_R = v_{4n}$, later.

For every $i \in [n]$, and vertex $a_{\pi(i)}$, the closest vertex $v_j$ to the left is $v_{2i-1}$ and to the right is $v_{2i}$. They are immediate neighbors of $a_{\pi(i)}$ in the sequence, so from the definition
of $c$, we have $|c(a_{\pi(i)}) - c(v_{2i-1})| = d(v_{2i-1}, a_{\pi(i)})$ and $|c(v_2) - c(a_{\pi(i)})| = d(a_{\pi(i)}, v_2)$. Note that for every $j \in [2n]$, we have $d(a_{\pi(i)}, v_j) \leq 2M$, and for every $j \in [2i-2]$, we have $|c(a_{\pi(i)}) - c(v_j)| = (c(v_2) - c(a_{\pi(i)})) + c(v_j) \leq 2M + M > d(a_{\pi(i)}, v_{2i-1})$. Similarly, for every $2i + 1 \leq j \leq 2n$, we have $|c(a_{\pi(i)}) - c(v_j)| = (c(v_2) - c(a_{\pi(i)})) + c(v_j) - c(v_{2i}) \geq M + 2M > d(a_{\pi(i)}, v_j)$. For every $2n + 1 \leq j \leq 4n$, we have $|c(v_j) - c(a_{\pi(i)})| = (c(v_2) - c(a_{\pi(i)})) + (c(v_2n) - c(v_{2i})) + (c(v_j) - c(v_{2n})) \geq M + 2M + 2 = d(a_{\pi(i)}, v_j)$. Because $\pi$ is a permutation, thus we obtain that for every $i \in [n]$, and for every $j \in [4n]$, we have $|c(a_i) - c(v_j)| \geq d(a_i, v_j)$.

For every $i \in [n]$, and $j \in [2n-1]$, there is at least one vertex $v_k$ with color between the colors $c(a_i)$ and $c(v_j)$ so $|c(a_i) - c(v_j)| = |c(a_i) - c(v_k)| + |c(v_k) - c(v_j)| \geq 2M = d(a_i, v_j)$. For every $i, j \in [n]$, such that $\pi^{-1}(i) < \pi^{-1}(j)$, there are at least two vertices $v_k, v_{k+1}$ with colors $c(a_i) \leq c(v_k) \leq c(v_{k+1}) \leq c(a_j)$. Therefore, $|c(a_i) - c(a_j)| = (c(v_k) - c(a_i)) + (c(v_{k+1}) - c(v_k)) + (c(a_j) - c(v_{k+1})) \geq M + 2M + M = 4M = d(a_i, a_j)$.

Similarly, we can check that for every $i \in [n]$ and $j \in [4n]$, we have $|c(b_i) - c(v_j)| \geq d(b_i, v_j)$ such that for every $i \in [n]$ and $j \in [2n-1]$, we have $|c(b_i) - c(v_j)| \geq d(b_i, v_j)$, and for every $i, j \in [n]$ such that $i \neq b$, we have $|c(b_i) - c(b_j)| \geq d(b_i, b_j)$.

We need also to check that for every $i \in [n]$, we have $|c(a_i) - c(b_j)| \geq n \cdot 4M = d(a_i, b_j)$. Indeed, $|c(b_i) - c(a_i)| = |c(v_2) - c(v_2i) + (c(v_{2n+2i-1}) - c(v_2i)) + (c(b_i) - c(v_{2n+2i-1})) \geq M + (2n-1) \cdot 2M + M = n \cdot 4M = d(a_i, b_j)$.

Now we are going to deal with the distances between $v_L, v_M$, and $v_R$. The sum of the minimum color distances of neighboring elements in the prefix of our sequence,

$$v_1, a_{\pi(1)}, v_2, a_{\pi(2)}, v_4, a_{\pi(4)}, v_5, v_6, a_{\pi(n)}, v_2n, w_n, v_{2n+1}$$

is exactly $2n \cdot 2M + w(M_x)$. The sum of the minimum color distances of neighboring elements in the suffix of our sequence,

$$v_{2n+1}, b_{\pi(1)}, v_{2n+2}, a_{\pi(2)}, v_{2n+3}, b_{\pi(2)}, v_{2n+4}, w_{n+2}, v_{2n+5} \ldots v_{4n-1}, b_{\pi(n)}, v_{4n}$$

