An exact algorithm for the generalized list T-coloring problem

Konstanty Junosza-Szaniawski, Pawel Rzazewski

Discrete Mathematics and Theoretical Computer Science DMTCS vol. **16:3**, 2014, 77–94

An Exact Algorithm for the Generalized List T-Coloring Problem

Konstanty Junosza-Szaniawski* Paweł Rzążewski†

Faculty of Mathematics and Information Science. Warsaw University of Technology, Poland

received 11th Apr. 2013, revised 10th June 2014, accepted 12th June 2014.

The generalized list T-coloring is a common generalization of many graph coloring models, including classical coloring, $L(p,q)$-labeling, channel assignment and T-coloring. Every vertex from the input graph has a list of permitted labels. Moreover, every edge has a set of forbidden differences. We ask for a labeling of vertices of the input graph with natural numbers, in which every vertex gets a label from its list of permitted labels and the difference of labels of the endpoints of each edge does not belong to the set of forbidden differences of this edge. In this paper we present an exact algorithm solving this problem, running in time $\mathcal{O}^*((\tau+2)^n)$, where τ is the maximum forbidden difference over all edges of the input graph and n is the number of its vertices. Moreover, we show how to improve this bound if the input graph has some special structure, e.g. a bounded maximum degree, no big induced stars or a perfect matching.

Keywords: graph labeling, exact algorithm, T-coloring, generalized list T-coloring, channel assignment

1 Introduction

Probably no graph-theoretic problem received as much attention from discrete mathematics and theoretical computer science community, as graph coloring. Various graph coloring models are extensively studied both for their practical motivations (e.g. in frequency assignment or scheduling), and for their interesting theoretical and structural properties. In the multitude of coloring (or labeling) models, some are similar to each other, while others require completely different techniques and approaches.

A natural tendency is to look for the similarities between different models and try to unify and generalize them. In this spirit, Fiala, Král' and Šrekovski [12] defined and studied the so-called generalized list T-coloring problem. An instance of the generalized list T-coloring problem is a triple (G,Λ,t), where $G=(V,E)$ is a graph and Λ and t are functions. Each vertex v of G has a *list of permitted labels* $\Lambda(v)\subseteq\mathbb{N}$. Moreover, each edge e has a list of *forbidden differences* $t(e)\subseteq\mathbb{N}\cup\{0\}$ over that edge.

*k.szaniawski@mini.pw.edu.pl
†p.rzazewski@mini.pw.edu.pl, corresponding author

Moreover, we assume that $0 \in t(e)$ for any $e \in E$. We aim to find a labeling $\varphi\colon V \to \mathbb{N}$, such that $\varphi(v) \in \Lambda(v)$ for every $v \in V$ and $|\varphi(v) - \varphi(w)| \notin t(vw)$ for any edge $vw \in E$.

The notion of the generalized list T-coloring unifies many well-studied graph problems. Here we present some of them and describe how to see them as a special case of our problem.

Graph coloring

Although the graph coloring problem dates back to 19th century, it still raises considerable attention from many researchers. See the book by Jensen and Toft [19] to get some information about the history and many still open problems in graph coloring. In the classical graph coloring problem, we want to assign colors (usually represented by natural numbers) to the vertices of the graph in such a way, that every pair of neighbors receives different colors. The list version of this problem has been introduced independently by Vizing [29] and by Erdös, Rubin and Taylor [10]. Here each vertex has a list of colors and its color has to be chosen from this list (a non-list coloring is a special case of the list coloring with all lists equal). Therefore an instance of the list coloring problem is a pair (G, Λ), where G is a graph and Λ is the function assigning the lists to vertices.

To see that list coloring is a special case of the generalized list T-coloring, consider an instance (G, Λ) of the list -coloring and let (G, Λ, t) be an instance of our problem, where $t(e) = \{0\}$ for all $e \in E$.

$L(p, q)$-labeling

The notion $L(p, q)$-labeling (for integers $p \geq q \geq 1$) is inspired by the frequency assignment problem in telecommunication. We look for a labeling of the vertices G with integers, so that the labels of adjacent vertices differ by at least p and the labels of the vertices with a common neighbor differ by at least q. The best studied case is $L(2, 1)$-labeling, inroduced by Griggs and Yeh [15]. We refer the reader to surveys on this problem by Yeh [30] and Calamoneri [6]. The list version of this problem has been studied for example by Fiala and Škrekovski [13].

Notice that (list) $L(p, q)$-labeling of a graph G is equivalent to an instance (G^2, Λ, t) of the generalized list T-coloring, where G^2 is a square of G, $\Lambda(v)$ is the list of labels available for v (or the set of all colors in a non-list case) and $t(e) = \{0, 1, .., p - 1\}$ for $e \in E(G)$ and $t(e) = \{0, 1, .., q - 1\}$ for $e \in E(G^2) \setminus E(G)$.

It is worth mentioning that the case of the $L(0, 1)$-labeling has also received some attention (see or example Fiala, Golovach and Kratochvíl [11]). However, it is not covered by our model, since we assume that $0 \in t(e)$ for every $e \in E$. Note that the same assumption was used by Fiala, Král' and Šrekovski [12] in their work on the generalized list T-coloring.

Channel assignment

The channel assignment problem is a generalization of $L(p, q)$-labeling. Its instance is a pair (G, ω), where G is a graph and ω is a weight function $\omega\colon E(G) \to \mathbb{N}$. We ask for such a labeling f of vertices of G with natural numbers, that $|f(v) - f(u)| \geq \omega(uv)$ for any $uv \in E(G)$. We refer the reader to the survey by Král' [24] for more information about this problem.

For an instance (G, ω) of the (list) channel assignment problem, we have an equivalent instance (G, Λ, t) of the generalized list T-coloring, where $\Lambda(v)$ is the list of labels available for v (or the set of all labels in a non-list case) and $t(e) = \{0, 1, .., \omega(e) - 1\}$ for $e \in E(G)$.

T-coloring

The last graph coloring model listed here is the so-called T-coloring. It has been first introduced by Hale [16] as *frequency constrained channel assignment problem*. The instance of T-coloring is a pair (G, T), where G is a graph and T is a subset of natural numbers. We ask for such a labeling f of vertices G with natural numbers, that $|f(u) - f(v)| \notin T$ for all $uv \in E(G)$. Unlike the channel assignment problem, T-coloring allows the case when forbidden differences do not form an interval. It is interesting to mention that for $T = \{0, 7, 14, 15\}$ we obtain the model for interferences for the UHF transmitters (see McDiarmid [26]). The list version of this problem has been studied for example by Alon and Zaks [1].

To obtain an instance of the generalized list T-coloring, which is equivalent to given instance of the (list) T-coloring, we have to set $t(e) = T$ for every $e \in E(G)$.

All the problems mentioned above are NP-complete for general graphs and remain so for many restricted graph classes. Therefore the generalized list T-coloring problem is NP-complete as well. When dealing with an NP-hard problem, there is little hope to design an exact algorithm, running in polynomial time. Therefore we try to design exact exponential algorithm with exponential factor in the complexity bound as small as possible.

