Algorithmica (2016) 74:1435–1452
DOI 10.1007/s00453-015-0004-z

Assigning Channels Via the Meet-in-the-Middle Approach

Łukasz Kowalik[1] · Arkadiusz Socała[1]

Received: 10 August 2014 / Accepted: 23 April 2015 / Published online: 13 May 2015

Abstract We study the complexity of the Channel Assignment problem. By applying the meet-in-the-middle approach we get an algorithm for the ℓ-bounded Channel Assignment (when the edge weights are bounded by ℓ) running in time $O^*((2\sqrt{\ell+1})^n)$. This is the first algorithm which breaks the $(O(\ell))^n$ barrier. We extend this algorithm to the counting variant, at the cost of slightly higher polynomial factor. Very recently the second author showed that Channel Assignment does not admit a $O(c^n)$-time algorithm, for a constant c independent of ℓ. We consider a similar question for Generalized T-Coloring , a CSP problem that generalizes Channel Assignment. We show that Generalized T-Coloring does not admit a $2^{2^{o(\sqrt{n})}}\mathrm{poly}(r)$-time algorithm, where r is the size of the instance.

Keywords Channel Assignment · Meet-in-the-middle · Exponential time hypothesis · T-Coloring · Hardness · Exponential time

1 Introduction

In the Channel Assignment problem, we are given a symmetric weight function $w\colon V^2 \to \mathbb{N}$ (we assume that $0 \in \mathbb{N}$). The elements of V will be called vertices

Research supported by National Science Centre of Poland, Grant Number UMO-2013/09/B/ST6/03136.
An extended abstract was presented at 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2014) in Copenhagen, Denmark.

✉ Arkadiusz Socała
a.socala@mimuw.edu.pl; as277575@students.mimuw.edu.pl; arkadiusz.socala@mimuw.edu.pl

Łukasz Kowalik
kowalik@mimuw.edu.pl

1 Institute of Informatics, University of Warsaw, Warsaw, Poland

(as w induces a graph on the vertex set V with edges corresponding to positive values of w). We say that w is ℓ-bounded when for every $x, y \in V$ we have $w(x, y) \leq \ell$. An assignment $c\colon V \to \{1, \ldots, s\}$ is called *proper* when for each pair of vertices x, y we have $|c(x) - c(y)| \geq w(x, y)$. The number s 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 $E = \{uv\colon w(u, v) > 0\}$ with edge weights $w_E\colon E \to \mathbb{N}$ 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 called also the span of (G, w) and denoted by $\mathrm{span}(G, w)$.

It is interesting to realize the place of Channel Assignment in a kind of hierarchy of constraint satisfaction problems. We have already seen that it is a generalization of the classical graph coloring. It is also a special case of the constraint satisfaction problem (CSP). In CSP, we are given a vertex set V, a constraint set $\mathcal{C}$ 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 $c\colon V \to \{1, \ldots, d\}$ is *proper* if every constraint $A \in \mathcal{C}$ is satisfied, i.e. 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 Channel Assignment corresponds to CSP where $d = s$ and every edge uv of weight $w(uv)$ in the instance of Channel Assignment corresponds to the set of constraints of the form $\{(u, t_1), (v, t_2)\}$ where $|t_1 - t_2| < w(uv)$.

1.1 Previous Results for Channel Assignment and Related Problems

Since graph coloring (and hence also Channel Assignment) is an NP-complete problem, polynomial algorithms are unlikely to exist. Instead, we focus on algorithms running in time $O^*(c^n)$, for some constant $c > 1$. (The O^* notation supresses polynomial factors, i.e., $f(n) = O^*(g(n))$ when $f(n) = g(n)n^{O(1)}$.) The best algorithm known so far for Channel Assignment runs in $O^*(n!)$ time (see McDiarmid [14]). Since graph coloring is solvable in time $O^*(2^n)$ [2] it is natural to ask whether Channel Assignment is solvable in time $O^*(c^n)$, for some constant c (see [5,9,13]). This long-standing question has been recently answered in the negative by the second author [15], under the assumption of the Exponential Time Hypothesis (ETH). ETH was introduced by Impagliazzo and Paturi [10]. 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. ETH is a commonly used assumption for proving lower bounds for NP-hard problems, e.g., Traxler [16] showed under ETH that CSP does not have a $O^*(c^n)$-time algorithm for a constant c independent of d. The precise statement of the result of Socała [15] is that there is no algorithm solving Channel Assignment in $2^{o(n \log n)}$ time, unless ETH is false. Note that $O^*(n!) = 2^{O(n \log n)}$ so the lower bound is tight.

Fortunately, the running time of Channel Assignment can be improved in the ℓ-bounded variant. McDiarmid [14] came up with an $O^*((2\ell + 1)^n)$-time algorithm which has been next improved by Král [13] to $O^*((\ell + 2)^n)$ and to $O^*((\ell + 1)^n)$ by Cygan and Kowalik [5]. These are all dynamic programming (and hence exponential space) algorithms, and the last one applies the fast zeta transform to get a minor speed-up. Interestingly, all these works show also algorithms which *count* all proper assignments of span at most s within the same running time (up to polynomial factors) as the decision algorithm.

Even faster algorithms are known for some special cases of the problem. Perhaps the most studied one is the $L(2,1)$-labeling. An $L(2,1)$-labeling of a graph is a mapping from its vertex set into nonnegative integers such that the labels assigned to adjacent vertices differ by at least 2, and labels assigned to vertices of distance 2 are different. Note that the problem of finding an $L(2,1)$ of minimum span is a special case of 2-bounded Channel Assignment. Currently fastest algorithm for $L(2,1)$-labelling, due to Junosza-Szaniawski, Kratochvíl, Liedloff, Rossmanith and Rzążewski [11], runs in time $O(2.6488^n)$.

Let us now consider some more problems in the CSP hierarchy. In the T-Coloring , introduced by Hale [7], we are given a graph $G=(V,E)$, a set $T \subseteq \mathbb{N}$, and a number $s \in \mathbb{N}$. An assignment $c\colon V \to \{1,\ldots,s\}$ is proper when for every edge $uv \in E$ we have $|c(u)-c(v)| \notin T$. As usual, the goal is to determine whether there exists a proper assignment. Like Channel Assignment, T-Coloring is a special case of CSP and generalizes graph coloring. However, T-Coloring is incomparable with Channel Assignment.[1] However, Fiala, Král' and Škrekovski introduced Generalized List T-coloring which is a common generalization of vertex list-coloring (a variant of the classical graph coloring where each vertex has a list, i.e., a set of allowed colors), Channel Assignment and T-Coloring . An instance of the Generalized List T-coloring is a triple (G,Λ,t,s) where $G=(V,E)$ is a graph, $\Lambda\colon V \to 2^{\mathbb{N}}$, $t\colon E \to 2^{\mathbb{N}}$ and $s \in \mathbb{N}$. An assignment $c\colon V \to \{1,\ldots,s\}$ is proper when for every $v \in V$ we have $c(v) \in \Lambda(v)$, and for every edge $uv \in E$ we have $|c(u)-c(v)| \notin t(uv)$. As usual, the goal is to determine whether there exists a proper assignment. Similarly as in the case of Channel Assignment, we say that an instance of Generalized List T-coloring is ℓ-bounded if $\max \bigcup_{e\in E} t(e) \le \ell$. Very recently, the Generalized List T-coloring was considered by Junosza-Szaniawski and Rzążewski [12]. They show Generalized List T-coloring can be solved in $O^*((\ell+2)^n)$ time, which matches the time complexity of the algorithm of Cygan and Kowalik [5] for Channel Assignment (note that an ℓ-bounded instance of Channel Assignment can be seen as an $(\ell-1)$-bounded instance of Generalized List T-coloring).