is exactly $(2n-1) \cdot 2M + n \cdot m - w(M_x)$. Therefore, the sum for the whole sequence is exactly $(4n-1) \cdot 2M + n \cdot m = s - 1$ and does not depend on the permutation $\pi$. Thus, $|c(v_R) - c(v_L)| = s - 1 = d(v_L, v_R)$. This was the last constraint to check, and hence we have shown that $c$ is proper. On the other hand, the span of $c$ is $s$, so $c$ is a YES-coloring. Moreover, we have $c(v_M) = c(v_L) + (4n-1) \cdot M + w(M_x) = c(v_L) + 1 + w(M_x)$. Note that all distances of pairs of consecutive elements of (the whole) sequence are tight—that is, these distances are equal to the minimum allowed distances for these pairs of vertices and thus we cannot decrease any of these distances. On the other hand, the span of $c$ is maximum, so we cannot increase any of these distances without exceeding the maximum span or violating some of the constraints provided by $d$. Therefore, $c$ is the only one YES-coloring for which the colors of the vertices of this sequence are increasing. Hence, $c$ is the only one YES-coloring such that $\pi_c = \pi$. This ends the proof of the claim.

Thus, there is a one-to-one correspondence between permutations and YES-colorings. Moreover, we know that for every YES-coloring $c$, we have $c(v_M) = c(v_L) + 1 + w(M_x)$, where $M_x$ is the perfect matching in $G$ corresponding to permutation $\pi_c$. Hence, we have shown (i) and (ii) as required. $\square$

**Lemma 3.4.** There is a polynomial time reduction such that for a given instance $I = (G_1, G_2)$ of **Equal Weight Matchings** with $n_1 = |V(G_1)|, n_2 = |V(G_2)|$ and such that the weight functions of $G_1$ and $G_2$ are bounded respectively by $m_1$ and $m_2$ reduces
Two weighted complete bipartite graphs \((G_1, v_1)\) (with \(n_1 = 2\)) and \((G_2, w_2)\) (with \(n_2 = 3\)) encoded in a Channel Assignment form. The color of the vertex \(v_M = w_2\) corresponds to the weight of some perfect matching in \(G_1\) and to the weight of some perfect matching in \(G_2\). These two weights have to be equal. The figure is simplified, and some of the edges are omitted. Note that the values \(M\) and \(m\) can be different for \((G_1, v_1)\) and for \((G_2, w_2)\).

It into an instance of Channel Assignment with \(O(n_1 + n_2)\) vertices and the maximum edge weight in \(O(n_1^2 m_1 + n_2^2 m_2)\).

In the proof, we use Lemma 3.3 to encode \(G_1\) and \(G_2\) in two instances of Channel Assignment, then we extend them to the common length using Lemma 3.2, and finally we merge them using Lemma 3.1. The simplified picture of the obtained Channel Assignment instance can be found in Figure 6.

Proof. Let us use Lemma 3.3 on graph \(G_1\) to obtain a \((v_1^{(1)}, v_R^{(1)})\)-spanned Channel Assignment instance \(I_1 = (V_1, d_1, s_1)\) with \(l_1 = O(n_1^2 m_1)\), \(s_1 = 2l_1 + n_1 \cdot m_1 = O(n_1^2 m_1)\) and with the vertex \(v_M^{(1)}\) (as in the statement of Lemma 3.3). The number of the vertices in \(V_1\) is \(O(n_1)\). Similarly, let \(I_2 = (V_2, d_2, s_2)\) be a \((v_1^{(2)}, v_R^{(2)})\)-spanned Channel Assignment instance with \(l_2 = O(n_2^2 m_2)\), \(s_2 = 2l_2 + n_2 \cdot m_2 = O(n_2^2 m_2)\) and with the vertex \(v_M^{(2)}\) obtained from Lemma 3.3 from graph \(G_2\). The number of vertices in \(V_2\) is \(O(n_2)\). Let us identify vertices \(v_M^{(1)}\) and \(v_M^{(2)}\) (i.e., \(v_M^{(1)} = v_M^{(2)} = v_M\) and \(V_1 \cap V_2 = \{v_M\}\)). In addition, let \(l_{\max} = \max\{l_1, l_2\} = O(n_1^2 m_1 + n_2^2 m_2)\) and \(s = l_{\max} + \max\{s_1 - l_1, s_2 - l_2\} = O(n_1^2 m_1 + n_2^2 m_2)\). Our span will be \(s\). Note that then every edge with a weight greater than \(s - 1\) forces such that our instance is a NO-instance. Thus, if we have an edge with a weight greater that \(s\), we can replace it with the same edge but with a weight equal to \(s\) and the instance will still be a NO-instance. Therefore, weights of all of our edges will be bounded by \(O(n_1^2 m_1 + n_2^2 m_2)\).