Many such algorithms have been presented for the best-studied of the mentioned problems, i.e. graph coloring (see for example Lawler [25], Eppstein [9], Byskov [5]). The currently best exact exact algorithm for graph coloring was presented by Björklund *et al.* [3] and runs in time $\mathcal{O}^*(2^n)$[(i)]. It is worth mentioning that this algorithm works also for the $L(0,1)$-labeling problem, since the $L(0,1)$-labeling of G is equivalent to the classical coloring of $G^2 - G$.

Quite a few algorithms for the $L(2,1)$-labeling problem have also been presented (see Havet *et al.* [17], Junosza-Szaniawski and Rzążewski [21, 22]). Currently best exact algorithm was presented by Junosza-Szaniawski *et al.* [20] and has time complexity $\mathcal{O}^*(2.6488^n)$.

An instance of channel assignment is called *ℓ-bounded* if $\omega(e) \leq \ell$ for all $e \in E(G)$. The first exact algorithm for the channel assignment problem with time complexity $\mathcal{O}^*((2\ell+1)^n)$ was proposed by McDiarmid [27]. Then Král' [23] presented the algorithm running in time $\mathcal{O}^*((\ell+2)^n)$. This bound was beaten by Cygan and Kowalik [8], who showed the algorithm with time complexity $\mathcal{O}^*((\ell+1)^n)$. It still remains a great challenge to design an exact algorithm for the channel assignment problem with time complexity bounded by $\mathcal{O}^*(c^n)$ for c being a constant (or to prove that there is no such algorithm, under standard complexity assumptions).

To our best knowledge, T-coloring itself has not raised any attention from the exact algorithms community so far. In this paper we present a method to solve the generalized T-coloring problem. To be specific, we adapt the algorithm for the $L(2,1)$-labeling presented by Junosza-Szaniawski *et al.* [20]. We focus on the case when $t(e) \neq \{0\}$ for at least one one edge e (otherwise we obtain a well-studied list coloring problem, which can be solved in time $\mathcal{O}^*(2^n)$ by adapting the algorithm by Björklund *et al.* [3]). The time and space complexity of our algorithm is bounded by $\mathcal{O}^*((\tau+2)^n)$, where τ is the maximum forbidden difference over all edges of the input graph.

Although the algorithm by Cygan and Kowalik [8] is designed for the channel assignment problem, it can be be adapted to solve the generalized T-coloring problem. Its time complexity is then the same as the time complexity of our algorithm. Their approach uses a well-known inclusion-exclusion principle and fast zeta transform. However, we believe that the method presented in this paper is interesting on its own and can be used to solve many different problems.

[(i)] In the $\mathcal{O}^*$ notation we suppress polynomially bounded terms.

In Section 3.1 we show that the complexity bound of our algorithm can be improved if the input graph has some special structure. We consider bounded degree graphs, $K_{1,d}$-free graphs (for integer d) and graphs having a clique factor, i.e. a spanning subgraph whose every connected component is a clique with at least 2 vertices.

2 Preliminaries

An instance of a the generalized list T-coloring problem is a triple (G, Λ, t), where $G = (V, E)$ is a graph, $\Lambda\colon V \to 2^{\mathbb{N}}$ is a function that assigns to each vertex a set (list) of permitted labels and $t\colon E \to 2^{\mathbb{N}\cup\{0\}}$ is a function that assigns to each edge a set of forbidden differences over that edge. We assume that $0 \in t(e)$ for any $e \in E$. We aim to find a mapping $\varphi\colon V \to \mathbb{N}$, satisfying the following conditions:

1. $\varphi(v) \in \Lambda(v)$ for every $v \in V$,
2. $|\varphi(v) - \varphi(w)| \notin t(vw)$ for every edge $vw \in E$.

Such a function is called a *proper labeling*.

Let Λ_{max} denote the maximum value in the set $\bigcup_{v\in V} \Lambda(v)$. We say that an instance of the generalized list T-coloring problem is τ-bounded if $\max\{\bigcup_{e\in E} t(e)\} \leq \tau$. In this paper we focus on the case when $\tau \geq 1$.

Observe that in any partial labeling of G, a vertex $v \in V$ may have at most $(2\tau+1)\cdot \deg v \leq (2\tau+1)n$ forbidden labels. Thus, if $|\Lambda(v)| > (2\tau + 1)n$, then we may label the graph $G - v$ and then restore v, giving it some non-forbidden label. So we may assume that $|\Lambda(v)| \leq (2\tau+1)n$ for any vertex v and thus:

$$|\bigcup_{v\in V} \Lambda(v)| \leq \sum_{v\in V} |\Lambda(v)| \leq (2\tau + 1)n^2. \tag{1}$$

Assume that there exist two labels $x, y \in \bigcup_{v\in V} \Lambda(v)$ such that:

1. $d := y - x - \tau - 1 > 0$,
2. there is no $z \in \bigcup_{v\in V} \Lambda(v)$ such that $x < z < y$.

It is easy to verify that the instance (G, Λ, t) of the generalized list T-coloring is equivalent to the instance (G, Λ', t), where $\Lambda'(v) = \{c \in \Lambda(v)\colon c \leq x\} \cup \{c - d\colon c \in \Lambda(v) \text{ and } c > x\}$ for every $v \in V$ Informally speaking, every label greater than x is just shifted down by d. Since they are still greater than $x + \tau$, no new conflict between labels appears (and clearly no conflicts disappear). We can repeat this transformation until the difference of every pair of subsequent labels in $\bigcup_{v\in V} \Lambda(v)$ is at most $\tau+1$. For the similar reason we may assume that $1 \in \bigcup_{v\in V} \Lambda(v)$ (otherwise we can shift all the labels down). Combining this with the formula (1), we may assume that:

$$\Lambda_{max} \leq |\bigcup_{v\in V} \Lambda(v)| \cdot (\tau + 2) \leq (2\tau + 1)(\tau + 2)n^2. \tag{2}$$

Let $[\tau + 1]$ denote the set $\{0, 1, .., \tau + 1\}$ and $[\![\tau + 1]\!]$ denote the set $[\tau + 1] \cup \{\bar{0}\}$, where $\bar{0}$ is a special symbol, whose meaning will be made clear later. Note that $|[\tau + 1]| = \tau + 2$.

For a vector $\mathbf{w}$ and a set of vectors A let $A_{\mathbf{w}}$ denote the set $\{\mathbf{v}\colon \mathbf{wv} \in A\}$ (by $\mathbf{wv}$ we denote the concatenation of vectors $\mathbf{w}$ and $\mathbf{v}$). Vector $\mathbf{w}$ is also called a *prefix* of a vector $\mathbf{wv}$.

3 The algorithm

Let (G, Λ, t) be an instance of the general list T-coloring problem. We assume that the graph G is connected – in other case we may label each of its connected components separately. For a graph $G = (V, E)$ we consider some ordering $v_1, v_2, .., v_n$ of the vertices in V.

For a partial k-labeling $\varphi\colon V \to \{1, \ldots, k\}$ let $\Gamma_\varphi\colon V \to \{true, false\}$ be a Boolean function saying, if φ can be extended by labeling a particular vertex with $k+1$. Formally, $\Gamma_\varphi(v)$ is $true$ if and only if both the following conditions are satisfied:

1. $k + 1 \in \Lambda(v)$,
2. there is no vertex $w \in N(v)$, which is labeled by φ with $(k + 1) - \varphi(w) \in t(vw)$.

Our strategy is to construct a labeling of G in a way similar to the one presented by Junosza-Szaniawski *et al.* [20].