1.2 Our Results

Our main result is a new $O^*((2\sqrt{\ell+1})^n)$-time algorithm for the ℓ-bounded Channel Assignment problem. Note that this is the first algorithm which breaks the $(O(\ell))^n$ barrier. Our algorithm follows the meet-in-the-middle approach and is surprisingly

[1] If in an instance (G,T,s) of T-Coloring the set T is not an interval $[0,d]$ for any $d \in \mathbb{N}$ then there is no way of defining a weight function $w : V(G)^2 \to \mathbb{N}$ such that (w,s) is an equivalent instance of Channel Assignment. Here, we mean that two instances are equivalent if they impose the same sets of constraints on every pair of vertices. Similarly, if the function w in an instance (w,s) of Channel Assignment has at least two different non-zero values, there is no set of edges E and set of forbidden colors T such that $(G=(V,E),T,s)$ is an equivalent instance of Channel Assignment.

simple, so we hope it can become a yet another clean illustration of this beautiful technique. At a high level, the meet-in-the-middle approach is to solve the problem for (sometimes carefully defined) two *halves* of the input instance, and then merge the results to get a solution for the original instance. Perhaps the most famous application of this idea to an NP-hard problem is the $O^*(2^{n/2})$ algortihm for Subset Sum by Horowitz and Sahni [8]. For more recent applications, see e.g. [1,4].

We show also a (more technical) counting version of our decision algorithm for Channel Assignment, which runs within the same time (up to a polynomial factor).

By the mentioned result of Socała [15], no algorithm solves Generalized List T-coloring in $2^{o(n \log n)}$ time, unless ETH is false. The second result of this paper is a much stronger lower bound for Generalized List T-coloring . We even consider a restricted case, i.e. the non-list version where every vertex is allowed to have any color, so the instance is just a triple (G, t, s). We call it Generalized T-Coloring . We show that, under ETH, Generalized T-Coloring does not admit a $2^{2^{o(\sqrt{n})}}\mathrm{poly}(r)$-time algorithm, where r is the size of the instance (including all the bits needed to represent the sets $t(e)$ for all $e \in E$). Note that this rules out an $O(n!)$ algorithm as well.

1.3 Organization of the Paper

In Sect. 2 we describe an $O^*((\ell + 2)^n)$-time dynamic programming algorithm for ℓ-bounded Channel Assignment. It is then used as a subroutine in the $O^*((2\sqrt{\ell+1})^n)$-time algorithm described in Sect. 3. In Sect. 4 we extend the algorithm from Sect. 3 to counting proper assignments of given span. In Sect. 5 we discuss hardness of Generalized T-Coloring under ETH. Finally, in Sect. 6 we briefly mention some directions for further work.

1.4 Notation

Throughout the paper n denotes the number of the vertices of the graph under consideration. For an integer k, by $[k]$ we denote the set $\{1, 2, \ldots, k\}$. We also use the Iverson's bracket notation, i.e., for a logical condition α, the expression $[\alpha]$ has value 1 if α holds and 0 otherwise. For a set S and an integer k by $\binom{S}{k}$ we denote the family of all subsets of S of size k. Similarly, $\binom{S}{\le k}$ denotes the family of all subsets of S of size at mist k. Let $G = (V, E)$ be a graph. For a subset $X \subseteq V$ consider the subgraph with vertex set X and edge set $\{uv \in E \colon u, v \in X\}$. This subgraph is called the subgraph of G *induced* by X and denoted by $G[X]$. Finally, $\uplus$ is the disjoint sum of sets i.e. the standard sum of sets $\cup$ but with an additional assumption that the sets are disjoint.

2 Yet Another $O^*((\ell + 2)^n)$-Time Dynamic Programming

In this section we provide a $O^*((\ell + 2)^n)$-time dynamic programming algorithm for Channel Assignment. It uses a different approach than e.g. the algorithm of Kral, and will be used as a subroutine in our faster algorithm.

For a subset $X \subseteq V$ and a function $f\colon X \to [\ell+1]$ let $\mathcal{A}_{X,f}$ be the set of all proper assignments $c\colon X \to \mathbb{N}$ of the graph $G[X]$ subject to the condition that for every $x \in X$ we have $c(x) \geq f(x)$.

For every subset $X \subseteq V$ and $f\colon X \to [\ell+1]$ we compute the value of $T[X, f]$ which is equal to the minimum span of an assignment from $\mathcal{A}_{X,f}$. Clearly, the minimum span of (G, w) equals to $T[V, f_1]$ where f_1 is the constant function which assigns 1 to every vertex.

The values of $T[X, f]$ are computed by dynamic programming as follows. First we initialize $T[\emptyset, \emptyset] = 0$ (note that the only function $f\colon \emptyset \to [\ell+1]$ is technically the empty set). Next, we iterate over all non-empty subsets of V in the order of nondecreasing cardinality. In order to determine the value of $T[X, f]$ we use the recurrence relation formulated in the following lemma.

Informally, it uses the observation that there is a minimum-span assignment c such that the vertex $v \in X$ with minimum color $c(v)$ is *left-shifted*, i.e. $c(v) = f(v)$. Hence we can check all possibilities for v and then the colors of all the other vertices from X have lower bounds in range $\{f(v), \dots, f(v)+\ell\}$, so we can translate the range back down to $\{1, \dots, \ell+1\}$ and use the previously computed values of $T[X \setminus \{v\}, \cdot]$.

Lemma 1 *For a subset $X \subseteq V$, a function $f\colon X \to [\ell+1]$ and a vertex $v \in X$ define the function $f_v\colon X \setminus \{v\} \to [\ell+1]$ given by the formula*

$$f_v(x) = 1 + \max\{w(v,x), f(x) - f(v)\} \quad \textit{for every } x \in X \setminus \{v\}.$$

Then,

$$T[X, f] = \min_{v \in X}(f(v) + T[X \setminus \{v\}, f_v] - 1), \tag{1}$$

Proof Fix $v \in X$. Denote $\mathcal{A}_{X,f,v} = \{c \in \mathcal{A}_{X,f}\colon c(v) = f(v) = \min_{x \in X} f(x)\}$. Then, for every assignment $c \in \mathcal{A}_{X,f,v}$, for every $x \in X \setminus \{v\}$ we have $c(x) \geq f(v) + \max\{w(v,x), f(x) - f(v)\}$. Hence, the minimum span of an assignment from $\mathcal{A}_{X,f,v}$ is equal to $f(v) + T[X \setminus \{v\}, f_v] - 1$. It suffices to show that there is an assignment $c^* \in \mathcal{A}_{X,f}$ of minimum span such that $c^*(v) \in A_{X,f,v}$ for some $v \in X$. Consider an arbitrary assignment $c^* \in \mathcal{A}_{X,f}$ of minimum span. Let $x \in X$ be the vertex of minimum color, i.e. $c^*(x)$ is minimum. If $c^*(x) = f(x)$ we are done. Otherwise consider a new assignment c^{**} which is the same as c^* everywhere except for x and $c^{**}(x) = f(x)$; then c^{**} is proper since $c^*(x)$ is minimal and clearly $c^{**} \in \mathcal{A}_{X,f}$. The span of c^{**} is not greater than the span of c^* (actually they are the same since c^* has minimal span), so the claim follows. □

The size of the array T is $\sum_{i=0}^{n} \binom{n}{i}(\ell+1)^i = (\ell+2)^n$. Computing a single value based on previously computed values for smaller sets takes $O(n^2)$ time, hence the total computation takes $O((\ell+2)^n n^2)$ time. As described, it gives the minimum span only, but we can retrieve the corresponding assignment within the same running time using standard techniques.

3 The Meet-in-the-Middle Speed-Up

In this section we present our main result, an algorithm for ℓ-bounded CHANNEL ASSIGNMENT that applies the meet-in-the-middle technique. Roughly, the idea is to find partial solutions for all possible *halves* of the vertex set and then merge the partial solutions efficiently to solve the full instance.