We can use Lemma 3.2 with \(l = l_{\max} - l_1\) and with \(r = s - (l_{\max} + s_1 - l_1)\) for extending the instance \(I_1\) into a \((v_1^{(1)}, v_R^{(1)})\)-spanned instance \(I'_1(V'_1 = V_1 \cup \{w_1^{(1)}, w_R^{(1)}, d'_1, s\})\) of Channel Assignment. For every YES-coloring \(c'_1\) of \(I'_1\), we know that \(c'_1|_{V'_1}\) is a YES-coloring of \(I_1\), and for every YES-coloring \(c_1\) of \(I_1\), there exists a YES-coloring \(c'_1\) of \(I'_1\) such that \(c'_1|_{V_1} = c_1\), so from the properties of \(I_1\) (obtained from Lemma 3.3), we know that

— for every YES-coloring \(c'_1\) of \(I'_1\) such that \(c'_1(w_1^{(1)}) \leq c'_1(w_R^{(1)})\), there exists a perfect matching \(M_1\) in \(G_1\) such that \(c'_1(v_M) = c'_1(w_1^{(1)}) + l_{\max} + w_1(M_1)\), and

— for every perfect matching \(M_1\) in \(G_1\), there exists a YES-coloring \(c'_1\) of \(I'_1\) such that \(c'_1(w_1^{(1)}) \leq (w_R^{(1)})\) and \(c'_1(v_M) = c'_1(w_1^{(1)}) + l_{\max} + w_1(M_1)\).

Similarly, we can use Lemma 3.2 with \(l = l_{\max} - l_2\) and with \(r = s - (l_{\max} - l_2 + s_2)\) for extending the instance \(I_2\) into a \((v_1^{(2)}, v_R^{(2)})\)-spanned instance \(I'_2(V'_2 = V_2 \cup \{w_1^{(2)}, w_R^{(2)}, d'_2, s\})\) of Channel Assignment such that

ACM Transactions on Algorithms, Vol. 12, No. 4, Article 48, Publication date: September 2016.
for every YES-coloring $c'_2$ of $I'_2$ such that $c'_2(w_L^{(2)}) \leq c'_2(w_R^{(2)})$, there exists a perfect matching $M_2$ in $G_2$ such that $c'_2(v_M) = c(w_L^{(2)}) + l_{\max} + w(M_2)$, and

— for every perfect matching $M_2$ in $G_2$, there exists a YES-coloring $c'_2$ of $I'_2$ such that $c'_2(w_L^{(2)}) \leq (w_R^{(2)})$ and $c'_2(v_M) = c'_1(w_L^{(2)}) + l_{\max} + w_1(M_1)$.

Now we can use Lemma 3.1 to merge the instances $I_1$ and $I'_2$ into a one $\{(w_L^{(1)}, w_L^{(2)}), (w_R^{(1)}, w_R^{(2)})\}$-spanned instance $I' = (V_1' \cup V_2', d, s)$. A simplified picture of the obtained instance can be found in Figure 6. Note that $V_1' \cap V_2' = \{v_M\}$, so

— for every YES-coloring $c$ of $I'$ such that $c(w_L^{(1)}) \leq c(w_R^{(1)})$, there exist perfect matchings $M_1$ in $G_1$ and $M_2$ in $G_2$ such that $c(v_M) = c(w_L^{(1)}) + l_{\max} + w_1(M_1) = c(w_L^{(1)}) + l_{\max} + w_2(M_2)$, so $w_1(M_1) = w_2(M_2)$, and

— for every two perfect matchings $M_1$ in $G_1$ and $M_2$ in $G_2$ such that $w_1(M_1) = w_2(M_2)$, there is a YES-coloring $c$ of $I'$ such that $c(w_L^{(1)}) \leq c(w_R^{(1)})$ and $c(v_M) = c(w_L^{(1)}) + l_{\max} + w_1(M_1)$.

Therefore, the CHANNEL ASSIGNMENT instance $I'$ has a YES-coloring if and only if there are two perfect matchings $M_1$ in $G_1$ and $M_2$ in $G_2$ such that $w_1(M_1) = w_2(M_2)$. By Lemmas 3.1 and 3.2, we know that $I'$ has $O(n_1 + n_2)$ vertices.

Now we can use the results of Section 2 to get the following two corollaries.

**Corollary 3.5.** There is no algorithm solving CHANNEL ASSIGNMENT in $2^{o(n \log n)}\text{poly}(r)$ where $n$ is the number of the vertices and $r$ is the bit size of the instance unless ETH fails.