For every $k \in \{1, .., \Lambda_{max}\}$ we introduce a set of vectors $T[k] \subseteq [\![\tau + 1]\!]^n$. The set $T[k]$ contains a vector $\mathbf{a} \in [\![\tau + 1]\!]^n$ if and only if there exists a partial labeling $\varphi\colon V \to \{1, 2, .., k\}$ such that :

1. $\mathbf{a}_i = 0$ iff v_i is not labeled by φ and $\Gamma_\varphi(v) = true$,
2. $\mathbf{a}_i = \bar{0}$ iff v_i is not labeled by φ and $\Gamma_\varphi(v) = false$,
3. $\mathbf{a}_i = 1$ iff $\varphi(v_i) \leq k - \tau$, and
4. $\mathbf{a}_i = j$ iff $\varphi(v_i) = k + j - \tau - 1$ for $j \in \{2, 3, .., \tau + 1\}$.

In other words, this encoding of partial k-labelings unifies the sets of vertices labeled with labels not exceeding $k - \tau$, since they do not interfere with the next label to be assigned, which is $k + 1$. Moreover, 0 indicates an unlabeled vertex, which can be labeled with label $k + 1$, while $\bar{0}$ indicates an unlabeled vertex with label $k + 1$ blocked.

Once we have computed $T[\Lambda_{max}]$, we can easily verify if there exists a proper labeling of the input graph – it corresponds to a vector having no 0s and $\bar{0}$s (as they encode unlabeled vertices). Formally, an instance of the generalized list T-coloring problem is a YES-instance if and only if $T[\Lambda_{max}] \cap \{1, 2, \ldots, \tau, \tau + 1\}^n \neq \emptyset$.

The only thing left is to compute the tables $T[k]$ quickly. Let us define a partial function: $\oplus\colon [\![\tau + 1]\!] \times \{0, 1\} \to [\tau + 1]$ in the following way:

$$x \oplus y = \begin{cases} 0 & \text{if } x \in \{0, \bar{0}\} \text{ and } y = 0 \\ 1 & \text{if } x \in \{1, 2\} \text{ and } y = 0 \\ x - 1 & \text{if } x \in \{3, 4, .., \tau + 1\} \text{ and } y = 0 \\ \tau + 1 & \text{if } x = 0 \text{ and } y = 1 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The table below shows the values of $\oplus$ for different inputs (entry „$-$” means that the value is undefined)

.

$\oplus$	0	$\bar{0}$	1	2	3	4	$\ldots$	τ	$\tau+1$
0	0	0	1	1	2	3	$\ldots$	$\tau-1$	τ
1	$\tau+1$	$-$	$-$	$-$	$-$	$-$	$\ldots$	$-$	$-$

We generalize $\oplus$ to vectors coordinate-wise, i.e.

$$x_1x_2..x_m \oplus y_1y_2..y_m = \begin{cases} (x_1 \oplus y_1)..(x_m \oplus y_m) & \text{if } x_i \oplus y_i \text{ is defined for all } i \in \{1,..,m\}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

For two sets of vectors $A \subseteq [\![\tau+1]\!]^m$ and $B \subseteq \{0,1\}^m$ we define

$$A \oplus B = \{\, \mathbf{a} \oplus \mathbf{b} \colon \mathbf{a} \in A,\ \mathbf{b} \in B,\ \mathbf{a} \oplus \mathbf{b} \text{ is defined} \,\}.$$

Let $P \subseteq \{0,1\}^n$ be the set of the characteristic vectors of all independent sets of G. Formally, $\mathbf{p} \in P$ if and only if there is an independent set $X \subseteq V$ such that for all $i \in \{1,2,..,n\}$ holds $\mathbf{p}_i = 1$ if and only if $v_i \in X$.

We shall use $T[k] \oplus P$ to find $T[k+1]$.

For a vector $\mathbf{b} \in [\tau+1]^n$ let $\overline{\mathbf{b}}^{(k)} = \overline{\mathbf{b}}_1^{(k)} \dots \overline{\mathbf{b}}_n^{(k)}$ denote a vector in $[\![\tau+1]\!]^n$ such that:

$$\overline{\mathbf{b}}_i^{(k)} = \begin{cases} 0 & \text{iff } \mathbf{b}_i = 0 \text{ and } k+2 \in \Lambda(v_i) \text{ and there is no } v_j \in N(v_i) \\ & \text{such that } \tau - \mathbf{b}_j + 2 \in t(v_iv_j) \\ \bar{0} & \text{iff } \mathbf{b}_i = 0 \text{ and } k+2 \notin \Lambda(v_i) \text{ or there exists } v_j \in N(v_i) \\ & \text{such that } \tau - \mathbf{b}_j + 2 \in t(v_iv_j) \\ j & \text{iff } \mathbf{b}_i = j \text{ for } j \in \{1,..,\tau+1\}. \end{cases}$$

Analogously, for a set of vectors B let $\overline{B}^{(k)}$ denote the set $\{\overline{\mathbf{b}}^{(k)} \colon \mathbf{b} \in B\}$.

Lemma 1 *For all $k \geq 1$ it holds that $\overline{T[k] \oplus P}^{(k)} = T[k+1]$.*

Proof: First, we shall prove $\overline{T[k] \oplus P}^{(k)} \subseteq T[k+1]$.

Let $\mathbf{b}'$ be a vector from $\overline{T[k] \oplus P}^{(k)}$. Thus there exist $\mathbf{b} \in T[k]$ and $\mathbf{p} \in P$ such that $\overline{\mathbf{b} \oplus \mathbf{p}}^{(k)} = \mathbf{b}'$. Let φ be a partial k-labeling corresponding to $\mathbf{b}$ and X be an independent set encoded by $\mathbf{p}$.

Note that since $\mathbf{b} \oplus \mathbf{p}$ is defined, the labeling φ could be extended by labeling every vertex from X with label $k+1$, obtaining a proper partial $(k+1)$-labeling φ'. Moreover, φ' can be extended to a proper partial $(k+2)$-labeling by labeling a vertex v_i with label $k+2$ if and only if:

- v_i is not labeled by φ' (so $\mathbf{b}_i \in \{0, \bar{0}\}$ and $\mathbf{p}_i = 0$),
- $k+2 \in \Lambda(v)$,
- there is no neighbor v_j of v_i, which is labeled by φ' such that $(k+2) - \varphi'(v_j) \in t(v_iv_j)$. Such a conflict cannot occur if $\varphi'(v) \leq k - \tau + 1$, which is equivalent to $\mathbf{b}'_j = 1$ (which happens when $\mathbf{b}_j = 1$ or $\mathbf{b}_j = 2$). In the remaining cases we have $(k+2) - \varphi'(v_j) \in t(v_iv_j)$ equivalent to $\tau - \mathbf{b}'_j + 2 \in t(v_iv_j)$.

To sum up, one can observe that $\mathbf{b}'$ corresponds to a partial $(k+1)$-labeling φ' such that:

1. $\varphi'(v) = \varphi(v)$ iff $\mathbf{b}'_i \notin \{0, \bar{0}, \tau+1\}$,

2. $\varphi'(v) = k+1$ iff $v \in X$ (thus $\mathbf{b}'_i = \tau + 1$),
3. v_i is unlabeled by φ', $k+2 \in \Lambda(v_i)$ and there is no neighbor w of v_i with $k+2-\varphi'(w) \in t(v_iw)$ iff $\mathbf{b}'_i = 0$,
4. v_i is unlabeled by φ', but $k+2 \notin \Lambda(v_i)$ or there is a neighbor w of v_i with $k+2-\varphi'(w) \in t(v_iw)$ iff $\mathbf{b}'_i = \bar{0}$,

Thus $\mathbf{b}' \in T[k+1]$.