For the clarity of the presentation we assume n is even (otherwise we just add a dummy isolated vertex). Before we describe the algorithm let us introduce some notation. For a set $X \subseteq V$, by $\overline{X}$ we denote $V \setminus X$. Moreover, for a function $f\colon X \to [\ell+1]$ we define function $\overline{f}\colon \overline{X} \to [\ell+1]$ such that for every $v \in \overline{X}$,

$$\overline{f}(v) = 1 + \max(\{1 + w(uv) - f(u)\colon uv \in E,\ u \in X\} \cup \{0\}).$$

The values $T[X, f]$ are defined as in Sect. 2. Our algorithm is based on the following observation.

Lemma 2 *The span of* (G, w) *is equal to*

$$\min(T[X, f] + T[\overline{X}, \overline{f}] - 1),$$

where the minimum is over all pairs (X, f) *where* $X \in \binom{V}{n/2}$ *and* $f\colon X \to [\ell+1]$.

Proof Let $c^*\colon V \to \mathbb{N}$ be a proper assignment of minimum span s. Order the vertices of $V = \{v_1, \ldots, v_n\}$ so that for every $i = 1, \ldots, n-1$ we have $c^*(v_i) \le c^*(v_{i+1})$. Consider the subset $X = \{v_1, \ldots, v_{n/2}\}$. Let $s_1 = c^*(v_{n/2})$. Define $f\colon X \to [\ell+1]$ such that $f(x) = 1 + \min\{s_1 - c^*(x), \ell\}$ for every $x \in X$. From the definition of T we have $T[X, f] \le s_1$ (because the assignment $x \mapsto 1 + s_1 - c^*(x)$ belongs to $\mathcal{A}_{X,f}$ and has span s_1). Moreover, note that for every $v \in \overline{X}$ it holds that

$$\begin{aligned} c^*(v) &\ge \max(\{c^*(u) + w(uv)\colon uv \in E,\ u \in X\} \cup \{s_1\}) \\ &= \max(\{s_1 + w(uv) - f(u) + 1\colon uv \in E,\ u \in X\} \cup \{s_1\}) \\ &= s_1 - 1 + \overline{f}(v). \end{aligned}$$

It follows that $s = \max_{v \in \overline{X}} c^*(v) \ge s_1 - 1 + T[\overline{X}, \overline{f}] \ge T[X, f] + T[\overline{X}, \overline{f}] - 1$.

Finally we show that $s > T[X, f] + T[\overline{X}, \overline{f}] - 1$ contradicts the optimality of c^*. Let $c_1 \in \mathcal{A}_{X,f}$ be an assignment of span $T[X, f]$ and let $c_2 \in \mathcal{A}_{\overline{X},\overline{f}}$ be an assignment of span $T[\overline{X}, \overline{f}]$. Consider the following assignment $c\colon V \to \mathbb{N}$.

$$c(x) = \begin{cases} 1 + T[X, f] - c_1(x) & \text{for } x \in X \\ T[X, f] + c_2(x) - 1 & \text{for } x \in \overline{X} \end{cases}$$

One can check that from the definition of $\overline{f}$ it follows that c is a proper assignment. Moreover, the span of c is equal to $T[X, f] + T[\overline{X}, \overline{f}] - 1$. Hence, if $s > T[X, f] + T[\overline{X}, \overline{f}] - 1$ then c^* is not optimal, a contradiction. □

From Lemma 2 we immediately obtain the following algorithm for computing the span of (G, w):

1. Compute the values of $T[X, f]$ for all $X \in \binom{V}{\le n/2}$ and $f\colon X \to [\ell+1]$ using the algorithm from Sect. 2.
2. Find the span of (G, w) using the formula from Lemma 2.

Note that Step 1 takes time proportional to $\sum_{i=0}^{n/2}\binom{n}{i}(\ell+1)^i n^2 = O(2^n(\ell+1)^{n/2}n^2)$. The size of array T is clearly $O(2^n(\ell+1)^{n/2})$. In Step 2 we compute a minimum of $\binom{n}{n/2}(\ell+1)^{n/2} = O(2^n(\ell+1)^{n/2})$ values. Hence the total time is $O(2^n(\ell+1)^{n/2}n^2)$. As described, the above algorithm gives the minimum span only, but we can retrieve the corresponding assignment within the same running time using standard techniques. We have just proved the following theorem.

Theorem 1 *For every ℓ-bounded weight function the channel assignment problem can be solved in $O(2^n(\ell+1)^{n/2}n^2)$ time.*

4 An Extension to Counting

In this section we present an extension of our meet-in-the-middle algorithm which finds the number of proper assignments of span s. This is slightly more technical than the decision algorithm because we need to avoid counting the same assignment more than once. We assume here that $V = \{1, \ldots, n\}$ (we will use the fact that V is linearly ordered).

For $X \in \binom{V}{n/2}$, function $f\colon X \to [\ell+1]$ and value $r = 1, \ldots, s$ denote the set of all assignments from $\mathcal{A}_{X,f}$ with span r by $\mathcal{A}_{X,f,r}$. Let us denote $Q[X, f, r] = |\mathcal{A}_{X,f,r}|$. For a subset $X \in \binom{V}{\le n/2}$, a function $f\colon X \to [\ell+1]$ and a vertex v define the function $f_v\colon X \setminus \{v\} \to [\ell+1]$ given by the formula

$$f_v(x) = \max\{f(x), 1 + w(vx), 1 + [x < v]\} \qquad \text{for every } x \in X \setminus \{v\}.$$

Also, for a function $f\colon X \to \{2, \ldots, \ell+1\}$ define the function $f_\downarrow\colon X \to [\ell+1]$ given by the formula

$$f_\downarrow(x) = \max\{f(x) - 1, 1\} \qquad \text{for every } x \in X.$$

We will use the recurrence relation formulated in the following lemma.

Lemma 3 *For every $X \in \binom{V}{n/2}$, $f\colon X \to [\ell+1]$ and $r = 1, \ldots, s$*

$$Q[X, f, r] = \begin{cases} \sum_{v \in f^{-1}(1)} Q[X \setminus \{v\}, f_v, r] + [r > 1]Q[X, f_\downarrow, r-1] & \textit{if } X \neq \emptyset \\ [r = 1] & \textit{otherwise} \end{cases} \tag{2}$$

Proof The proof is by induction on $|X|+r$. The formula (2) clearly holds when $X = \emptyset$, since there is exactly one assignment with empty domain, it is proper and its span is 1.

Assume $X \neq \emptyset$. The set $\mathcal{A}_{X,f,r}$ partitions into two subsets $\mathcal{B}$ and $\mathcal{C}$, where $\mathcal{B}$ contains the assignments which assign color 1 to some vertex and $\mathcal{C}$ contains the remaining assignments.

We can further partition $\mathcal{B} = \bigcup_{v \in f^{-1}(1)} \mathcal{B}_v$, where

$$\mathcal{B}_v = \{c \in \mathcal{B}\colon \min c^{-1}(1) = v\}.$$

Define $\mathcal{B}'_v = \{c|_{X\setminus\{v\}}\colon c \in \mathcal{B}_v\}$. Then $|\mathcal{B}'_v| = |\mathcal{B}_v|$. Consider an arbitrary $c \in \mathcal{B}_v$. Then for every $x \in X\setminus\{v\}$ we have $c(x) \geq f(x)$, $c(x) \geq f(v)+w(vx) = 1+w(vx)$, and if $x < v$ then $c(x) \geq 2$. In other words, for every $x \in X \setminus \{v\}$ we have $c(x) \geq f_v(x)$ and hence $c|_{X\setminus\{v\}} \in \mathcal{A}_{X\setminus\{v\},f_v,r}$. It follows that $\mathcal{B}'_v \subseteq \mathcal{A}_{X\setminus\{v\},f_v,r}$. It is also easy to verify that every assignment $c' \in \mathcal{A}_{X\setminus\{v\},f_v,r}$ can be extended to a proper assignment $c \in \mathcal{B}_v$ by putting $c(v) = 1$ and $c|_{X\setminus\{v\}} = c'$. Hence $\mathcal{A}_{X\setminus\{v\},f_v,r} \subseteq \mathcal{B}'_v$. It follows that $\mathcal{B}'_v = \mathcal{A}_{X\setminus\{v\},f_v,r}$ and hence $|\mathcal{B}_v| = |\mathcal{A}_{X\setminus\{v\},f_v,r}| = Q[X\setminus\{v\}, f_v, r]$, where the last equality follows from the induction hypothesis. We get $|\mathcal{B}| = \sum_{v\in f^{-1}(1)} Q[X \setminus \{v\}, f_v, r]$.

