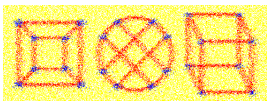
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 26, no. 2, pp. 241–255 (2022)
DOI: 10.7155/jgaa.00592

Computing $L(p,1)$-Labeling with Combined Parameters

Tesshu Hanaka[1] *Kazuma Kawai*[1] *Hirotaka Ono*[1]

[1]Department of Mathematical Informatics,
Nagoya University, Nagoya 464-8601, Japan

Submitted: March 2021 Reviewed: October 2021 Revised: December 2021

Accepted: December 2021 Final: December 2021 Published: June 2022

Article type: Regular paper

Communicated by: S.-H. Hong, S. C. Nandy, R. Uehara

Abstract. Given a graph, an $L(p,1)$-labeling of the graph is an assignment f from the vertex set to the set of nonnegative integers such that for any pair of vertices u and v, $|f(u)-f(v)|\geq p$ if u and v are adjacent, and $f(u)\neq f(v)$ if u and v are at distance 2. The $L(p,1)$-LABELING problem is to minimize the span of f (i.e.,$\max_{u\in V}(f(u))-\min_{u\in V}(f(u))+1$). It is known to be NP-hard even for graphs of maximum degree 3 or graphs with tree-width 2, whereas it is fixed-parameter tractable with respect to vertex cover number. Since the vertex cover number is a kind of the strongest parameter, there is a large gap between tractability and intractability from the viewpoint of parameterization. To fill up the gap, in this paper, we propose new fixed-parameter algorithms for $L(p,1)$-LABELING by the twin cover number plus the maximum clique size and by the tree-width plus the maximum degree. These algorithms reduce the gap in terms of several combinations of parameters.

1 Introduction

Let G be an undirected graph, and p and q be constant positive integers. An $L(p,q)$-labeling of a graph G is an assignment f from the vertex set $V(G)$ to the set of nonnegative integers such that $|f(u)-f(v)|\geq p$ if u and v are adjacent and $|f(u)-f(v)|\geq q$ if u and v are at distance 2, for all pairs of u and v in $V(G)$. We call the former *distance*-1 *condition* and the latter *distance*-2 *condition*. A k-$L(p,q)$-labeling is an $L(p,q)$-labeling $f:V(G)\to\{0,\ldots,k\}$, where the labels start from 0 for conventional reasons. The k-$L(p,q)$-LABELING problem determines whether given G

Special Issue on the 15th Int. Conference and Workshops on Algorithms and Computation, WALCOM 2021

This work is partially supported by JSPS KAKENHI Grant Numbers JP17K19960, JP17H01698, JP19K21537, JP20H05967, JP21K17707, JP21H05852, and JP21K19765. A preliminary version of this paper appeared in [25]

E-mail addresses: hanaka@nagoya-u.jp (Tesshu Hanaka) kazuma.k0622@gmail.com (Kazuma Kawai) ono@nagoya-u.jp (Hirotaka Ono)

has a k-$L(p,q)$-labeling, or not, and the $L(p,q)$-Labeling problem asks the minimum k among all possible assignments. The minimum value k is called the $L(p,q)$-labeling number, and we denote it by $\lambda_{p,q}(G)$, or simply $\lambda_{p,q}$. Notice that we can use $k+1$ different labels when $\lambda_{p,q}(G) = k$.

The original notion of $L(p,q)$-labeling can be seen in the context of frequency assignment. Suppose that vertices in a graph represent wireless devices. The presence/absence of edges indicates the presence/absence of direct communication between the devices. If two devices are very close, that is, they are connected in the graph, they need to use sufficiently different frequencies, that is, their frequencies should be apart at least p. If two devices are not very but still close, that is, they are at distance 2 in the graph, their frequencies should be apart at least q ($\leq p$). Thus, the setting of $q = 1$ as one unit and $p \geq q = 1$ is considered natural and interesting, and the minimization of used range becomes the issue. Note that $L(1,1)$-labeling on G is equivalent to the ordinary coloring on the square of G. From these, $L(p,1)$-Labeling for $p > 1$ is intensively and extensively studied among several possible settings of p. In particular, $L(2,1)$-Labeling is considered the most important. A reason is that it is natural and suitable as a basic step to consider, and another reason is that the computational complexity (e.g., hardness or polynomial-time solvability) tends to be inherited from $L(2,1)$ to $L(p,1)$ of $p > 2$; for example, if $L(2,1)$-Labeling is NP-hard in a setting, the hardness proof could be modified to $L(p,1)$-Labeling in the same setting. Also many polynomial-time algorithms of $L(2,1)$-labeling for specific graph classes can be easily extended to $L(p,1)$. We can find various related results in surveys by Calamoneri [6]. See also [27] for algorithmic results.

The notion of $L(p,q)$-Labeling firstly appeared in [23] and [32]. Griggs and Yeh formally introduced the $L(2,1)$-Labeling problem [21]. They also show that $L(2,1)$-Labeling is NP-hard in general. Furthermore, $L(2,1)$-Labeling is shown to be NP-hard even for planar graphs, bipartite graphs, chordal graphs [4], graphs with diameter of 2 [21] and graphs with tree-width 2 [14]. Moreover, for every $k \geq 4$, k-$L(2,1)$-Labeling, that is the decision version of $L(2,1)$-Labeling is NP-complete for general graphs [17] and even for planar graphs [11]. These results imply that k-$L(2,1)$-Labeling is NP-complete for every $\Delta \geq 3$, where Δ denotes the maximum degree. On the other hand, $L(2,1)$-Labeling can be solved in polynomial time for paths, cycles, wheels [21], but these are rather trivial. For non-trivial graph classes, only a few graph classes (e.g., co-graphs [7]) are known to be solvable in polynomial time. In particular, Griggs and Yeh conjectured that $L(2,1)$-Labeling on trees was NP-hard, which was later disproved (under P$\neq$NP) by the existence of an $O(n^{5.5})$-time algorithm [7]. It is now known that $L(p,1)$-Labeling on trees can be solved in linear time [26].