On the other hand, let $\mathbf{b}' \in T[k+1]$. Let φ' be a partial labeling corresponding to $\mathbf{b}'$ and φ be partial labeling defined as follows:

$$\varphi(v) = \begin{cases} \varphi'(v) & \text{if } \varphi'(v) \neq k+1, \\ \text{unlabeled} & \text{otherwise.} \end{cases}$$

Since φ is a partial k-labeling of G, there exists $\mathbf{b} \in T[k]$ corresponding to it. Clearly the set $X = \{v\colon \varphi'(v) = k+1\}$ is independent in G (since $0 \in t(e)$ for all $e \in E$) and therefore its characteristic vector $\mathbf{p}$ belongs to P. Note that $\mathbf{b}' = \overline{\mathbf{b} \oplus \mathbf{p}}^{(k)}$ and therefore $\mathbf{b}' \in \overline{T[k] \oplus P}^{(k)}$. □

Moreover, observe that if we set $T[0] := \{0^n\}$, the following holds: $T[1] = \overline{T[0] \oplus P}^{(0)}$.

Note that computing $T[k+1]$ from $T[k] \oplus P$ takes time $\mathcal{O}(|T[k+1]| \cdot n^2)$ (for all pairs of vertices we have to check if they are adjacent and for every vertex we have to check if $k+2$ is on its list of permitted colors). Having computed all sets $T[k]$, determining the optimal span can be performed in time linear in sizes of these sets.

To compute $T[k] \oplus P$ efficiently we will partition the vertex set into subsets of a bounded size (they shall be defined later, but for a general result the partition to singletons is suitable). Let $\mathcal{S} = (S_1, S_2, .., S_r)$ be an ordered partition of V. Let $s_i := |S_i|$ for all $i \in \{1, .., r\}$ (we require that all s_i's are bounded by some constant D). Moreover, the ordering of sets in $\mathcal{S}$ corresponds to the ordering of vertices of the graph (the vertices within each S_i appear in any order):

$$\underbrace{v_1, v_2, \ldots, v_{s_1}}_{S_1}, \underbrace{v_{s_1+1}, v_{s_1+2}, \ldots, v_{s_1+s_2}}_{S_2}, \ldots, \underbrace{v_{n-s_r+1}, v_{n-s_r+2}, \ldots, v_n}_{S_r}.$$

Then we shall process each of the subsets at once, in every step considering the first set from $\mathcal{S}$. After that we remove S_1 from $\mathcal{S}$ and reduce the index of every remaining set in $\mathcal{S}$ by one, so that the first one is still called S_1. The formula (3) describes a single step of this procedure.

$$T[k] \oplus P = \bigcup_{\substack{\mathbf{b} \in [\![\tau+1]\!]^{s_i} \\ \mathbf{p} \in \{0,1\}^{s_i} \\ \text{s.t. } \mathbf{b} \oplus \mathbf{p} \text{ is defined}}} (\mathbf{b} \oplus \mathbf{p})(T[k]_{\mathbf{b}} \oplus P_{\mathbf{p}}). \tag{3}$$

To show where the advantage of processing whole sets S_i at once, consider the following example. If we process each vertex separately, at each step we have to consider $\tau + 2$ prefixes $(0, 1, \ldots, \tau + 1)$. Suppose now that our input graph has a perfect matching $\mathcal{S}$. Then, by processing two adjacent vertices at once, we can omit all prefixes in the form aa for $a \in \{2, 3, \ldots, \tau + 1\}$. This is because each of values in $\{2, 3, \ldots, \tau + 1\}$ describes a single label and no two adjacent vertices can get the same label. Therefore

instead of considering all $(\tau+2)^2$ prefixes in $[\tau+1]\times[\tau+1]$ (as we would do when processing each vertex separately), we have to deal with $(\tau+2)^2-\tau$ prefixes. One can observe that the choice of the partition $\mathcal{S}$ is crucial for this approach and will be discussed in Section 3.1.

We can rewrite the formula (3) in the following way.

$$T[k]\oplus P=\bigcup_{\substack{\mathbf{a}\in[\tau+1]^{s_i}\\ \mathbf{p}\in\{0,1\}^{s_i}}}\mathbf{a}\left[\left(\bigcup_{\substack{\mathbf{b}\in[\![\tau+1]\!]^{s_i}\\ \text{s.t. }\mathbf{b}\oplus\mathbf{p}=\mathbf{a}}}T[k]_{\mathbf{b}}\right)\oplus P_{\mathbf{p}}\right]$$

The computation can be omitted whenever the prefix $\mathbf{a}$ is infeasible, i.e. it cannot appear in any vector of $T[k]\oplus P$. Thus $\mathbf{p}$ is uniquely determined by the choice of $\mathbf{a}$: we have $\mathbf{p}_i=1$ whenever $\mathbf{a}_i=\tau+1$ (and $\mathbf{p}_i=0$ otherwise). See the pseudo-code of the Algorithm 1 for another description of the computation of $T[k+1]$. The input arguments are: a graph G, a partition of its vertex set $(S_1,S_2,\ldots,S_r)$, previously computed table $T[k]$ and the set P of encodings of independent sets in G.

Algorithm 1: `Compute`$(G,(S_1,S_2,\ldots,S_r),T[k],P)$

if $r=0$ **then return** $\emptyset$
$Q\leftarrow\emptyset$
foreach $\mathbf{a}\in[\tau+1]^{s_1}$, *being a feasible prefix of a vector in* $T[k]\oplus P$ **do**
  $\mathbf{p}\leftarrow$ the characteristic vector of the set $\{v_i\colon \mathbf{a}_i=\tau+1\}$
  $A\leftarrow\emptyset$
  foreach $b\in[\![\tau+1]\!]^{s_1}$ *such that* $\mathbf{b}\oplus\mathbf{p}=\mathbf{a}$ **do**
    $A\leftarrow A\cup T[k]_{\mathbf{b}}$
  if $A\neq\emptyset$ **then**
    $Q'\leftarrow$ `Compute`$(G,(S_2,\ldots,S_r),A,P_{\mathbf{p}})$
    $Q\leftarrow\{\mathbf{av}\colon \mathbf{v}\in Q'\}$
return Q

Finally, all sets $T[k]$ are computed and the solution is found by the Algorithm 2. Again, G is the input graph and $\mathcal{S}$ is the partition of its vertex set.