If $r = 1$ then $\mathcal{C} = \emptyset$. Assume $r > 1$. It is clear that the assignments in $\mathcal{C}$ are in 1–1 correspondence with the assignments in $\mathcal{C}' = \{c_{\downarrow}\colon c \in \mathcal{C}\}$ and the assignments in $\mathcal{C}'$ have span $r-1$. Hence $|\mathcal{C}| = |\mathcal{A}_{X,f_{\downarrow},r-1}| = Q[X, f_{\downarrow}, r-1]$, where the last equality follows from the induction hypothesis.

To sum up,

$$|\mathcal{A}_{X,f,r}| = |\mathcal{B}| + |\mathcal{C}| = \sum_{v\in f^{-1}(1)} Q[X \setminus \{v\}, f_v, r] + [r > 1] \cdot Q[X, f_{\downarrow}, r-1],$$

as required. □

With Lemma 3 it is easy to describe a dynamic programming algorithm which for every subset $X \in \binom{V}{n/2}$, function $f\colon X \to [\ell+1]$ and value $r = 1, \ldots, s$ computes the value of $Q[X, f, r]$. First we initialize $Q[\emptyset, \emptyset, r] = [r = 1]$ for every $r = 1, \ldots, s$ and next the values of $Q[X, f, r]$ are computed according to Formula (2), using previously computed values of array Q; to this end we iterate over the triples (X, f, r) in nondecreasing order of $|X|+r$ (note that by Eq. 2 the value of $Q[X, f, r]$ depends only from the entries of array Q with smaller value of $|X|+r$). The number of triples considered is $O(2^n(\ell+1)^{n/2}s)$ and processing each triple takes $O(n^2)$ time. We have just shown the following.

Lemma 4 *There is an $O(2^n(\ell+1)^{n/2}sn^2)$-time $O(2^n(\ell+1)^{n/2}s)$-space algorithm which finds the values of $Q[X, f, r]$ for all subsets $X \in \binom{V}{n/2}$, functions $f\colon X \to [\ell+1]$ and values $r = 1, \ldots, s$.*

If we use just the values of $Q[X, f, r]$ in the merge phase of the meet-in-the-middle approach, it is unclear how to avoid double-counting the same assignments. To overcome this problem, for a subset $X \subseteq V$, a function $f\colon X \to [\ell+1]$ and a value $r = 1, \ldots, s$ define $\mathcal{A}^*{}_{X,f,r}$ as the set of all proper assignments $c\colon X \to \mathbb{N}$ of the graph $G[X]$ such that c has span r and for every $x \in X$, if $f(x) \leq \ell$ then $c(x) = f(x)$ and otherwise $c(x) \geq f(x)$. Denote $Q^*[X, f, r] = |\mathcal{A}^*{}_{X,f,r}|$.

For a subset $X \subseteq V$ and a function $f\colon X \to [\ell+1]$ define the function $f_{\leftarrow \ell}\colon X \setminus f^{-1}([\ell]) \to [\ell+1]$ given by the formula

$$f_{\leftarrow \ell}(x) = \max(\{f(y) + w(yx) - \ell\colon y \in f^{-1}([\ell])\} \cup \{1\}),$$

for every $x \in X \setminus f^{-1}([\ell])$. Observe the following.

Observation 2 *For every $X \subseteq V$, $f\colon X \to [\ell+1]$ and $r = 1, \ldots, s$*

(i) if $f|_{f^{-1}([\ell])}$ is not a proper assignment then $Q^*[X, f, r] = 0$;
(ii) if $f|_{f^{-1}([\ell])}$ is a proper assignment and $r \leq \ell$ then

$$Q^*[X, f, r] = [f^{-1}(\{r\}) \neq \emptyset \text{ and } f^{-1}(\{r+1, \ldots, \ell+1\}) = \emptyset];$$

(iii) if $f|_{f^{-1}([\ell])}$ is a proper assignment and $r \geq \ell + 1$ then

$$Q^*[X, f, r] = Q[X \setminus f^{-1}([\ell]), f_{\leftarrow \ell}, r - \ell]. \tag{3}$$

Proof Claim (i) follows from the definition of $\mathcal{A}^*{}_{X,f,r}$. The condition in (ii) makes all the values of the assignment fixed, so there is exactly one assignment, unless $f^{-1}(\{r\}) = \emptyset$ (the assignment has span smaller than r) or $f^{-1}(\{r+1, \ldots, \ell+1\}) \neq \emptyset$ (the assignment has span bigger than r). Once (i) and (ii) are excluded, there is a natural one-to-one correspondence between the elements of $\mathcal{A}^*{}_{X,f,r}$ and $\mathcal{A}^*{}_{X \setminus f^{-1}([\ell]), f_{\leftarrow \ell}, r-\ell}$. Hence (iii) follows. □

Now we proceed to the merge phase of our meet-in-the-middle algorithm. For a function $f\colon X \to [\ell+1]$ we define function $\tilde{f}\colon \overline{X} \to [\ell+1]$ such that for every $v \in \overline{X}$,

$$\tilde{f}(v) = 1 + \max(\{1 + w(uv) - f(u)\colon uv \in E,\ u \in X\} \cup \{[v < \max f^{-1}(1)]\}).$$

The role of the function $\tilde{f}$ is similar as $\overline{f}$ in determining the span using the meet-in-the-middle approach; the only difference is that if for some $x \in X$ we have $f(x) = 1$ then for every $v \in \overline{X}$, if $v < x$ then $\tilde{f}(v) \geq 2$. Informally, this helps us to avoid counting the same assignment once for every partition of the "middle color" into parts of relevant sizes. Now we can formulate the counting counterpart of Lemma 2.

Lemma 5 *For a given graph G, weight function w and integer $s \in \mathbb{N}$ the number of proper assignments of span s is equal to*

$$\sum_{s^*=1}^{s} \sum_{X \in \binom{V}{n/2}} \sum_{\substack{f: X \to [\ell+1] \\ f^{-1}(1) \neq \emptyset}} Q^*[X, f, s^*] \cdot Q[\overline{X}, \tilde{f}, s - s^* + 1].$$

Proof Let $\mathcal{D}$ be the set of all proper assignments of span s. For an assignment $c \in \mathcal{D}$ define a total order of V as follows: for $i, j \in V$ we have $i \prec j$ iff $(c(i), i) \leq_{\text{lex}}$

$(c(j), j)$, where $\leq_{\text{lex}}$ is the lexicographic order. Then c defines a permutation of the vertices $v_1^c \prec v_2^c \prec \cdots \prec v_n^c$. Then $\mathcal{D} = \biguplus_{s^*=1}^{s} \mathcal{D}_{s^*}$, where

$$\mathcal{D}_{s^*} = \{c \in \mathcal{D}\colon c(v_{n/2}^c) = s^*\}$$

Moreover, $\mathcal{D}_{s^*} = \biguplus_{X \in \binom{V}{n/2}} \mathcal{D}_{s^*,X}$, where

$$\mathcal{D}_{s^*,X} = \{c \in \mathcal{D}_{s^*}\colon \{v_1^c, \ldots, v_{n/2}^c\} = X\}.$$

Finally,

$$\mathcal{D}_{s^*,X} = \biguplus_{\substack{f\colon X \to [\ell+1] \\ f^{-1}(1) \neq \emptyset}} \mathcal{D}_{s^*,X,f},$$

where $\mathcal{D}_{s^*,X,f}$ is the set of assignments $c \in \mathcal{D}_{s^*,X}$ such that for every $x \in X$, if $f(x) \leq \ell$ then $c(x) = s^* - f(x) + 1$ and if $f(x) = \ell + 1$ then $c(x) \leq s^* - f(x) + 1$. Note that the condition $f^{-1}(1) \neq \emptyset$ is necessary to satisfy the defining condition of $\mathcal{D}_{s^*}$; in particular $v_{n/2}^c = \max f^{-1}(1)$.