From these results, we roughly understand the boundary between polynomial time solvability and NP-hardness concerning graph classes, and studies are going to fixed-parameter (in)tractability. For a problem A with input size n and parameter t, A is called *fixed-parameter tractable* with respect to t if there is an algorithm whose running time is $g(t)n^{O(1)}$, where g is a computable function. Such an algorithm is called a *fixed-parameter* algorithm. Moreover, A is called *slice-wise polynomial (XP)* with respect to t if there is an algorithm whose running time is $g(t)n^{h(t)}$ where g, h are computable functions, and such an algorithm is called an *XP* algorithm. If problem A is NP-hard for a constant value of t, there is neighter fixed-parameter algorithm nor XP algorithm unless P=NP; we say A is paraNP-hard. Unfortunately, $L(2,1)$-Labeling is already shown to be paraNP-hard for several parameters such as $\lambda_{2,1}$, maximum degree and tree-width as seen above. For positive results, it is fixed-parameter tractable with respect to vertex cover number [16] or neighborhood diversity [13]. Note that vertex cover number is a stronger parameter than tree-width, which means that if the vertex cover number is bounded, the tree-width is also. There is still a gap on fixed-parameter (in)tractability between them. For such a situation, two approaches can be taken. One is to

finely classify intermediate parameters and see fixed-parameter (in)tractability for them, and the other is to combine two or more parameters and see fixed-parameter (in)tractability under the combinations. In this paper, we take the latter approach.

1.1 Our contribution

In this paper, we present algorithms with combined parameters. The parameters that we focus on are clique-width ($\mathtt{cw}$), tree-width ($\mathtt{tw}$), maximum clique size (ω), maximum degree (Δ) and twin cover number ($\mathtt{tc}$). These are selected in connection with aforementioned parameters, $\lambda_{p,1}$, maximum degree and tree-width. Maximum clique size and clique-width are well used parameters weaker than tree-width. Maximum degree itself is a considered parameter, which is strongly related to $\lambda_{p,q}(G)$. In fact, it is easy to see that $\lambda_{p,1} \geq \Delta + p - 1$, and $\lambda_{p,1} \leq \Delta^2 + (p-1)\Delta - 2$ [20]. Thus, $\lambda_{p,1}$ and Δ are parameters equivalent in terms of fixed-parameter (in)tractability. Twin cover number is picked up as a parameter that is moderately weaker than vertex cover number but stronger than clique-width and is also incomparable to neighborhood diversity.

These parameters are ordered in the following two ways: (1) $(\mathtt{vc} \succeq)\{\mathtt{tw}, \mathtt{tc}\} \succeq \mathtt{cw}$ and (2) $(\lambda_{p,1} \simeq)\Delta \succeq \omega$. Here, for graph parameters α and β, $\alpha \succeq \beta$ represents that there is a positive function g such that $g(\alpha(G)) \geq \beta(G)$ holds for any G, and we denote $\alpha \simeq \beta$ if $\alpha \succeq \beta$ and $\beta \succeq \alpha$. For combined parameters of one from (1) and another from (2), we design fixed-parameter algorithms. Note that some combination yields essentially one parameter. For example, $\mathtt{tw} + \omega$ is equivalent to $\mathtt{tw}$, because $\mathtt{tw} \geq \omega - 1$ holds. The obtained results are listed below:

- $L(p,1)$-Labeling can be solved in time $\Delta^{O(\mathtt{tw}\Delta)}n$ for $p \geq 1$. Since it is known that $\mathtt{tw} \leq 3\mathtt{cw}\Delta - 1$ ([22]), it is also a $\Delta^{O(\mathtt{cw}\Delta^2)}n$-time algorithm, which implies $L(p,1)$-Labeling is actually FPT with respect to $\mathtt{cw} + \Delta$. This result also implies that $L(p,1)$-Labeling is FPT when parameterized by band-width.
- $L(p,1)$-Labeling is FPT when parameterized by $\mathtt{tc} + \omega$. Since $\mathtt{tc} + \omega \leq \mathtt{vc} + 1$ for any graph, it generalizes the fixed-parameter tractability with respect to vertex cover number in [16]. Since $\mathtt{tc} + \omega \geq \mathtt{tw}$, $\mathtt{tc} + \omega$ is located between $\mathtt{tw}$ and $\mathtt{vc}$.
- $L(1,1)$-Labeling is FPT when parameterized by *only* twin cover number. This also yields a fixed-parameter p-approximation algorithm for $L(p,1)$-Labeling with respect to twin cover number.

Figure 1 illustrates the detailed relationship between graph parameters and the parameterized complexity of $L(p,1)$-Labeling, which includes our new results and previous results shown in the next subsection.

1.2 Related work

As mentioned above, $L(p,1)$-Labeling is NP-hard even on graphs of tree-width 2 [14]. Using stronger parameters than tree-width, Fiala et al. showed that $L(p,1)$-Labeling is fixed-parameter tractable when parameterized by vertex cover [16] and neighborhood diversity [13]. Moreover, Fiala, Kloks and Kratochvíl showed that the problem is XP when parameterized by feedback edge set number [17]. For approximation, it is NP-hard to approximate $L(p,1)$-Labeling within a factor of $n^{0.5-\varepsilon}$ for any $\varepsilon > 0$, whereas it can be approximated within $O(n(\log\log n)^2/\log^3 n)$ [24]. For $L(1,1)$-Labeling, it can be solved in time $O(\Delta^{2^{8(\mathtt{tw}+1)+1}}n + n^3)$, and hence it is XP by tree-width [34]. This result is tight in the sense of fixed-parameter (in)tractability, because it is

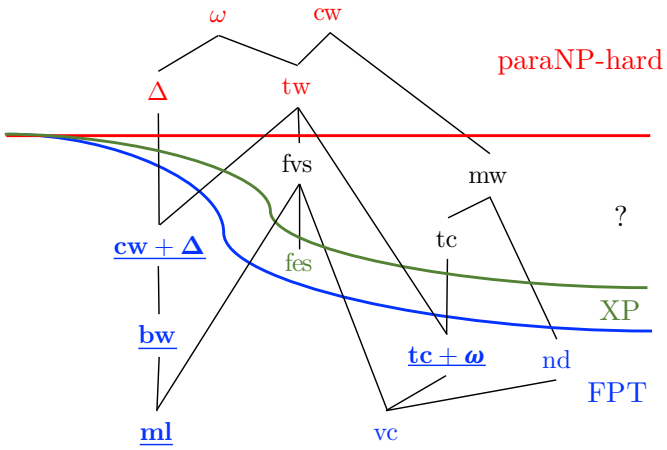

Figure 1: The relationship between graph parameters and the parameterized complexity of $L(p,1)$-Labeling. Let $\omega, \Delta, \mathtt{cw}, \mathtt{mw}, \mathtt{nd}, \mathtt{tc}, \mathtt{tw}, \mathtt{fvs}, \mathtt{fes}, \mathtt{bw}, \mathtt{ml}$, and $\mathtt{vc}$ denote maximum clique size, maximum degree, clique-width, modular-width, neighborhood diversity, twin cover number, tree-width, feedback vertex set number, feedback edge set number, band-width, max leaf number, and vertex cover number, respectively. Connections between two parameters imply that the upper is bounded by a function of the lower. The underlines for parameters indicate that they are obtained in this paper.