Algorithm 2: `Solve-GLTC` $(G,\mathcal{S})$

$P\leftarrow$ a set of characteristic vectors of independent sets of G
$T[0]\leftarrow\{0^n\}$
for $k\leftarrow 1$ **to** Λ_{max} **do**
  $Q\leftarrow$ `Compute`$(G,\mathcal{S},T[k-1],P)$
  $T[k]\leftarrow\overline{Q}^{(k-1)}$
if $T[\Lambda_{max}]\cap\{1,2,\ldots,\tau+1\}^n\neq\emptyset$ **then return** YES
else return NO

To estimate the computational complexity of this approach, we have to calculate the number of pairs $\mathbf{a}$, $\mathbf{p}$, for which there exists at least one $\mathbf{b}$ such that $\mathbf{b}\oplus\mathbf{p}=\mathbf{a}$. Notice that if we fix $\mathbf{a}$, then $\mathbf{p}_i=1$ whenever $\mathbf{a}_i=\tau+1$ (otherwise $\mathbf{p}_i=0$). Therefore the number of such pairs $\mathbf{a},\mathbf{p}$ is equal to the number

of segments $\mathbf{a}$ (corresponding to a partial labeling of $G[S_i]$ from lists $\Lambda|_{S_i}$), which can appear in a vector in $T[k+1]$ under a partition $\mathcal{S}$. Let $f_i(k+1, \Lambda)$ denote this number.

It is not hard to observe that for any choice of Λ we have $f_i(k, \Lambda) \leq f_i(k, \Lambda')$, where $\Lambda'(v) = \{1, 2, .., \Lambda_{max}\}$ for every $v \in V$. Thus we define:

$$f_i := \max_{k \leq \Lambda_{max}} f_i(k, \Lambda').$$

In each recursive computation we have to prepare up to f_i pairs of sets of vectors of length $n - s_i$ and compute $\oplus$ on these pairs. Preparing the recursive calls and combining their results takes only time linear in the sizes of the tables $\overline{T}[k]$ and P. The total size of $\overline{T}[k]$ (for each k) is equal to the total size of $T[k]$, which is at most $n \cdot \prod_{i=1}^{r} f_i$ (recall that each vector has n coordinates). Observe that for any independent set X, we can label all the vertices from X with label 1, obtaining a proper partial labeling of G from lists Λ' with some corresponding vector in $T[1]$. Thus the total size of P does not exceed the total size of $T[1] \leq n \cdot \prod_{i=1}^{r} f_i$.

So the time needed to compute $\overline{T}[k] \oplus P$ in a way shown in the Algorithm 1 is given by the following recursive formula (recall that in every step we remove S_1 from $\mathcal{S}$ and the index of every remaining set in $\mathcal{S}$ is reduced by one, so that the first one is still called S_1):

$$\begin{aligned}
F'(n) \leq & \underbrace{\mathcal{O}(1)}_{\text{lines 1 – 3}} + f_1 \cdot \left(\underbrace{\mathcal{O}(1)}_{\text{lines 4–5, 8}} + \underbrace{\mathcal{O}(1) \cdot n \cdot |T[k]|}_{\text{lines 6–7, 10}} + \underbrace{n \cdot |P|}_{\text{prepare } P_{\mathbf{p}}} + \underbrace{F'(n - s_1)}_{\text{line 9}} \right) \\
\leq & \mathcal{O}(1) + f_1 \cdot \left(\mathcal{O}(1) + \mathcal{O}(1) \cdot n \cdot \prod_{i=1}^{r} f_i + F'(n - s_1) \right) = \\
\leq & \mathcal{O}(1) + \mathcal{O}(1) \cdot n \cdot \prod_{i=1}^{r} f_i + f_1 \cdot F'(n - s_1) \\
\leq & c \cdot n \cdot \prod_{i=1}^{r} f_i + f_1 \cdot F'(n - s_1)
\end{aligned}$$

for some constant c. We can verify by induction that the above inequality implies that $F'(n) \leq c \cdot n^2 \cdot \prod_{i=1}^{r} f_i$ for large n. Indeed, by induction hypothesis we have:

$$\begin{aligned}
F'(n) \leq & c \cdot n \cdot \prod_{i=1}^{r} f_i + f_1 \cdot F'(n - s_1) \\
\leq & c \cdot n \cdot \prod_{i=1}^{r} f_i + \left(f_1 \cdot c \cdot (n - s_1)^2 \cdot \prod_{i=2}^{r} f_i \right) \\
= & c \cdot n^2 \cdot \prod_{i=1}^{r} f_i \cdot \left(\frac{1}{n} + \frac{(n - s_1)^2}{n^2} \right) \\
= & c \cdot n^2 \cdot \prod_{i=1}^{r} f_i \cdot \underbrace{\left(1 - \frac{(2s_1 - 1)n + s_1^2}{n^2} \right)}_{\leq 1} \\
\leq & c \cdot n^2 \cdot \prod_{i=1}^{r} f_i
\end{aligned}$$

Recall that $\overline{T}[k]$ can be obtained from $T[k]$ in time $\mathcal{O}(|T[k]| \cdot n^2) = \mathcal{O}(n^2 \cdot \prod_{i=1}^{r} f_i)$. Since we need to compute sets $T[k]$ for all $k \leq \Lambda_{max}$, we obtain the total running time of

$$\begin{aligned}
F(n) = & \mathcal{O}\left(\Lambda_{max} \left(n^2 \cdot \prod_{i=1}^{r} f_i + \cdot F'(n) \right) \right) = \\
= & \mathcal{O}\left(\Lambda_{max} \left(n^2 \cdot \prod_{i=1}^{r} f_i + c \cdot n^2 \cdot \prod_{i=1}^{r} f_i \right) \right) \\
= & \mathcal{O}\left(\Lambda_{max} \cdot n^2 \cdot \prod_{i=1}^{r} f_i \right).
\end{aligned}$$

Recall that by (2) we have $\Lambda_{max} \leq (2\tau + 1)(\tau + 2)n^2$. Thus we obtain the following bound:

$$F(n) = \mathcal{O}^*\left(\prod_{i=1}^{r} f_i \right). \tag{4}$$

The space complexity of the algorithm is bounded by the total size of sets $T[k]$ and the set P. Therefore it is bounded by the same expression as computational complexity, i.e. $\mathcal{O}^*\left(\prod_{i=1}^{r} f_i\right)$.

3.1 Complexity bounds

In this section we shall consider several possible partitions $\mathcal{S}$ of the vertex set and use them to bound the complexity of the algorithm with functions of various invariants of G.

Theorem 1 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on a graph G with n vertices in time $\mathcal{O}^*((\tau + 2)^n)$.*

Proof: Let $v_1, v_2, \ldots, v_n$ be an arbitrary ordering of vertices of G and let $S_i = \{v_i\}$ for $i \in \{1, .., n\}$. Clearly there are at most $\tau + 2$ prefixes $\mathbf{a}$ of length 1, which can appear in any vector from $T[k] \oplus P$. Using formula (4) we obtain the bound for the complexity $F(n) = \mathcal{O}^*(\prod_{i=1}^{n}(\tau + 2)) = \mathcal{O}^*((\tau + 2)^n)$. □

However, we can improve it by a more careful construction of the partition $\mathcal{S}$. In general, if we can partition the vertex set into subsets, each of which induces a graph with some simple structure, we can estimate the number of infeasible prefixes. The remainder of this section contains technical lemmas describing the construction of partitions into stars and cliques.

3.1.1 Partitions into stars

Let $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ be a partition of the vertex set of G, such that for any $i = 1, 2, \ldots, r$ we have $s_i \geq 2$ and $G[S_i]$ has a spanning subgraph, which is a star. We call such a partition a *star partition* of G. Note that every connected graph (with at least 2 vertices) has a star partition. Some ways of constructing star partitions will be described further in this section.