Consider an arbitrary $c \in \mathcal{D}_{s^*,X,f}$. Now observe that for every $v \in \overline{X}$ and $u \in X$ such that $uv \in E$, we have $c(v) \geq \max\{c(u) + w(uv), s^*\}$. Moreover, if $v < v_{n/2}^c$, i.e. $v < \max f^{-1}(1)$, then $c(v) \geq s^* + 1$. Hence,

$$\begin{aligned} c(v) &\geq \max(\{c^*(u) + w(uv)\colon uv \in E,\ u \in X\} \cup \{s^* + [v < \max f^{-1}(1)]\}) \\ &= \max(\{s^* + w(uv) - f(u) + 1\colon uv \in E,\ u \in X\} \cup \{s^* + [v < \max f^{-1}(1)]\}) \\ &= s^* - 1 + \tilde{f}(v). \end{aligned}$$

It follows that $|\mathcal{D}_{s^*,X,f}| = Q^*[X, f, s^*] \cdot Q[\overline{X}, \tilde{f}, s - s^* + 1]$, as required. □

From Lemma 4, Observation 2 and Lemma 5 we infer the following theorem.

Theorem 3 *For every ℓ-bounded weight function the number of all proper assignments of a given span can be computed in $O^*(2^n(\ell+1)^{n/2})$ time.*

5 Hardness of Generalized T-Coloring

In this section we give a lower bounds for the time complexity of Generalized T-Coloring , under ETH. To this end we present a reduction from Set Cover. The instance of the decision version of Set Cover consists of a family of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ and a number k. The set $U = \bigcup \mathcal{S}$ is called *the universe* and we denote $n = |U|$. The goal is to decide whether there is a subfamily $\mathcal{C} \subseteq \mathcal{S}$ of size at most k such that $\bigcup \mathcal{C} = U$ (then we say the instance is *positive*).

In the following lemma we reduce Set Cover to the decision version of Generalized T-Coloring , where for a given instance (G, w, s) we ask whether there is a proper assignment of span at most s (then we say the instance is *positive*). We say that an instance $(\mathcal{S}, k)$ of SET COVER is equivalent to an instance (G, w, s) of GENERALIZED T-COLORING when $(\mathcal{S}, k)$ is positive iff (G, w, s) is positive. For every edge e of G, every pair (e, d) for $d \in t(e)$ is called a *constraint*.

Lemma 6 *Let* $(\mathcal{S}, k)$ *be an instance of* SET COVER *with m sets and universe of size n and let $A \in [1, m]$ and $B \in [1, n]$ be two reals. Then we can generate in polynomial time an equivalent instance of* GENERALIZED T-COLORING *which has* $O\left(\frac{n}{B} + \frac{m}{A} \cdot \max\{1, \log A\}\right)$ *vertices,* $O^*\left(2^A \cdot m^B\right)$ *constraints and is* $O\left(2^A \cdot m^B\right)$*-bounded.*

Proof For convenience we assume that A and B are natural numbers, since otherwise we round A and B down and the whole construction and its analysis is the same, up to some details.

In the proof we consider coloring of the vertices as placing the vertices on a number line in such a way that every vertex is placed in the coordinate equal to its color.

Let $\mathcal{S} = \{S_1, \ldots, S_m\}$. We are going to construct a complex instance $(G = (V, E), t, s)$ of GENERALIZED T-COLORING . We describe it step-by-step and show some of its properties.

We begin by putting vertices v_L and v_R in V and $t(v_L v_R) = \{0, \ldots, s-2\}$, i.e. in every proper assignment v_L has color 1 and v_R has color s, or the other way around; w.l.o.g. we assume the first possibility. We specify s later.

In what follows, whenever we put a new vertex v in V, we will specify the set $A(v)$ of its *allowed* colors. Formally, this corresponds to putting $t(v_L v) = \{d \in \{0, \ldots, s-1\} : d + 1 \notin A(v)\}$.

Our instance will consist of three separate modules (the set choice module, the witness module and the parsimonious module). By separate we mean they have disjoint sets of vertices V_S, V_U and V_P and moreover they have disjoint sets of allowed colors, i.e. for $i, j \in \{S, U, P\}$, when $x \in V_i$ and $y \in V_j$ for $i \neq j$ then $A(x) \cap A(y) = \emptyset$. However the modules will interfere with each other by forbidding some distances between pairs of vertices from two different modules.

The set choice module The first module represents the sets in $\mathcal{S}$. For every $i = 1, \ldots, \left\lceil \frac{m}{A} \right\rceil$ the set V_S contains a vertex s_i. Vertex s_i represents the A sets

$$\mathcal{S}_i = \{S_{(i-1)\cdot A+1}, S_{(i-2)\cdot A+2}, \ldots, S_{i\cdot A}\}$$

(and the last vertex $s_{\lceil m/A \rceil}$ represents $\mathcal{S}_{\lceil m/A \rceil} = S_{(\lceil m/A \rceil - 1)A+1}, \ldots, S_m$). We also put $A(s_i) = \{1, \ldots, 2^A\}$ for every $s_i \in V_S$. The intuition is that the color $c \in [2^A]$ of a vertex s_i corresponds to a subset $\mathcal{S}_i(c) \subseteq \mathcal{S}_i$, i.e. the choice of sets from $\mathcal{S}_i$ to the solution of SET COVER (Fig. 1).

The witness module Let denote the elements of the universe as $e_1, e_2, \ldots, e_n$. For every $i = 1, \ldots, \left\lceil \frac{n}{B} \right\rceil$ the set V_U contains a vertex u_i. Vertex u_i represents the B elements

$$U_i = \{e_{(i-1)\cdot B+1}, e_{(i-2)\cdot B+2}, \ldots, e_{i\cdot B}\}$$

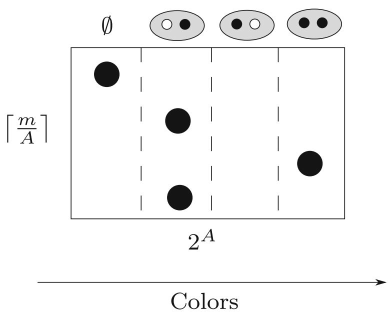

Fig. 1 The set choice module

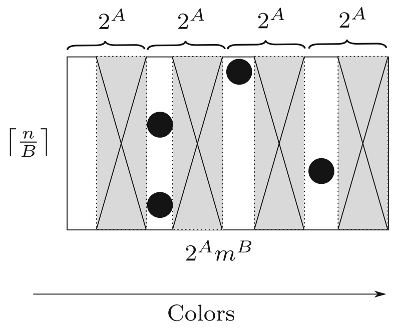

Fig. 2 The witness module (the *grey* areas are the gaps between the m^B potentially allowed positions)

(and the last vertex $u_{\lceil n/B \rceil}$ represents $U_{\lceil n/B \rceil} = e_{(\lceil n/B \rceil - 1)B+1}, \ldots, e_n$).

This time vertices V_U do not need to have the same sets of allowed colors, but for every $u \in V_U$ we have $A(u) \subseteq \{1 + i \cdot 2^A \colon i = 1, \ldots, m^B\}$. Note that every vertex has at most m^B allowed colors and there are gaps of length $2^A - 1$ where no vertex is going to be assigned.