W[1]-hard with respect to tree-width [16]. Moreover, $L(1,1)$-Labeling can be solved in time $O(\mathtt{cw}^3 2^{6\mathtt{cw}} n^{2^{4\mathtt{cw}}+2^{2\mathtt{cw}}+1})$ [33].

Apart from $L(p,1)$-Labeling, twin cover number is a relatively new graph parameter, which is introduced in [18] as a weaker parameter than vertex cover number. In the same paper, many problems are shown to be FPT when parameterized by twin cover number, and it is getting to be a standard parameter (e.g., [1, 12, 19, 28, 29]). Recently, for Imbalance, which is one of graph layout problems, a parameterized algorithm is presented [31]. It is interesting that they also adopt twin cover number plus maximum clique size as the parameters.

2 Preliminaries

In this paper, we use the standard graph notations. Suppose that $G = (V, E)$ is a simple and connected graph with the vertex set V and the edge set E. We sometimes use $V(G)$ or $E(G)$ instead of V or E respectively, to specify graph G. For $G = (V, E)$, we denote the numbers of vertices and edges by $n = |V|$ and $m = |E|$, respectively. For $V' \subseteq V$, we denote by $G[V']$ the subgraph of G induced by V'. For two vertices u and v, the *distance* $\mathrm{dist}_G(u, v)$ is defined by the length of a shortest path between u and v where the length of a path is the number of edges of it. We denote the closed neighbourhood and the open neighbourhood of a vertex v by $N_G[v]$ and $N_G(v)$, respectively. We also define $N_G^{\leq \ell}[v] = \{u \mid \mathrm{dist}_G(u, v) \leq \ell\}$ and $N_G^{\leq \ell}(v) = N_G^{\leq \ell}[v] \setminus \{v\}$. For a set $S \subseteq V$, let $N_G(S) = \bigcup_{v \in S} N_G(v)$ and $N_G[S] = \bigcup_{v \in S} N_G[v]$. The degree of v is denoted by $d_G(v) = |N_G(v)|$. The maximum degree of G is denoted by $\Delta(G)$. For simplicity, we sometimes omit the subscript G.

The k-th power $G^k = (V, E^k)$ of a graph $G = (V, E)$ is a graph such that the set of vertices is V and there is an edge (u, v) in E^k if and only if there is a path of length at most k between u and v in G [5]. In particular, G^2 is called the *square* of G.

Graph parameters

In the following, we introduce several graph parameters.

Clique-width We introduce the definition of clique-width (see also [8, 9]). A vertex-labeled graph is a graph whose vertices have exactly one integer as a label. For a positive integer c, a *c-graph* is a vertex-labeled graph with labels in $\{1, 2, \ldots, c\}$. Then the *clique-width* $\mathtt{cw}(G)$ of G is defined as the minimum integer c such that G is constructed with c labels by the following operations:

- Create a new vertex with label $i \in \{1, 2, \ldots, c\}$,
- Take a disjoint union of two c-graphs,
- For two labels i and j, connect every pair of a vertex labeled by i and a vertex labeled by j by an edge, and
- Relabel all the labels of vertices with label i to label j.

Tree-width

Definition 1 (Tree decomposition) *A* tree decomposition *of a graph $G = (V, E)$ is defined as a pair $\langle \mathcal{X}, T \rangle$, where T is a tree with node set $I(T)$ and $\mathcal{X} = \{X_i \mid i \in I(T)\}$ is a collection of subsets, called* bags*, of V such that:*

1. (vertex condition) $\bigcup_{i \in I(T)} X_i = V$,
2. (edge condition) *For every $\{u, v\} \in E$, there exists an $i \in I(T)$ such that $\{u, v\} \subseteq X_i$, and*
3. (coherence property) *For every $u \in V$, $I_u = \{i \in I(T) \mid u \in X_i\}$ induces a connected subtree of T.*

The width *of a tree decomposition is defined as* $\max_{i \in I} |X_i| - 1$ *and the* tree-width *of G, denoted by $\mathtt{tw}(G)$, is defined as the minimum width among all possible tree decompositions of G.*

Definition 2 (Nice tree decomposition) *A tree decomposition $\langle \mathcal{X}, T \rangle$ is* nice *if it satisfies the following conditions:*

1. *T has a* root node $r(T) \in I$ that satisfies $X_{r(T)} = \emptyset$.

2. Each node of T has at most two children.

3. Each node i in T is one of the following five types:

- A *leaf* node i has no children and $X_i = \emptyset$,
- An *introduce vertex v* node i has exactly one child j satisfying $X_i = X_j \cup \{v\}$ for a vertex $v \in V$,

- An *introduce edge* $\{u,v\}$ node i has exactly one child j satisfying $X_i = X_j$ and it is labeled with an edge $\{u,v\} \in E$ where $u,v \in X_i$,
- A *forget* v node i has exactly one child j satisfying $X_i = X_j \setminus \{v\}$ for a vertex $v \in V$, and
- A *join* node i has exactly two children j_1, j_2 such that $X_{j_1} = X_i$ and $X_{j_2} = X_i$.

We additionally require that every edge in E is introduced exactly once.

By the last statement, every edge is assigned to exactly one node. An assignment is done by an introduce edge node, for a pair of vertices that have already been introduced. This implies that for an introduce vertex v node i, v is an isolated vertex in G_i, where $G_i = (V_i, E_i)$ is defined by V_i, the union of all bags X_j such that $j = i$ or j is a descendant of i, and $E_i \subseteq E$, the set of all edges introduced at i (if i is an introduce edge node) or a descendant of i.

Any tree decomposition with ℓ nodes can be transformed to a nice tree decomposition with $O(\mathtt{tw} \cdot n)$ bags and the same width in time $O(\mathtt{tw}^2 \cdot \max\{\ell, n\})$ [10].

Twin cover Two vertices u, v are called *twins* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. Moreover, if twins u, v have edge $\{u,v\}$, they are called *true twins* and the edge is called a *twin edge*. Then a *twin cover* of G is defined as follows.

Definition 3 (Twin cover, [18]) *A set of vertices X is a* twin cover *of G if every edge $\{u,v\} \in E$ satisfies either (1) $u \in X$ or $v \in X$, or (2) u, v are true twins. The* twin cover number *of G, denoted by $\mathtt{tc}(G)$, is defined as the minimum size of twin covers in G.*

An important observation is that the complement $V \setminus X$ of a twin cover X induces disjoint cliques. Moreover, for each clique Z of $G[V \setminus X]$, $N(u) \cap X = N(v) \cap X$ for every $u, v \in Z$ [18]. That is, any pair of vertices in a clique are true twins and each edge in $G[V \setminus X]$ is a twin edge.