Lemma 2 *Let G be a graph with n vertices having a star partition $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ such that $2 \leq s_i \leq D + 1$ for some constant D and each $i \in 1, 2, \ldots, r - 1$ and and $s_r \leq D'$ for some constant D'. The algorithm `Solve-GLTC` solves the τ-bounded generalized list T-coloring problem on G in time*

$$\mathcal{O}^*\left((\tau^2 + 3\tau + 4)^{n/2} + (2(\tau + 2)^D + \tau(\tau + 1)^D)^{\frac{n}{(D-1)}}\right).$$

Proof: Let us consider a subgraph $G[S_i]$ for some $i = 1, 2, \ldots, r$. Without a loss of generality let v_j be a central vertex of the spanning star of $G[S_i]$ and $S_i = \{v_j, .., v_{j+s_i-1}\}$. Note that for any $\mathbf{a} \in T[k] \oplus P$ if $\mathbf{a}_j = h \in \{2, .., \tau + 1\}$, then $\mathbf{a}_{j'} \neq h$ for $j' \in \{j + 1, .., j + d_i - 1\}$, since h represents a single label and each label of any feasible (possibly partial) labeling induces an independent set in G.

Hence the number of possible s_i-element prefixes of a vector $\mathbf{a}$ in $T[k] \oplus P$ is at most

$$f_i \leq \underbrace{2(\tau + 2)^{s_i - 1}}_{\mathbf{a}_j \in \{0,1\}} + \underbrace{\tau(\tau + 1)^{s_i - 1}}_{\mathbf{a}_j \notin \{0,1\}}.$$

This combined with formula (4) gives us the following bound on the complexity:

$$\begin{aligned}
F(n) &= \mathcal{O}^*(\prod_{i=1}^{r} f_i) \\
&= \mathcal{O}^*\left(\prod_{i=1}^{r}(2(\tau + 2)^{s_i - 1} + \tau(\tau + 1)^{s_i - 1})\right) \\
&= \mathcal{O}^*\left(\prod_{i=1}^{r-1}(2(\tau + 2)^{s_i - 1} + \tau(\tau + 1)^{s_i - 1}) \underbrace{(2(\tau + 2)^{s_r - 1} + \tau(\tau + 1)^{s_r - 1})}_{\text{constant}}\right) \\
&= \mathcal{O}^*\left(\prod_{i=1}^{r-1}(2(\tau + 2)^{s_i - 1} + \tau(\tau + 1)^{s_i - 1})\right).
\end{aligned}$$

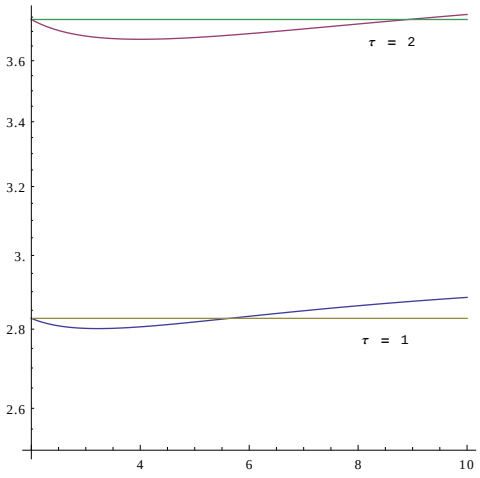

Fig. 1: The values of $\alpha_\tau(d)$ compared with $\alpha_\tau(2)$ for $\tau = 1, 2$.

Note that the value of $\prod_{i=1}^{r-1}(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1})$ is maximized if all s_i's are equal. Consider this case and let $d := s_1 = \ldots = s_{r-1}$. Note that $d \leq \frac{n}{r-1}$ and therefore $r - 1 \leq n/d$. Thus we obtain the following:

$$\begin{aligned} F(n) &= \mathcal{O}^*\left(\prod_{i=1}^{r-1}(2(\tau+2)^{s_i-1} + \tau(\tau+1)^{s_i-1})\right) \\ &= \mathcal{O}^*\left(\prod_{i=1}^{n/d}(2(\tau+2)^{d-1} + \tau(\tau+1)^{d-1})\right) \\ &= \mathcal{O}^*\left((2(\tau+2)^{d-1} + \tau(\tau+1)^{d-1})^{n/d}\right). \end{aligned}$$

By analyzing the derivative of the function $\alpha_\tau(d) = (2(\tau+2)^{d-1} + \tau(\tau+1)^{d-1})^{1/d}$ (defined for $d \geq 2$), one can observe that for every fixed τ there exists d_0 such that $\alpha_\tau(d)$ is decreasing for $d < d_0$ and increasing for $d > d_0$. Since $d \leq D + 1$, we can bound the value of $\alpha_\tau(d)$ by $\max(\alpha_\tau(2), \alpha_\tau(D+1))$. Having in mind that $\tau \geq 1$, we obtain the following solution.

$$\begin{aligned} F(n) &= \mathcal{O}^*\left(\alpha_\tau(2)^n + \alpha_\tau(D+1)^n\right) \\ &= \mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2} + (2(\tau+2)^D + \tau(\tau+1)^D)^{\frac{n}{(D+1)}}\right) \end{aligned}$$

□

Observe that for every τ there exists d_τ such that $\alpha_\tau(d) > \alpha_\tau(2)$ for any $d \geq d_\tau$ (see Figure 1).

The remainder of this section is devoted to various ways of constructing the initial star factor $\mathcal{S}$.

Remark.

Notice that a star partition can be constructed from a spanning tree of our input graph. Let us consider T being a spanning tree of G. Let v and u be, respectively, the end-vertex and its neighbor on a longest

path in T. If T is not a star, then all neighbors of u in T except exactly one are leaves in T. We include the set S_i consisting of u and all its neighbors which are leaves in T to our partition $\mathcal{S}$. Then we proceed recursively with the tree $T \setminus S_i$. If T is a star, we set $S_r = V(T)$ and finish.

Moreover, notice that if we construct our star partition using spanning tree T, in a way described, each set S_i (for $i \in \{1, 2, \ldots, r-1\}$) has at most $\Delta(T)$ elements, while S_r has at most $\Delta(T)+1$ elements.

Observe $\Delta(T) \leq \Delta(G)$ for any spanning tree T of G, we obtain the following corollary.

Corollary 1 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on G with maximum degree bounded by a constant Δ in time*

$$\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2} + (2(\tau+2)^{\Delta-1} + \tau(\tau+1)^{\Delta-1})^{n/\Delta}\right).$$

The following theorem shows that we may obtain a star partition consisting of smaller stars (and therefore a better bound on the complexity of the algorithm).

Theorem 2 (Amashi, Kano [2], Payan) *Let $D \geq 2$ be an integer such that $\Delta(G)/\delta(G) \leq D$. Then G has a star factor $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ such that $s_i \leq D+1$ for any $i = 1, 2, \ldots, r$.*

From this we can get a significantly better bound for regular graphs (in fact in works for a much wider class of graph G with $\Delta(G) \leq 2\delta(G)$).

Corollary 2 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on a regular graph with n vertices in time $\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}\right)$.*

We can improve this bound for graphs with no big induced stars. Let d be number, such that G has no induced $K_{1,d}$ star (i.e. it is a $K_{1,d}$-free graph).