We say that a sequence $(S_{w_1}, \ldots, S_{w_B}) \in \mathcal{S}^B$ is a *witness* for a vertex $u_i \in V_U$ when

$$U_i \subseteq \bigcup_{j=1}^{B} S_{w_j}.$$

For every $i = 1, \ldots, m^B$ color $1 + i \cdot 2^A$ corresponds to the i-th sequence in the set $\mathcal{S}^B$ (say, in the lexicographic order of indices); we denote this sequence by $\mathcal{W}_i$. Then, for every $u \in V_U$,

$$A(u) = \{1 + i \cdot 2^A \colon \mathcal{W}_i \text{ is a witness for u}, i = 1, \ldots, m^B\}.$$

The intuition should be clear: color of a vertex $u_i \in V_U$ in a proper assignment represents the choice of at most B sets in the solution of Set Cover which cover U_i (Fig. 2).

The interaction between the set choice module and the witness module As we have seen, every assignment c of colors to the vertices determines a choice of a subfamily

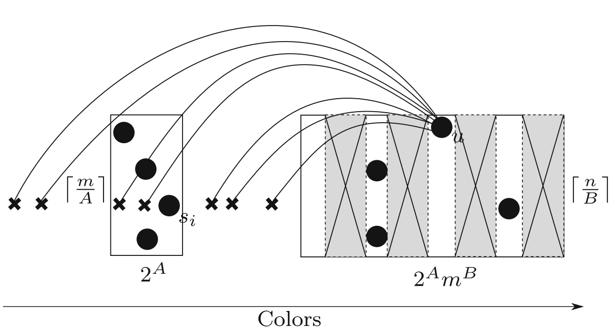

Fig. 3 The interaction between a vertex s_i in the set choice module and a vertex u in the witness module. All the drawn arcs are forbidden distances between s_i and u. Note that for every possible color $1 + j \cdot 2^A$ of u the subset of $[2^A]$ excluded by the forbidden distances in $t(us_i)$ is exactly $F_{i,j}$

$\mathcal{S}(c) \subseteq \mathcal{S}$, where $\mathcal{S}(c) = \bigcup_{i=1}^{\lceil m/A \rceil} \mathcal{S}_i(c(i))$. Similarly, c determines a choice of a subfamily $\mathcal{S}'(c) \subseteq \mathcal{S}$, where $\mathcal{S}'(c) = \bigcup_{u \in V_U} \mathcal{W}_{c(u)}$. It should be clear that we want to force that in every proper assignment $\mathcal{S}'(c) \subseteq \mathcal{S}(c)$. To this end we introduce edges between the two modules.

For $i = 1, \dots, \lceil \frac{m}{A} \rceil$ and $j = 1, \dots, m^B$ define the following set of forbidden colors

$$F_{i,j} = \{c \in [2^A] \colon \mathcal{W}_j \cap \mathcal{S}_i \nsubseteq \mathcal{S}_i(c)\}.$$

The intuition is the following: If a proper assignment colors a vertex $u_i \in V_U$ with color $1 + j \cdot 2^A$ (i.e. it assigns the witness $\mathcal{W}_j$ to the set U_i) then it cannot color the vertex s_i with colors from $F_{i,j}$ (i.e. choose this subsets of $\mathcal{S}_i$ corresponding to these colors), for otherwise $\mathcal{S}'(c) \nsubseteq \mathcal{S}(c)$ (Fig. 3).

Claim 1 Consider any proper assignment $c\colon V \to [s]$. If for every $i = 1, \dots, \lceil \frac{m}{A} \rceil$ we have $c(s_i) \notin \bigcup_{u \in V_U} F_{i,c(u)}$, then $\mathcal{S}'(c) \subseteq \mathcal{S}(c)$.

Proof of the claim: Consider a set $S_t \in \mathcal{W}_{c(u)}$ for an arbitrary $u \in V_U$. Then $S_t \in \mathcal{S}_i$ for some i. From the assumption, $c(s_i) \notin F_{i,c(u)}$, so $\mathcal{W}_{c(u)} \cap \mathcal{S}_i \subseteq \mathcal{S}_i(c)$. Hence, $S_t \in \mathcal{S}(c)$, as required.

Hence we would like to add some forbidden distances to our instance to make the assumption of Clam 1 hold. To this end, for every $u \in V_U$ and every $s_i \in V_S$ we put

$$t(us_i) = \bigcup_{j=1}^{m^B} \{1 + j \cdot 2^A - f \colon f \in F_{i,j}\}.$$

In other words, for every possible color $1 + j \cdot 2^A$ of u we forbid all distances between u and s_i that would result in coloring s_i with $F_{i,j}$. Then indeed the assumption from Claim 1 holds. ◻

Claim 2 For any proper assignment $c\colon V \to [s]$ we have $\mathcal{S}'(c) \subseteq \mathcal{S}(c)$.

Proof of the claim: We need to verify the assumption in Claim 1. Assume for the contradiction that for some i and some $u \in V_U$ we have $c(s_i) \in F_{i,c(u)}$. Recall that in a proper assignment $c(u) = 1 + j \cdot 2^A$ for some $j = 1, \ldots, m^B$. Then $|c(u) - c(s_i)| = 1 + j \cdot 2^A - c(s_i) \in t(us_i)$, a contradiction. □

Claim 3 For any proper assignment c: $V \to [s]$ we have $\mathcal{S}(c)$ covers the universe.

Proof of the claim: This is an immediate corollary from Claim 2 and the fact that every vertex $u \in V_U$ is colored with a color from $A(u)$. □

Claim 4 For every cover $\mathcal{C} \subseteq \mathcal{S}$ of the universe, there is a proper assignment c: $V \to [s]$ such that $\mathcal{S}(c) = \mathcal{C}$.

Proof of the claim: We color v_L and v_R with 1 and s, and every vertex s_i with the color from $[2^A]$ corresponding to the subset $\mathcal{S}_i \cap \mathcal{C}$ of $\mathcal{S}_i$. For every set U_i for every $e \in U_i$ we pick a set $S_e \in \mathcal{C}$ that contains e and we build a witness $\mathcal{W}$ from the sets S_e. We color u_i with the color $1 + j \cdot 2^A$, where j is the number of $\mathcal{W}$ in the lexicographic order of all witnesses. It remains to check that the resulting assignment c is proper. The only nontrivial issue is whether for every $u \in V_U$ and $s_i \in V_S$ we have $|c(u) - c(s_i)| \notin t(us_i)$. It is clear that $|c(u) - c(s_i)| \notin \{c(u) - f \colon f \in F_{i,j}\}$, where j is such that $c(u) = 1 + j \cdot 2^A$. However, for every $j' \neq j$ the set $\{1 + j' \cdot 2^A - f \colon f \in F_{i,j'}\}$ is disjoint from 2^A (this is where we make use of the 'gaps' of length $2^A - 1$). □

Bounding the number of sets chosen to the solution The last thing we need in a proper assignment c is to keep the number of the sets in $\mathcal{S}(c)$ bounded by k. To this end we use the parsimonious module with the vertex set V_P.

The third parsimonious module consists of $\lceil \frac{m}{A} \rceil$ consecutive submodules and an additional free space of length $2k$ (meaning that for every $v \in V_P$ the set of allowed colors $A(v)$ contains this free space). Between those submodules and the additional free space we put a gap of length 2^A, where no vertex can be assigned. The intuition is that in a proper assignment c the i-th submodule represents the number of sets from $\mathcal{S}_i$ chosen to the solution, i.e. $|\mathcal{S}_i(c_i)|$.