A *vertex cover* X is the set of vertices such that for every edge, at least one endpoint is in X. The *vertex cover number* of G, denoted by $\mathtt{vc}(G)$, is defined as the minimum size of vertex covers in G. Since every vertex cover of G is also a twin cover of G, $\mathtt{tc}(G) \leq \mathtt{vc}(G)$ holds. Because $\omega(G) \leq \mathtt{vc}(G) + 1$, we have $\mathtt{tc}(G) + \omega(G) \leq 2\mathtt{vc}(G) + 1$.

Other graph parameters

Definition 4 (Band-width) *For a graph $G = (V,E)$, the* band-width $\mathtt{bw}(f)$ *of a map $f : V \to [1,n]$ is defined by $\max_{(i,j)\in E} |f(i) - f(j)|$. The* band-width $\mathtt{bw}(G)$ *of G is the minimum value of $\mathtt{bw}(f)$ among all possible f, that is, $\mathtt{bw}(G) = \min_{f:V\to[1,n]} \mathtt{bw}(f)$.*

Definition 5 (Max leaf number) *For a graph $G = (V,E)$, the* max leaf number $\mathtt{ml}(G)$ *of G is defined as the maximum number of leaves among all possible spanning trees of G.*

Integer Linear Programming

Integer Linear Programming Feasibility is formulated as follows.

Input: An $q \times p$ matrix A with integer elements, an integer vector $b \in \mathbb{Z}^q$

Question: Is there a vector $x \in \mathbb{Z}^p$ such that $A \cdot x \leq b$?

Lenstra [30] proved that Integer Linear Programming Feasibility is FPT when parameterized by the number of variables.

3 Parameterization by $\mathtt{cw}+\Delta$ and $\mathtt{tw}+\Delta$

As $L(p,1)$-Labeling is paraNP-hard for tree-width, so is for clique-width. In this section, as a complement, we show that $L(p,1)$-Labeling (actually, $L(p,q)$-Labeling for any constant p and q) is fixed-parameter tractable when parameterized by $\mathtt{cw}+\Delta$.

To this end, we give a fixed-parameter algorithm for $L(p,1)$-Labeling parameterized by not $\mathtt{cw}+\Delta$ but $\mathtt{tw}+\Delta$, which actually implies that the problem is FPT with respect to $\mathtt{cw}+\Delta$, because it is known that $\mathtt{tw} \leq 3\mathtt{cw}\Delta - 1$ [22]. The running time of the algorithm is $\Delta^{O(\mathtt{tw}\Delta)}n$, and so it is $\Delta^{O(\mathtt{cw}\Delta^2)}n$.

In the algorithm, we first construct the square G^2 of G and then compute $L(p,1)$-Labeling of G by dynamic programming on a tree decomposition $\langle \mathcal{X}', T' \rangle$ of G^2. Actually, the algorithm runs for $L(p,q)$-Labeling though the running time depends on $\lambda_{p,q}(G)$. One can obtain the square of G^2 in time $O(m\Delta(G)) = O(\Delta(G)^2 n)$. We then prove the following lemma.

Lemma 1 *Given a tree decomposition of a graph G of width t with ℓ bags, one can construct a tree decomposition of G^2 of width at most $(t+1)\Delta(G)+t$ with ℓ bags in time $O(t\Delta(G)\ell)$.*

Proof: We are given a tree decomposition $\langle \mathcal{X}, T \rangle$ of G of width t. Let $X'_i = X_i \cup N(X_i)$ and $\mathcal{X}' = \{X'_i \mid i \in I(T)\}$ be the set of bags. We here define $\langle \mathcal{X}', T' \rangle$ as a tree decomposition of G^2, where T' and T are identical; T and T' has the same node set and the same structure, where each $i \in I(T')$ corresponds to $i \in I(T)$. In the following, we denote $\langle \mathcal{X}', T \rangle$ instead of $\langle \mathcal{X}', T' \rangle$.

We can see that $\langle \mathcal{X}', T \rangle$ is really a tree decomposition of G^2 with width $(t+1)\Delta(G)+t$. It satisfies the properties of tree decomposition indeed: Since $\bigcup_{i \in I(T)} X'_i = \bigcup_{i \in I(T)} (X_i \cup N(X_i)) = V(G) = V(G^2)$, the vertex condition is satisfied. We next see the edge condition. For each $e \in E$, there is X_i containing e, so $e \in X'_i$. For each $\{u,v\} \in E^2 \setminus E$, there is a vertex $v' (\neq u, v)$ such that $\{u,v'\} \in E$ and $\{v',v\} \in E$. Thus there is X_i satisfying $\{u,v'\} \subseteq X_i$, which implies $\{u,v\} \subseteq X_i \cup \{v\} \subseteq X_i \cup N(\{v'\}) \subseteq X'_i$. These show that the edge condition is satisfied.

Finally, we check coherence property: we show that for every $u \in V$, $I'_u = \{i \in I(T) \mid u \in X'_i\}$ induces a connected subtree of T. Note that

$$I'_u = \{i \in I(T) \mid u \in X'_i\} = \{i \in I(T) \mid u \in X_i\} \cup \bigcup_{v \in N(u)} \{i \in I(T) \mid v \in X_i\}.$$

Here, the subgraph T_u of T induced by $\{i \in I(T) \mid u \in X_i\}$ is connected by the coherent property of $\langle \mathcal{X}, T \rangle$. Also for each $v \in N(u)$, the subgraph T_v of T induced by $\{i \in I(T) \mid v \in X_i\}$ is connected. By $\{u,v\} \in E$, the edge condition of $\langle \mathcal{X}, T \rangle$ implies that there exists a bag X_j containing both u and v. Since T_u and T_v has a common node j, the subgraph of T induced by $\{i \in I(T) \mid u \in X_i\} \cup \{i \in I(T) \mid v \in X_i\}$ is also connected, which leads that the subgraph of T induced by I'_u is also connected.

Hence, $\langle \mathcal{X}', T \rangle$ is a tree decomposition of G^2. Since the size of bag X'_i is $|X'_i| = |X_i \cup N(X_i)| = |\bigcup_{u \in X_i} N[u]| \leq (t+1)(\Delta(G)+1)$, the width is at most $(t+1)(\Delta(G)+1) - 1 = (t+1)\Delta(G) + t$. The construction of $\langle \mathcal{X}', T \rangle$ is done by preparing each X'_i, which takes $O(t\Delta(G))$ steps for each i. Thus it can be done in time $O(t\Delta(G)\ell)$ in total. □

Corollary 1 $\mathtt{tw}(G^2) \leq (\mathtt{tw}(G)+1)\Delta(G) + \mathtt{tw}(G)$ *holds.*

By the above lemma, the tree-width of G^2 is bounded if $\mathtt{tw}(G)$ and $\Delta(G)$ are bounded. Thus we can design a dynamic programming algorithm on a tree decomposition of G^2.