The simplest and probably best-studied class with such a property are claw-free graphs (i.e. $K_{1,3}$-free graphs, see for example Brandstädt *et al.* [4] for more information). Sumner [28] showed that every claw-free graph with even number of vertices has a perfect matching. Therefore it has a star partition with every set of cardinality 2 (but at most one set with cardinality 3). This observation gives us the following bound.

Corollary 3 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on a claw-free graph with n vertices in time $\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}\right)$.*

However, we may obtain a similar improvement for $K_{1,d}$-free graph for different d.

Lemma 3 *Every connected $K_{1,d}$-free graph G has a star partition $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ such that $s_i \leq d-1$ for any $i = 1, 2, \ldots, r-1$ and $s_r \leq d$.*

Proof: Again we shall construct $\mathcal{S}$ using a spanning tree. If G has at most d vertices, we set $S_r = V(G)$ and finish. Otherwise, let T be a spanning tree of G and let v_1, u and x be, respectively, the first, the second and the third vertex of the longest path in T. Notice that v_1 is a leaf in T and every neighbor of u in T but at most one vertex (i.e. x) is a leaf in T as well. Let $\{v_1, v_2, \ldots, v_k\}$ be the set of neighbors of u, which are leaves in T. We shall consider two cases.

Case 1: If $k \leq d-2$, then we include the set $\{u, v_1, v_2, \ldots, v_k\}$ to our partition and proceed with the tree $T \setminus \{u, v_1, v_2, \ldots, v_k\}$. This set has at most $d-1$ vertices.

Case 2: Suppose now that $k \geq d-1$. Observe that if $k = d-1$, then x is not a leaf, since we assumed that G has at least $d+1$ vertices. Therefore the set $\{u\} \cup \{x, v_1, v_2, \ldots, v_k\}$ does not induce a star in G (as G is $K_{1,d}$-free). Thus $v_i v_j \in E(G)$ for some $1 \leq i < j \leq k$ or $v_i x \in E(G)$ for some $1 \leq i \leq k$. We shall recursively transform our spanning tree T using one of the following transformations.

(T1) If $v_i v_j \in E(G)$ for some $1 \leq i < j \leq k$ (without loss of generality let $i = 1$ and $j = 2$), add it to T and remove from T the edge $v_1 u$ ($H := (V(G), E(T) \cup \{v_1 v_2\} \setminus \{u v_1\})$ (see Figure 2).

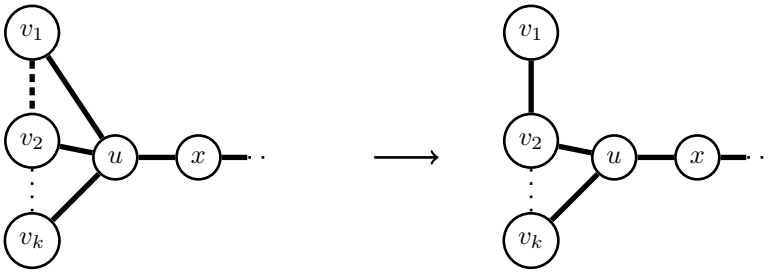

Fig. 2: Transformation (T1)

(T2) If $v_i x \in E(G)$ for some $1 \leq k$ (without loss of generality let $i = 1$), add it to T and remove from T the edge $v_1 u$ ($H := (V(G), E(T) \cup \{v_1 x\} \setminus \{u v_1\})$ (see Figure 3).

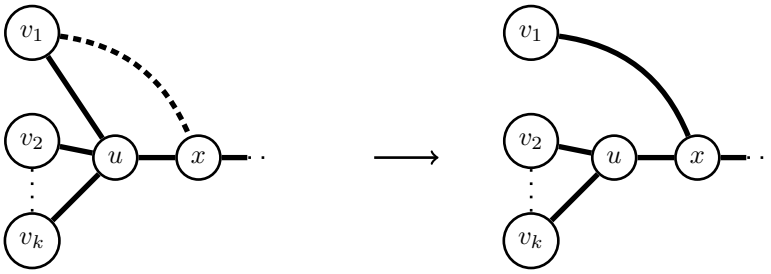

Fig. 3: Transformation (T2)

Notice that if none of the above transformations can be applied to T, then $k \leq d-2$ and this is exactly Case 1. This finishes the proof. □

Having such a partition, we obtain the following complexity bound.

Corollary 4 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on a $K_{1,d}$-free graph with n vertices in time*

$$\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2} + (2(\tau+2)^{d-2} + \tau(\tau+1)^{d-2})^{n/(d-1)}\right).$$

Another class of graphs we want to mention are unit disk graphs, i.e. intersection graphs of unit disks on a plane (see for example Clark *et al.* [7]). They are particularly interesting due to their applications in modeling of ad-hoc networks. It is widely known that unit disk graphs are $K_{1,7}$-free.

Corollary 5 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on a unit disk graph with n vertices in time*

$$\mathcal{O}^*\left((\tau^2+3\tau+4)^{n/2}+(2(\tau+2)^5+\tau(\tau+1)^5)^{n/6}\right).$$

Recall that star partitions consisting of small stars yield lower complexity bound of the algorithm. Therefore, if we want to construct them using a spanning tree, the maximum degree of such a tree should be the lowest possible. However, deciding if the input graph has a spanning tree with maximum degree at most k is NP-complete for every $k \geq 2$ (see Garey, Johnson [14]).

3.1.2 Partitions into cliques

Another subgraphs in the input graph may also prove useful in constructing the partition $\mathcal{S}$. Hell and Kirkpatrick [18] studied the problem of partitioning the vertex set of a graph into cliques. Let *clique packing* be a subgraph whose every connected component is a clique with at least 2 vertices. Let $\rho(G)$ denote the order of the largest (in terms of the number of vertices) clique packing in G. Note that a matching is a special case of a clique packing of G. Therefore, if m is the size of maximum matching in G, we have: $2m \leq \rho(G)$.

Let G be a graph, such that $|V(G)| = \rho(G)$ and let H be its largest clique packing. A *clique partition* is a partition of $V(G)$ into vertex sets of connected components of H. In other words, every set of the clique partition induces a clique in G.

Hell and Kirkpatrick [18] presented an elegant structural theorem allowing to compute the value of $\rho(G)$. Moreover, they described graphs G having a clique partition.

Theorem 3 *The algorithm* `Solve-GLTC` *solves the τ-bounded generalized list T-coloring problem on a graph G with n vertices in time $\mathcal{O}^*\left((\tau^2+3\tau+4)^{\rho(G)/2}(\tau+2)^{n-\rho(G)}\right)$.*

Proof: Let H be the largest clique packing of G. Consider a spanning subgraph H' of H in which every connected component is isomorphic to K_2 or K_3. It can be clearly done – the vertex set of connected component of H with even number of vertices is partitioned into single edges (a perfect matching), while each component with odd number of vertices is partitioned into single edges and exactly one triangle.

Let p be a number of components isomorphic to K_2 and q be the number of components isomorphic to K_3. Clearly $2p+3q = \rho(G)$. Let $\mathcal{S} = \{S_1, .., S_r\}$ be a partition of $V(G)$, such that the sets S_i for $i \leq p$ correspond to K_2-components in H' and the sets S_i for $p \leq p+q$ correspond to K_3-components of H'. The remaining sets S_i for $i > c$ contain the remaining vertices of G, one vertex per set.