More precisely, $V_P = \biguplus_{i=1}^{\lceil m/A \rceil} V_i$, where V_i is a set of $1 + \lfloor \log A \rfloor$ vertices representing numbers $2^0, 2^1, \ldots, 2^{\lfloor \log A \rfloor}$. For a vertex $x \in V_P$ let $r(x)$ denote the number represented by x. For every two vertices $x, y \in V_P$ we define

$$t(xy) = \{0, \ldots, r(x) + r(y) - 1\}.$$

It follows that we can interpret those vertices as disjoint disks with radii equal to the represented numbers (see Fig. 4). Let $q = (1 + m^B)2^A$, i.e. q is the number of colors used by the first two modules. For every i, we define i-th slot as the set of colors $\{q + 1 + (i - 1) \cdot 4A, \ldots, q + i \cdot 4A\}$. Note that the length of each slot is $4A$. Define also the free space as $Q = \{q + \lceil m/A \rceil \cdot 4A + 2^A + 1, \ldots, q + \lceil m/A \rceil \cdot 4A + 2^A + 2k\}$. Each vertex $x \in V_i$ is either in i-th slot or in the free space Q. However, x has exactly one allowed color in the i-th slot chosen so that we can put all the disks in the i-th slot and they will be disjoint. Let j be such that $r(x) = 2^j$. Then we denote the allowed color by $a_x = q + (i - 1) \cdot 4A + \sum_{r<j} 2 \cdot 2^r + 2^j$. In precise terms, $A(x) = \{a_x\} \cup Q$.

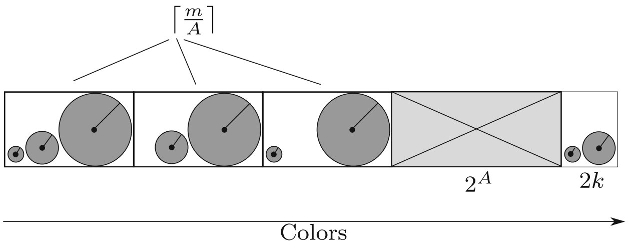

Fig. 4 The parsimonious module

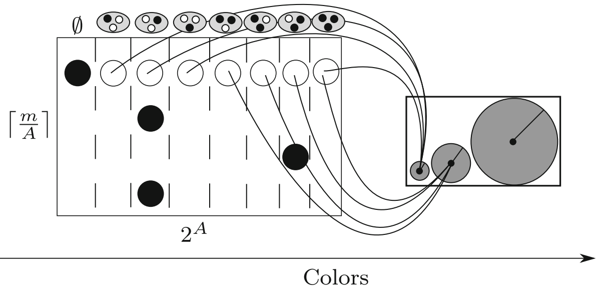

Fig. 5 The interaction between the set choice module and one of the submodules of the parsimonious module. Note that *colors* in $[2^A]$ are ordered according to the cardinality of the chosen collection of sets (0, 1, 1, 1, 2, 2, 2, 3)

Vertices of the i-th submodule have some edges to the vertex s_i of the set choice module. As we mentioned, for a proper assignment c the i-th submodule is going to be a counter representing the number of sets in $\mathcal{S}_i(c)$; in fact the vertex representing 2^j corresponds to the j-th bit of the counter. So if $r(x) = 2^j$ for $x \in V_i$, then $t(s_i x)$ contains all distances d such that $a_x - d$ is a color b from 2^A such that the j-th bit of $|\mathcal{S}_i(b)|$ is 1. Hence, in a proper assignment c, if the j-th bit of the number of sets in $\mathcal{S}_i(c)$ is 1 then x is thrown away from the i-th slot and it is colored by a color from the free space Q. However, the $|Q| = 2k$ so the sum of the radii of the disks thrown out from its slots is at most k. It follows that the total number of the chosen sets is also at most k. Also, if there is a cover $\mathcal{C} \subseteq \mathcal{S}$ of the universe such that $|\mathcal{C}| \leq k$, then for every i, if $|\mathcal{C} \cap \mathcal{S}_i|$ has 1 on the j-th bit we put the vertex of V_i representing 2^j in Q. It is clear that since $|\mathcal{C}| \leq k$ we have enough space for them in Q. Moreover, we do not violate any edge between these vertices and V_S because of the gap 2^A inside the parsimonious module. Together with Claim 4 it implies that $(\mathcal{S}, k)$ is a YES-instance of SET COVER iff (G, t, s) is a YES-instance of GENERALIZED T-COLORING , provided that s is sufficiently large to provide disjoint intervals of colors for all the modules. From the construction we infer that it is sufficient to put $s = 2^A + 2^A \cdot m^B + 4A \cdot \lceil \frac{m}{A} \rceil + 2^A + 2k$ (Fig. 5).

Calculating the parameters Note that $s = O(2^A m^B)$ and in particular our instance is $O(2^A m^B)$-bounded. Moreover, $|V| = \lceil \frac{n}{B} \rceil + \lceil \frac{m}{A} \rceil + \lceil \frac{m}{A} \rceil \cdot (1 + \lfloor \log A \rfloor) + 2 = O\left(\frac{n}{B} + \frac{m}{A} \cdot (1 + \log A)\right) = O\left(\frac{n}{B} + \frac{m}{A} \cdot \max\{1, \log A\}\right)$. Finally, the total number of constraints is bounded by $O^*((\frac{n}{B}+\frac{m}{A}\cdot\max\{1,\log A\})^2\cdot(2^A\cdot m^B)) = O^*\left(2^A\cdot m^B\right)$, i.e., the number of pairs of the vertices times the maximum forbidden distance $s-1$. It ends the proof. □

Corollary 1 *Let (G,k) be an instance of* Dominating Set *where G is a graph on n vertices and $k\in\mathbb{N}$. Then, for any real number $A\in[1,n]$ we can generate in polynomial time an equivalent instance of* Generalized T-Coloring *with $O\left(\frac{n}{A}\cdot\max\{1,\log A\}\right)$ vertices and with $O^*\left((2n)^A\right)$ constraints and such that all the numbers in the instance have $O\left(A\cdot\max\{1,\log n\}\right)$ bits.*

Proof The instance of Dominating Set with n vertices can be transformed to an equivalent instance of Set Cover with n sets and also n elements of the universe in a standard way (the sets are exactly the neighborhoods of the vertices). The number k stays the same. Therefore we can use the Lemma 6 with $A=B$ and $m=n$. □

Theorem 4 *If there exists an algorithm solving* Generalized T-Coloring *in one of the following time complexities:*

(i) $2^{2^{o(\sqrt{n})}}\text{poly}(r)$,
(ii) $2^{n\cdot o\left(\log\ell/\log^2\log\ell\right)}\text{poly}(r)$,

where n is the number of vertices in the input graph, r is the bit size of the input and ℓ is the maximum edge weight, then there exists an algorithm solving Dominating Set *in time $2^{o(n)}$.*

Proof We begin with proving (i). Let us assume that we have an algorithm solving Generalized T-Coloring in time $2^{2^{f(n)}}\text{poly}(r)$ where f is some function such that $f(n)=o\left(\sqrt{n}\right)$. We can assume without loss of generality that f is positive and nondecreasing. Let C be a constant such that Corollary 1 will give us always at most $C\cdot\frac{n}{A}\cdot\max\{1,\log A\}$ vertices. Let α be a positive nondecreasing function such that $\alpha(n)\le\frac{\sqrt{n}}{f(Cn)}$ and $\alpha(n)=\omega(1)$. Such a function always exists because $\frac{\sqrt{n}}{f(Cn)}=\frac{1}{\sqrt{C}}\cdot\frac{\sqrt{Cn}}{f(Cn)}=\omega(1)$. For every instance of Dominating Set with n vertices we can take $A=\frac{n}{\alpha\left(\log^2 n\right)\log n}$ and use Corollary 1 to obtain an instance of Generalized T-Coloring with $O\left(\frac{n}{A}\log A\right)=O\left(\alpha\left(\log^2 n\right)\log^2 n\right)$ vertices and

$$O^*\left((2n)^A\right)=O^*\left(2^{A+A\log n}\right)=O^*\left(2^{O\left(n/(\alpha\log^2 n)\right)}\right)=2^{o(n)}$$

constraints. Moreover the numbers in the instance have polynomial size, so the size of the whole instance is $2^{o(n)}$. Thus this instance can be built in $\text{poly}\left(n,2^{o(n)}\right)=2^{o(n)}$ time. Then we can solve this instance in $2^{2^{f\left(C\cdot\frac{n}{A}\log A\right)}}\text{poly}\left(2^{o(n)}\right)$ time. But $f\left(C\cdot\frac{n}{A}\log A\right)\le f\left(C\cdot\alpha\left(\log^2 n\right)\log^2 n\right)\le\frac{\sqrt{\alpha\left(\log^2 n\right)\log^2 n}}{\alpha\left(\alpha\left(\log^2 n\right)\log^2 n\right)}=\log n\cdot\frac{\sqrt{\alpha\left(\log^2 n\right)}}{\alpha\left(\alpha\left(\log^2 n\right)\log^2 n\right)}=\log n\cdot\frac{\sqrt{\alpha\left(\log^2 n\right)}}{\alpha\left(\log^2 n\right)}\cdot\frac{\alpha\left(\log^2 n\right)}{\alpha\left(\alpha\left(\log^2 n\right)\log^2 n\right)}\le\frac{\log n}{\sqrt{\alpha\left(\log^2 n\right)}}=o\left(\log n\right)$.