Lemma 2 *Given a tree decomposition of G^2 of width at most t, one can compute k-$L(p,q)$-*Labeling *on G in time $O((k+1)^{t+1}t^2n)$.*

Proof: We present a dynamic programming algorithm for $L(p,q)$-Labeling of G on a nice tree decomposition of G^2, which is almost the ordinary nice tree decomposition except that each introduce edge node has an extra one bit information that represents whether $e \in E$ or not. Here, we assume that an introduce edge e node has additional information whether $e \in E$ or not. This can be done by letting each introduce edge node have 1 bit when we make a nice tree decomposition.

In the algorithm, we guess every assignment of labels for vertices in each bags. For each bag X_i and each labeling $f_i : X_i \to \{0,\ldots,k\}$, we define $L[i,f_i]$, which is `true` if there is a partial k-$L(p,q)$-labeling in G_i such that the labels of vertices in X_i follows f_i, and `false` otherwise. In the root node r, if $L[r,f_r] =$ `true` for some f_r, there exists a k-$L(p,q)$-labeling of G. The algorithm computes $L[i,f_i]$'s from leaves to the root by the bottom-up manner.

Leaf node In a leaf node $X_i = \{v\}$, we set $L[i,f_i(v)] =$ `true` for $\forall f_i(v) \in \{0,\ldots,k\}$.

Introduce vertex v node: For an introduce vertex node i having a child j, $X_i = X_j \cup \{v\}$. Since v is isolated in G_i, we define $L[i,f_i] = L[j, f_i \setminus f_i(v)]$.

Introduce edge $\{u,v\}$ node: In an introduce edge node i, suppose that $e = \{u,v\}$ is introduced. If $e \in E(G)$, $L[i,f_i] =$ `true` if and only if $L[j,f_j] =$ `true` in node j such that $f_j(w) = f_i(w)$ for every vertex $w \in X_i (= X_j)$ and $|f_j(u) - f_j(v)| \geq p$. Otherwise, $e \in E(G^2) \setminus E(G)$. This implies that the distance between u and v is 2 in G by the definition of G^2. Therefore, we define $L[i,f_i] =$ `true` if and only if $L[j,f_j] =$ `true` in node j such that $f_j(w) = f_i(w)$ for every vertex $w \in X_i (= X_j)$ and $|f_j(u) - f_j(v)| \geq q$.

Forget v node: In a forget node i, we have $X_i = X_j \setminus \{v\}$. By the definition of a tree decomposition, v never appears in any later nodes in the tree decomposition of G^2. Thus, we can compute $L[i,f_i] = \bigvee_{f_j \setminus f_j(v) = f_i} L[j,f_j]$.

Join node: In a join node i having two children j_1, j_2, $X_i = X_{j_1} = X_{j_2}$ holds. Thus, for each labeling f_i, we can compute $L[j,f_i] = L[j_1,f_i] \wedge L[j_2,f_i]$.

The correctness of the dynamic programming algorithm is clear. Then we analyze the running time. In the algorithm, the size of each DP table in a node is at most $(k+1)^{t+1}$. The update time of each entry in a DP table is bounded by $O(t)$. Because the number of nodes of a tree decomposition is bounded by $O(tn)$ [10], the running time of the dynamic programming is $O((k+1)^{t+1}t^2n)$. □

Here, one can construct a tree decomposition $\langle \mathcal{X}, T\rangle$ of G of width $5\mathtt{tw}(G)+4$ with $O(n)$ bags in time $2^{O(\mathtt{tw}(G))}n$ [3]. By Lemma 1, we can obtain a tree decomposition $\langle \mathcal{X}', T\rangle$ of G^2 of width $(5\mathtt{tw}(G)+4+1)\Delta(G) + 5\mathtt{tw}(G) + 4 = O(\mathtt{tw}(G)\Delta(G))$ from $\langle \mathcal{X}, T\rangle$ in time $O(\mathtt{tw}(G)\Delta(G)n)$. By Lemma 2 and $\lambda_{p,q} \leq \max\{p,q\}\Delta^2$, we have the following theorem.

Theorem 1 *For any positive constant p and q, there is an algorithm to solve $L(p,q)$-*Labeling *in time $\Delta^{O(\mathtt{tw}\Delta)}n$, which is also bounded by $\Delta^{O(\mathtt{cw}\Delta^2)}n$.*

Note that $\mathtt{tw} \leq 3\mathtt{cw}\Delta - 1$ holds for any graph [22].

For the band-width $\mathtt{bw}(G)$ and the max leaf number $\mathtt{ml}(G)$ of G, we have $\mathtt{tw}(G) \leq \mathtt{bw}(G) \leq \mathtt{ml}(G)$ and $\Delta(G) \leq 2\mathtt{bw}(G)$ [2]. Thus, the following corollary holds.

Corollary 2 *For any positive constant p and q, $L(p,q)$-Labeling is fixed-parameter tractable when parameterized by max leaf number, and even band-width.*

4 Parameterization by twin cover number

4.1 $L(p,1)$-Labeling parameterized by $\mathtt{tc}+\omega$

We design a fixed-parameter algorithm for $L(p,1)$-Labeling with respect to $\mathtt{tc}+\omega$. Notice that for a twin cover X of $G=(V,E)$, each of the connected components of $G[V\setminus X]$ forms a clique. We categorize vertices in $V\setminus X$ with respect to the neighbors in X. Let $T_1,T_2,\ldots,T_s$ be the sets of vertices having common neighbors in X, called *types* of vertices in $V\setminus X$, where s is the number of types. Moreover, we say that a clique $C\subseteq V\setminus X$ is of type T_i if $C\subseteq T_i$. Note that $V\setminus X=\bigcup_{i=1}^{s}T_i$. Let $n_i=|T_i|$ and ω_i be the maximum clique size in T_i.

We first see a general property about cliques with the common neighbors: Suppose that a graph G consists of only cliques $C_1,C_2,\ldots,C_h$ and the common neighbors Y of all the vertices in the cliques. That is, all the vertices are within distance 2. Note that a twin cover focuses on such a substructure in a graph. Then the following lemma holds.