Let us consider a subgraph $G[S_i]$ for some $i \leq p+q$. Since it is a clique, each vertex has to be labeled with a different label. Therefore in any prefix of a vector from $T[k] \oplus P$, corresponding to the vertices from S_i, each element from $\{2, .., \tau+1\}$ may appear at most once. Recall that for S_i inducing an edge (i.e. for $i \leq p$) we have at most $\tau^2+3\tau+4$ possible prefixes.

Now let us consider 3-element prefix $\mathbf{a}$ corresponding to a triangle. There are:

- $2^3 = 8$ prefixes with no element from $\{2, .., \tau+1\}$,

- $3\tau \cdot 2^2 = 12 \cdot \tau$ prefixes having exactly one element from $\{2, .., \tau + 1\}$,
- $3\tau(\tau - 1) \cdot 2 = 6 \cdot \tau(\tau - 1)$ prefixes having exactly two elements from $\{2, .., \tau + 1\}$,
- $\tau(\tau - 1)(\tau - 2)$ prefixes having exactly three elements from $\{2, .., \tau + 1\}$.

For the remaining sets S_i (containing single vertices, i.e. for $i > p + q$), there are $\tau + 2$ possible prefixes of length 1. Hence the number of possible s_i-element prefixes of a vector $\mathbf{a}$ in $T[k] \oplus P$ is at most

$$f_i \leq \begin{cases} \tau^2 + 3\tau + 4 & \text{for } i \leq p \\ \tau^3 + 3\tau^2 + 8\tau + 8 & \text{for } p < i \leq p + q \\ \tau + 2 & \text{for } i > p + q. \end{cases}$$

Again, using formula (4), we obtain the following.

$$\begin{aligned} F(n) &= \mathcal{O}^*(\prod_{i=1}^{r} f_i) \\ &= \mathcal{O}^*\left((\tau^2 + 3\tau + 4)^p(\tau^3 + 9\tau^2 + 2\tau + 8)^q(\tau + 2)^{n-\rho(G)}\right) \\ &= \mathcal{O}^*\left((\tau^2 + 3\tau + 4)^{\frac{\rho(G)-3q}{2}}(\tau^3 + 3\tau^2 + 8\tau + 8)^q(\tau + 2)^{n-\rho(G)}\right) \end{aligned}$$

This expression is maximized for $q = 0$. So finally we obtain the bound:

$$F(n) = \mathcal{O}^*\left((\tau^2 + 3\tau + 4)^{\rho(G)/2}(\tau + 2)^{n-\rho(G)}\right).$$

□

The Table 1 compares the complexity bounds (more precisely, the bases of the exponential factor) of the algorithm `Solve-GLTC` applied to various graph classes for some values of τ.

τ	general graphs	subcubic graphs	claw-free graphs	regular graphs	graphs having clique partition	unit disk graphs
1	3.0000	2.8021	2.8285			2.8340
2	4.0000	3.6841	3.7417			
3	5.0000	4.6105	4.6905			
4	6.0000	5.5613	5.6569			
5	7.0000	6.5266	6.6333			

Tab. 1: Comparison of bases of the exponential factor in the complexity bound.

Acknowledgements

The authors are grateful to Łukasz Kowalik and two anonymous reviewers for their comments which helped improving the paper. We also thank Professor Zbigniew Lonc for useful advice on graph factors.

References

[1] N. Alon and A. Zaks. T-choosability in graphs. *Discrete Applied Mathematics*, 82:1 – 13, 1998.

[2] A. Amahashi and M. Kano. On factors with given components. *Discrete Mathematics*, 42:1 – 6, 1982.

[3] A. Björklund, T. Husfeldt, and M. Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39:546–563, 2009.

[4] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[5] J. M. Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32:547–556, 2004.

[6] T. Calamoneri. The $L(h, k)$-labelling problem: An updated survey and annotated bibliography. *The Computer Journal*, 54:1344–1371, 2011.

[7] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165 – 177, 1990.

[8] M. Cygan and L. Kowalik. Channel assignment via fast zeta transform. *Information Processing Letters*, 111:727–730, 2011.

[9] D. Eppstein. Small maximal independent sets and faster exact graph coloring. *Journal on Graph Algorithms and Applications*, 7:131–140, 2003.

[10] P. Erdös, A. Rubin, and H. Taylor. Choosability in graphs. *Proc. West Coast Conference on Combinatorics, Graph Theory and Computing, Arcata, Congressus Numerantium*, 26:125 – 157, 1979.

[11] J. Fiala, P. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412:2513–2523, 2011.

[12] J. Fiala, D. Král', and R. Skrekovski. A Brooks-Type Theorem for the Generalized List T-Coloring. *SIAM Journal on Discrete Mathematics*, 19:588–609, 2005.

[13] J. Fiala and R. Škrekovski. List distance labelings of graphs. *KAM Series*, 530, 2001.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[15] J. Griggs and R. K. Yeh. Labeling graphs with a condition at distance two. *SIAM Journal on Discrete Mathematics*, 5:586 – 595, 1992.

[16] W. Hale. Frequency assignment: theory and applications. *Proc. 1EEE*, 68:1497–1514, 1980.

[17] F. Havet, M. Klazar, J. Kratochvíl, D. Kratsch, and M. Liedloff. Exact algorithms for $L(2, 1)$-labelling. *Algorithmica*, 59:169–194, 2011.

[18] P. Hell and D. G. Kirkpatrick. Packings by cliques and by finite families of graphs. *Discrete Mathematics*, 49:45–59, 1984.

[19] T. R. Jensen and B. Toft. *Graph Coloring Problems*. John Wiley and Sons, 1995.

[20] K. Junosza-Szaniawski, J. Kratochvíl, M. Liedloff, P. Rossmanith, and P. Rzążewski. Fast exact algorithm for L(2,1)-labeling of graphs. *Theoretical Computer Science*, 505:42 – 54, 2013.

[21] K. "Junosza-Szaniawski and P. Rzążewski. On improved exact algorithms for $L(2,1)$-labeling of graphs. *Proc. of IWOCA 2010, Lecture Notes in Computer Science*, "6460":"34–37", "2011".

[22] K. Junosza-Szaniawski and P. Rzążewski. On the complexity of exact algorithm for $L(2,1)$-labeling of graphs. *Information Processing Letters*, 111:697 – 701, 2011.

[23] D. Král'. An exact algorithm for the channel assignment problem. *Discrete Applied Mathematics*, 145:326–331, 2005.

[24] D. Král'. The channel assignment problem with variable weights. *SIAM Journal on Discrete Mathematics*, 20:690–704, 2006.

[25] E. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5:66–67, 1976.

[26] C. McDiarmid. Discrete mathematics and radio channel assignment. *CMS Books Math. Ouvrages Math. SMC*, 11:27–63, 2003.

[27] C. McDiarmid. On the span in channel assignment problems: bounds, computing and counting. *Discrete Mathematics*, 266:387 – 397, 2003.

[28] D. Sumner. Graphs with 1-Factors. *Proceedings of the AMS*, 42:8 – 12, 1974.

[29] V. Vizing. Vertex colorings with given colors. *Metody Diskret. Analiz. (in Russian)*, 29:3 – 10, 1976.

[30] R. K. Yeh. A survey on labeling graphs with a condition at distance two. *Discrete Mathematics*, 306:1217 – 1231, 2006.