So the time of the whole procedure is $2^{o(n)}+2^{2^{o(\log n)}}\text{poly}\left(2^{o(n)}\right)=2^{o(n)}$.

Now we focus on (ii). Let us assume we have an algorithm solving GENERALIZED T-COLORING in time $2^{n \cdot f(\ell)}\text{poly}(m)$ where f is a positive function such that $f(\ell) = o\left(\frac{\log \ell}{\log^2 \log \ell}\right)$. Let $A = \frac{n}{\log^2 n}$. For every instance of DOMINATING SET we can use Corollary 1 to obtain an instance of GENERALIZED T-COLORING with $O\left(\log^2 n \cdot \log \frac{n}{\log^2 n}\right) = O\left(\log^3 n\right)$ vertices, $O^*\left((2n)^{\frac{n}{\log^2 n}}\right) = 2^{O\left(\frac{n}{\log n}\right)} = 2^{o(n)}$ constraints and every number with $O\left(\frac{n}{\log n}\right)$ bits. We can obtain it in poly $\left(n, 2^{o(n)}\right) = 2^{o(n)}$ time. Note that then $\log \ell \leq C\frac{n}{\log n}$ for some constant C. The function $x/\log^2 x$ is nondecreasing for big values of x so for big values of n we have $\log \ell/\log^2 \log \ell \leq C\frac{n}{\log n}/\log^2\left(C\frac{n}{\log n}\right)$. So we can solve our instance of GENERALIZED T-COLORING in time

$$
\begin{aligned}
&2^{O\left(\log^3 n\right)\cdot o\left(C\frac{n}{\log n}/\log^2\left(C\frac{n}{\log n}\right)\right)} \cdot \text{poly}\left(2^{o(n)}\right) = 2^{o\left(n \log^2 n/\log^2\left(C\frac{n}{\log n}\right)\right)} \cdot 2^{o(n)} \\
&\quad = 2^{o\left(n \log^2 n/(\log C + \log n - \log\log n)^2\right)} \cdot 2^{o(n)} = 2^{o\left(n/\left(\frac{\log C}{\log n} + 1 - \frac{\log\log n}{\log n}\right)^2\right)} \cdot 2^{o(n)} \\
&\quad = 2^{o(n)} \cdot 2^{o(n)} = 2^{o(n)}.
\end{aligned}
$$

So we have solved the given instance of DOMINATING SET in time $2^{o(n)} + 2^{o(n)} = 2^{o(n)}$.
□

Corollary 2 *There is no algorithm solving an n-vertex instance of* GENERALIZED T-COLORING *with bit size r and the maximum edge weight ℓ, in any of the listed time complexities*

- $2^{2^{o(\sqrt{n})}}\text{poly}(r)$,
- $2^{n \cdot o\left(\log \ell/\log^2 \log \ell\right)}\text{poly}(r)$,

unless the Exponential Time Hypothesis fails.

Proof Under the ETH assumption there is no algorithm solving DOMINATING SET in time $2^{o(n)}$ where n is a number of the vertices (See [6]). Therefore the claim follows immediately from Theorem 4. □

Regarding the second claim the theorem above, we note that there is a $2^{O(n \log \ell)}$ poly(r)-time algorithm for GENERALIZED T-COLORING , see [12].

6 Further Research

An obvious open problem is to improve the upper bound for ℓ-bounded CHANNEL ASSIGNMENT even more. For a lower bound, Socała [15] has shown that unless ETH fails there is no algorithm solving CHANNEL ASSIGNMENT in time $2^{n \cdot o(\log\log \ell)} r^{O(1)}$ where r is the bit size of the instance. It would be very intersting to get an $O(\log \ell)^n$-time algorithm, or to rule it out under ETH or some other resonable complexity assumption. For a less impressive improvements, it is natural to ask whether partitioning the instance in, say, *thirds* rather than halves leads to a better running time. A direct modification of our approach from this paper does not seem to lead anywhere, since the size of the interface for the middle third is of size $O(l)^{2/3n}$. However, we cannot rule out the possibility that there is some tricky way out, like in the related work of Björklund, Kaski and Kowalik [3].

Acknowledgments The authors thank anonymous reviewers for careful reading and helpful comments.

References

1. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Counting paths and packings in halves. In: Fiat, A., Sanders, P. (eds.) Algorithms - ESA 2009. Lecture Notes in Computer Science, vol. 5757, pp. 578–586. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04128-0_52
2. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion–exclusion. SIAM J. Comput. **39**(2), 546–563 (2009)
3. Björklund, A., Kaski, P., Kowalik, Ł.: Counting thin subgraphs via packings faster than meet-in-the-middle time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'14, pp. 594–603. SIAM (2014)
4. Chen, J., Kneis, J., Lu, S., Mölle, D., Richter, S., Rossmanith, P., Sze, S.H., Zhang, F.: Randomized divide-and-conquer: improved path, matching, and packing algorithms. SIAM J. Comput. **38**(6), 2526–2547 (2009)
5. Cygan, M., Kowalik, L.: Channel assignment via fast zeta transform. Inf. Process. Lett. **111**(15), 727–730 (2011)
6. Fomin, F.V., Kratsch, D., Woeginger, G.J.: Exact (exponential) algorithms for the dominating set problem. In: Proceedings of the WG'04. Lecture Notes in Computer Science, vol. 3353, pp. 245–256 (2004)
7. Hale, W.: Frequency assignment: theory and applications. Proc. IEEE **68**(12), 1497–1514 (1980). doi:10.1109/PROC.1980.11899
8. Horowitz, E., Sahni, S.: Computing partitions with applications to the knapsack problem. J. ACM **21**(2), 277–292 (1974)
9. Husfeldt, T., Paturi, R., Sorkin, G.B., Williams, R.: Exponential algorithms: algorithms and complexity beyond polynomial time (Dagstuhl Seminar 13331). Dagstuhl Rep. **3**(8), 40–72 (2013)
10. Impagliazzo, R., Paturi, R.: On the complexity of k-sat. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)
11. Junosza-Szaniawski, K., Kratochvíl, J., Liedloff, M., Rossmanith, P., Rzążewski, P.: Fast exact algorithm for labeling of graphs. Theor. Comput. Sci. **505**, 42–54 (2013)
12. Junosza-Szaniawski, K., Rzążewski, P.: An exact algorithm for the generalized list t-coloring problem. Discrete Math. Theor. Comput. Sci. **16**(3), 77–94 (2014)
13. Král, D.: An exact algorithm for the channel assignment problem. Discrete Appl. Math. **145**(2), 326–331 (2005)
14. McDiarmid, C.J.H.: On the span in channel assignment problems: bounds, computing and counting. Discrete Math. **266**(1–3), 387–397 (2003)
15. Socała, A.: Tight lower bound for the channel assignment problem. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15), pp. 662–675. SIAM (2015)
16. Traxler, P.: The time complexity of constraint satisfaction. In: Grohe, M., Niedermeier, R. (eds.) IWPEC. Lecture Notes in Computer Science, vol. 5018, pp. 190–201. Springer (2008)