Lemma 3 *Suppose that a graph G is above defined by cliques $C_1,C_2,\ldots,C_h$, in the descending order of the size and their common neighbors Y, where the vertices in Y are labeled by $a_1,a_2,\ldots,a_{|Y|}$. For an arbitrary set L of labels that are at least p apart from $a_1,a_2,\ldots,a_{|Y|}$, if $|L|\geq\sum_j|C_j|$ and $\sum_j|C_j|\geq p|C_1|$ hold, there exists an $L(p,1)$-labeling of $C_1,\ldots,C_h$ using only labels in L.*

Proof: Let $n'=\sum_j|C_j|$ and $\omega=|C_1|$. Let us assume $L=\{l_1,l_2,\ldots,l_{n'}\}$. Since we can use distinct labels for vertices in $C_1,C_2,\ldots,C_h$, only the distance-1 condition inside of a same clique matters. If $n'\equiv 1 \pmod p$, we label the vertices in $C_1,C_2,\ldots,C_{n'}$ in this order by using labels in order of $l_1,l_{p+1},l_{2p+1},\ldots,l_{n'},l_2,l_{p+2},l_{2p+2}\ldots,\ l_{n'-p+2},l_3\ldots,l_p,l_{2p},\ldots,l_{n'-1}$. Note that the vertices in C_1 are labeled by $l_1,l_{p+1},\ \ldots,\ l_{p(\omega-1)+1}$ (note that $p\omega\leq n'$). Since the difference between $l_{\alpha p+i}$ and $l_{(\alpha+1)p+i}$ for each i and α is at least p, the labeling for cliques does not violate the distance-1 condition. We can choose similar orderings for the other residuals. □

Now we go back to the algorithm parameterized by $\mathtt{tc}+\omega$. Given a twin cover X, we say that a k-$L(p,1)$-labeling is *good* for X if it uses only labels in $\{0,1,\ldots,(2p-1)|X|-p\}\cup\{k-(2p-1)|X|+p,\ldots,k\}$ for X. The following lemma is also important. It can be shown by repeatedly applying Lemma 3.

Lemma 4 *Let X be a twin cover in G such that each T_i satisfies $\omega_i\leq n_i/p$. If G has a k-$L(p,1)$-labeling, then G also has a good k-$L(p,1)$-labeling for X.*

Proof: Let f be an $L(p,1)$-labeling, and $a,b\in\{0,1,\ldots,k\}$ be two labels such that (1) they are not used in X, (2) they are at least p apart from all the labels used in X, and (3) there is at least one label l used in X where $a+p\leq l\leq b-p$. By the definition of a and b, l can exist only when $b-a\geq 2p$. For example, the triplet of (a,l,b) is possible for $l=a+p$ and $b=a+2p$, but we cannot take l for $b<a+2p$. Then we rotate labels between a and b in f as follows: $a\to a+1$, $a+1\to a+2$, …, $b\to a$.

Let f' be a labeling obtained by the above relabeling. The rotation does not affect the distance-2 condition, though it may affect distance-1 condition. As for X, we notice that only labels in $\{a+p,\ldots,b-p\}$ are changed in X, which does not yield any new conflict inside of X. Therefore, f' satisfies the distance-1 condition of $L(p,1)$-labeling in $G[X]$. Also $b-p+1$ is only a label that

could be newly used in X of f', which does not affect any label in $V \setminus X$; f' also satisfies the distance-1 condition of $L(p,1)$-labeling between X and $V \setminus X$.

We see that f' does not violate the condition of $L(p,1)$-labeling within X and between X and $V \setminus X$. On the other hand, it may violate the condition within $V \setminus X$. For example, if a clique in $G[V \setminus X]$ has two vertices labeled with $b-p+1$ and $b+1$ in f, they are labeled with $b-p+2$ and $b+1$ in f', which violates the distance-1 condition by $(b+1)-(b-p+2) = p-1$. Fortunately, such a violation can be easily avoided by further relabeling vertices in $V \setminus X$ as follows.

For each T_i, we first observe that labels used in f for T_i are different from each other due to the distance 2-condition, as so in f'. A problem may occur inside of a clique, which may violate the distance-1 condition. However, even if a conflict occurs, we can obtain a proper k-$L(p,1)$-labeling by relabeling the vertices in T_i with the same label set. This is because the cliques inside of T_i have exactly same neighbors and $p\omega_i \leq n_i$ holds, by which we can apply the argument of Lemma 3.

The above procedure can push up a label in a middle range used in X. It can be applied as long as a triplet of a, b and l exists. As mentioned above, l can exist only when $b-a \geq 2p$. Consider the labeling where all the vertices in X are labeled by $|X|$ labels near k: $k-(2p-1)|X|+p, k-(2p-1)(|X|-1)+p, k-(2p-1)(|X|-2)+p, \ldots, k-(2p-1)+p$ $(= k-p+1)$. It is easy to see that we cannot take a and b for the labeling, though we can take a and b if we use $k-(2p-1)|X|+p-1$ or a smaller label instead of $k-(2p-1)|X|+p$. On the other hand, consider the labeling where all the vertices in X are labeled by $|X|$ labels near 0: $p-1, 3p-2, \ldots, (p-1)+(2p-1)(|X|-2), p-1+(2p-1)(|X|-1)(= (2p-1)|X|-p)$. We cannot take a and b again.

By these, if we cannot apply the above procedure, all the labels for X are in $\{0,1,\ldots,(2p-1)|X|-p-1, (2p-1)|X|-p\} \cup \{k-(2p-1)|X|+p, k-(2p-1)|X|+p+1, \ldots, k-1, k\}$. Hence, by applying the above procedure repeatedly, we eventually obtain a good k-$L(p,1)$-labeling f^*, which implies that if there is a k-$L(p,1)$-labeling in G, then there is a good k-$L(p,1)$-labeling for X in G. □

Thus, we consider to find a good $L(p,1)$-labeling. Using the lemma, we show that $L(p,1)$-Labeling is fixed-parameter tractable with respect to $\mathtt{tc}+\omega$.

Theorem 2 *$L(p,1)$-Labeling is fixed-parameter tractable when parameterized by $\mathtt{tc}+\omega$.*

Proof: We present an algorithm to solve k-$L(p,1)$-Labeling instead of $L(p,1)$-Labeling. We first compute a minimum twin cover X in time $O(1.2738^{\mathtt{tc}} + \mathtt{tc}n + m)$ [18]. For twin cover X, we define T_i's. Then, we define another twin cover of $X' = X \cup \bigcup_{i:\omega_i > n_i/p} T_i$, where X' is obtained by adding every T_i breaking the condition of Lemma 4 to X. By this modification, our algorithm can utilize a twin cover that satisfies the condition of Lemma 4. The size of X' is at most $\mathtt{tc}+2^{\mathtt{tc}}\cdot p\cdot\omega$, because the number of types is at most $2^{\mathtt{tc}}$ and the size of T_i joining X is at most $p\cdot\omega$. Let $\mathtt{tc}' = |X'|$.