**Proof.** For a given instance of EQUAL WEIGHT MATCHINGS with $n$ vertices and the weights bounded by $m$, we can transform it by Lemma 3.4 into an instance of CHANNEL ASSIGNMENT with $n' = O(n)$ vertices and the weights bounded by $\ell = O(n^2 m)$. Note that for the bit size $r'$ of the new instance, we have $\text{poly}(r') = \text{poly}((n')^2 \ell) = \text{poly}(n, m) = \text{poly}(r)$. Let us assume that we can solve CHANNEL ASSIGNMENT in $2^{o(n \log n)}\text{poly}(r)$-time. Then we can solve our instance in $2^{o(n' \log n')}\text{poly}(r') = 2^{o(n \log n)}\text{poly}(r)$, which contradicts ETH by Corollary 2.9. □

**Corollary 3.6.** There is no algorithm solving CHANNEL ASSIGNMENT in $2^{n o(\log \log \ell)}\text{poly}(r)$ where $n$ is the number of the vertices, $\ell$ is a maximum weight of the edge, and $r$ is the bit size of the instance unless ETH fails.

**Proof.** For a given instance of 3-CNF-SAT with $n$ variables and $m$ clauses, we use the reduction from Theorem 2.7 to obtain an instance of EQUAL WEIGHT MATCHINGS with $|V_1| = O(\frac{n}{\log n})$, $|V_2| = O(\frac{n}{\log m})$ and the maximum matching weights bounded by $2^m_3$. Then we use the reduction from Lemma 3.4 to obtain an instance of CHANNEL ASSIGNMENT with

$$n' = O\left(\frac{n}{\log n} + \frac{m}{\log m}\right) = O\left(\frac{n + m}{\log(n + m)}\right)$$

vertices and the weights on the edges bounded by

$$\ell = O\left(\left(\frac{n}{\log n} + \frac{m}{\log m}\right)^2 2^3\right).$$

Then $\log \ell = O(n + m)$ and $r = O((n')^2 \cdot \log \ell) = O((n + m)^3)$. Let us assume that there is an algorithm solving CHANNEL ASSIGNMENT in $2^{n o(\log \log \ell)}\text{poly}(r)$-time and then we can
solve our instance in time
\[
2^{O(\frac{n+m}{\log(n+m)}) \cdot \log((n+m)^3))} \cdot \text{poly}(O((n+m)^3)) \cdot \text{poly}(n+m) = 2^{O(n+m)}
\]
which contradicts ETH by Corollary 2.8. □

ACKNOWLEDGMENTS
We thank Łukasz Kowalik, the originator of this study, for his comprehensive support.

REFERENCES
Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. 2009. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing* 39, 2, 546–563.

Marek Cygan and Łukasz Kowalik. 2011. Channel assignment via fast zeta transform. *Information Processing Letters* 111, 15, 727–730.

Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. 2013. Known algorithms for EDGE CLIQUE COVER are probably optimal. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’13)*. 1044–1053. DOI: http://dx.doi.org/10.1137/1.9781611973105.75

Thore Husfeldt, Ramamohan Paturi, Gregory B. Sorkin, and Ryan Williams. 2013. Exponential algorithms: Algorithms and complexity beyond polynomial time (Dagstuhl seminar 13331). *Dagstuhl Reports* 3, 8, 40–72. DOI: http://dx.doi.org/10.4230/DagRep.3.8.40

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63, 4, 512–530.

Łukasz Kowalik and Arkadiusz Socała. 2014. Assigning channels via the meet-in-the-middle approach. In *Algorithm Theory—SWAT 2014*. Lecture Notes in Computer Science, Vol. 8503. Springer, 282–293.

Daniel Král. 2005. An exact algorithm for the channel assignment problem. *Discrete Applied Mathematics* 145, 2, 326–331.

Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2011a. Known algorithms on graphs on bounded treewidth are probably optimal. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’11)*. 777–789. DOI: http://dx.doi.org/10.1137/1.9781611973082.61

Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2011b. Slightly superexponential parameterized problems. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’11)*. 760–776. DOI: http://dx.doi.org/10.1137/1.9781611973082.60

Colin J. H. McDiarmid. 2003. On the span in channel assignment problems: Bounds, computing and counting. *Discrete Mathematics* 266, 1–3, 387–397.

Patrick Traxler. 2008. The time complexity of constraint satisfaction. In *Parameterized and Exact Computation*. Lecture Notes in Computer Science, Vol. 5018. Springer, 190–201.

Received February 2015; revised January 2016; accepted January 2016