We are now ready to present the core of the algorithm. We classify an instance into two cases. If k is small enough, we can apply a brute-force type algorithm. Otherwise, we try to find a good k-$L(p,1)$-labeling.

(**Case:** $k < 4p\mathtt{tc}'$) For each type T_i, the distance between two vertices in T_i is at most 2. Thus, the labels of vertices in T_i must be different each other. Due to $k < 4p\mathtt{tc}'$, if $|T_i| \geq 4p\mathtt{tc}'$, we conclude that the input is a no-instance. Otherwise, $n = |X'| + \sum_i |T_i| \leq \mathtt{tc}' + 4p\mathtt{tc}'2^{\mathtt{tc}}$ holds, because the number of T_i's is at most $2^{\mathtt{tc}}$. Thus we check all the possible labelings in time $O((4p\mathtt{tc}')^{\mathtt{tc}'(4p2^{\mathtt{tc}}+1)})$.

(**Case:** $k \geq 4p\mathtt{tc}'$) Let $\mathcal{C}_0, \mathcal{C}_1, \ldots \mathcal{C}_t$ be the family of all possible set systems on $\{T_1, \ldots, T_s\}$ such that whenever two distinct T_j and $T_{j'}$ are in $\mathcal{C}_i$ then $N(T_j) \cap N(T_{j'}) = \emptyset$. Here, $\mathcal{C}_0$ is the empty set. Roughly speaking, each of $\mathcal{C}_i$ represents a set of types that can use an identical label, because the distance between T_j and $T_{j'}$ in $\mathcal{C}_i$ is at least 3 and they do not violate the distance-2 condition. The idea to reduce the number of essential patterns of assignments is to consider not assignments of labels to vertices in $V \setminus X'$ but $\mathcal{C}_i$ instead. Note that cliques in a type T_j can be labeled by the same way due to Lemma 3.

For each $\mathcal{C}_i$, we prepare a set L_i of labels, which will be used during the execution of the algorithm to represent the set of labels that could be used for vertices in $T_j \in \mathcal{C}_i$. Note that $L_0, L_1, \ldots, L_t$ must be disjoint each other, and a label in L_i is used exactly once per T_j. We also define L_0 as the set of labels not used in $V \setminus X'$. Each L_i can be empty.

By Lemma 4, there is a good k-$L(p,1)$-labeling for X' such that vertices in X' only use labels in $\{0, 1, \ldots, (2p-1)|X'|-p\} \cup \{k-(2p-1)|X'|+p, \ldots, k\}$ if the input is an yes-instance. Thus we try all the possible partial labelings for X', each of which uses only labels in $\{0, 1, \ldots, (2p-1)|X'|-p\} \cup \{k-(2p-1)|X'|+p, \ldots, k\}$. Since the number of labels used in X' is at most $2((2p-1)|X'|-p+1) \leq 4p\mathtt{tc}'$, there are at most $(4p\mathtt{tc}')^{\mathtt{tc}'}$ possible labelings of X'. For each of them we further try all the possible placement of labels in $\{0, 1, \ldots, (2p-1)|X'|-1\} \cup \{k-(2p-1)|X'|+1, \ldots, k\}$ into $L_0, L_1, \ldots, L_t$, which is a little wider than above. The number of possible placements is at most $(t+1)^{4p\mathtt{tc}'}$ due to the disjointness of L_i's. Therefore, the total possible nonisomorphic partial labelings is at most $(4p\mathtt{tc}')^{\mathtt{tc}} \cdot (t+1)^{4p\mathtt{tc}'}$. Note that no vertex will be labeled by a label in $\{0, 1, \ldots, (2p-1)|X'|-1\} \cup \{k-(2p-1)|X'|+1, \ldots, k\}$ hereafter. Thus we consider how we use labels in $\{(2p-1)|X'|, \ldots, k-(2p-1)|X'|\}$ for $V \setminus X'$, which does not yield any conflict with X'.

We then formulate how many labels should be placed in $L_0, L_1, \ldots, L_t$ for one partial labeling using $\{0, 1, \ldots, (2p-1)|X'|-p\} \cup \{k-(2p-1)|X'|+p, \ldots, k\}$ as Integer Linear Programming. For a fixed partial labeling, let a_i be the number of labels that have been already assigned to L_i, and x_i be a variable representing the number of labels used in L_i in the desired labeling.

The following is the ILP formulation.

$$\begin{cases} x_0 + \cdots + x_t \leq k+1 & \\ x_i \geq a_i, & \text{for } i \in \{0, \ldots, t\} \\ \sum_{i:T_j \in \mathcal{C}_i} x_i = |T_j|, & \text{for } j \in \{1, \ldots, s\} \end{cases}$$

The first constraint shows that the total number of labels is at most $k+1$. Note that the number of unused labels is x_0. The second one is for consistency to the partial labeling. The last one, which is the most important, guarantees that every vertex in T_j can receive a label; the number of usable labels is $|\{i \mid T_j \in \mathcal{C}_i\}|$, because a label in L_i is used exactly once per T_j.

If the above ILP has a feasible solution, it is possible to assign labels to all the vertices in $V \setminus X'$ if we ignore the distance-1 condition inside of each clique. Actually, we can see that the information is sufficient to give a proper k-$L(p,1)$-labeling. At the beginning of the algorithm, we take twin cover X', which means that for every $T_i \subseteq V \setminus X'$, $n_i \geq p\omega_i$ holds. Since cliques in $G[T_i]$ have common neighbors and $n_i \geq p\omega_i$, only the number of available labels matters by Lemma 3. Since the existence of an ILP solution guarantees this, we can decide whether a partial labeling can be extended to a proper k-$L(p,1)$-labeling, or not.

Because $t \leq 2^s \leq 2^{2^{\mathtt{tc}'}}$, the number of variables of the ILP is at most $2^{2^{\mathtt{tc}'}}$; it can be solved in FPT time with respect to $\mathtt{tc}'$ [30]. Since $\mathtt{tc}' \leq \mathtt{tc} + 2^{\mathtt{tc}} \cdot p \cdot \omega$, the total running time is FPT time with respect to $\mathtt{tc} + \omega$. □

4.2 $L(1,1)$-Labeling parameterized by tc

Unlike $L(p,1)$-labeling with $p \geq 2$, the distance-1 condition of $L(1,1)$-labeling requires just that the labels between adjacent vertices are different. Thus, $L(1,1)$-Labeling seems to be easier than $L(p,1)$-Labeling with $p \geq 2$. Actually, we can show that $L(1,1)$-Labeling is fixed-parameter tractable parameterized only by twin cover number.

Lemma 5 *For a connected graph G, let X be a non-empty twin cover of G and G' is a graph obtained from G by deleting all the twin edges in $G[V \setminus X]$. Then, any $L(1,1)$-labeling on G' is an $L(1,1)$-labeling on G and vice versa.*

Proof: The statement is true, if $N_G^{\leq 2}[v] = N_{G'}^{\leq 2}[v]$ holds for any vertex $v \in V$, and we show this below. We first show this for $v \in X$. Since X is a twin cover in G and only twin edges in $G[V \setminus X]$ are deleted, $N_G^{\leq 2}[v] = N_{G'}^{\leq 2}[v]$ holds for any $v \in X$. Note that $G[V \setminus X]$ forms a set of cliques of twins and each vertex in such a clique has the same neighborhood in X.

Next, we show $N_G^{\leq 2}[v] = N_{G'}^{\leq 2}[v]$ for $v \in V \setminus X$. Let C be the clique in $G[V \setminus X]$ that contains v. Since we only delete the twin edges in $G[V \setminus X]$, the distance from v to $w \in V \setminus C$ in G' and the distance in G are the same. For $v, w \in C$, since they are true twins in G, there is a common neighbor in X in G', which implies that the distance between v and w is two in G'. Therefore, for every $v \in V$, $N_G^{\leq 2}[v] = N_{G'}^{\leq 2}[v]$ holds. This completes the proof. □

Corollary 3 *For G' defined as above, $\lambda_{1,1}(G') = \lambda_{1,1}(G)$ holds.*

Then we give an fixed-parameter algorithm for $L(1,1)$-Labeling parameterized by twin cover number. First, we compute a minimum twin cover X of G in time $O(1.2738^{\mathtt{tc}} + \mathtt{tc}n + m)$ [18]. If there exists an empty twin cover of G, it consists of complete graphs. In this case, we can immediately obtain an optimal labeling. Otherwise, we compute the $L(1,1)$-labeling number of G', which equals to the $L(1,1)$-labeling number of G by Corollary 3. Since X is a twin cover in G, it is a vertex cover in G' by the deletion of twin edges in $G[V \setminus X]$. Because $L(1,1)$-Labeling is fixed-parameter tractable when parameterized by vertex cover number [16], we have the following theorem.

Theorem 3 *$L(1,1)$-Labeling is fixed-parameter tractable when parameterized by twin cover number.*

Since $\lambda_{1,1}(G) \leq \lambda_{p,1}(G) \leq \lambda_{p,p}(G) = p\lambda_{1,1}(G)$ holds, an $L(1,1)$-labeling gives an approximation for $L(p,1)$-Labeling. In fact, by replacing the labels of an optimal $L(1,1)$-labeling of G with multiples of p, we obtain an $L(p,1)$-labeling whose approximation factor is at most p.

Corollary 4 *For $L(p,1)$-Labeling, there is a fixed-parameter p-approximation algorithm with respect to twin cover number.*

5 Concluding Remarks

Some FPT results hold for more general settings, that is, $L(p,q)$-Labeling with any constant p and q. For example, $L(p,q)$-Labeling with any constant p and q is FPT when parameterized by tree-width plus maximum degree. This implies that bounding maximum degree is essential for NP-hardness, because $L(p,q)$-Labeling for trees (i.e., graphs with tree-width 1) is NP-hard for every pair of p and q having no common divisor [15].

An interesting open question is whether $L(p,1)$-Labeling parameterized by only twin cover number is FPT or not.

Acknowledgements

We are grateful to Dr. Yota Otachi for his insightful comments. We also thank the anonymous reviewers for their helpful comments.

References

[1] Y. Asahiro, H. Eto, T. Hanaka, G. Lin, E. Miyano, and I. Terabaru. Parameterized algorithms for the happy set problem. In *International Conference and Workshops on Algorithms and Computation (WALCOM 2020)*, pages 323–328. Springer, 2020. doi:10.1007/978-3-030-39881-1_27.

[2] J. Blum. Hierarchy of Transportation Network Parameters and Hardness Results. In *International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148, pages 4:1–4:15, 2019. doi:10.4230/LIPIcs.IPEC.2019.4.

[3] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. doi:10.1137/130947374.

[4] H. L. Bodlaender, T. Kloks, R. B. Tan, and J. Van Leeuwen. Approximations for λ-colorings of graphs. *The Computer Journal*, 47(2):193–204, 2004. doi:10.1093/comjnl/47.2.193.

[5] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 2008.

[6] T. Calamoneri. The $L(h,k)$-labelling problem: an updated survey and annotated bibliography. *The Computer Journal*, 54(8):1344–1371, 2011. doi:10.1093/comjnl/bxr037.

[7] G. J. Chang and D. Kuo. The L(2,1)-labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9(2):309–316, 1996. doi:10.1137/S0895480193245339.

[8] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. *Theory of Computing Systems*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

[9] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

[10] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[11] N. Eggemann, F. Havet, and S. D. Noble. k-$L(2,1)$-labelling for planar graphs is NP-complete for $k \geq 4$. *Discrete Applied Mathematics*, 158(16):1777–1788, 2010. doi:10.1016/j.dam.2010.06.016.

[12] H. Eto, T. Hanaka, Y. Kobayashi, and Y. Kobayashi. Parameterized Algorithms for Maximum Cut with Connectivity Constraints. In *International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148, pages 13:1–13:15, 2019. doi:10.4230/LIPIcs.IPEC.2019.13.

[13] J. Fiala, T. Gavenčiak, D. Knop, M. Koutecký, and J. Kratochvíl. Parameterized complexity of distance labeling and uniform channel assignment problems. *Discrete Applied Mathematics*, 248:46–55, 2018. doi:10.1016/j.dam.2017.02.010.

[14] J. Fiala, P. A. Golovach, and J. Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. In *International Colloquium on Automata, Languages, and Programming (ICALP 2005)*, pages 360–372. Springer, 2005. doi:10.1007/11523468_30.

[15] J. Fiala, P. A. Golovach, and J. Kratochvíl. Computational complexity of the distance constrained labeling problem for trees (extended abstract). In *International Colloquium on Automata, Languages, and Programming (ICALP 2008)*, pages 294–305. Springer, 2008. doi:10.1007/978-3-540-70575-8_25.

[16] J. Fiala, P. A. Golovach, and J. Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theoretical Computer Science*, 412(23):2513–2523, 2011. doi:10.1016/j.tcs.2010.10.043.

[17] J. Fiala, T. Kloks, and J. Kratochvíl. Fixed-parameter complexity of λ-labelings. *Discrete Applied Mathematics*, 113(1):59 – 72, 2001. doi:10.1016/S0166-218X(00)00387-5.

[18] R. Ganian. Improving vertex cover as a graph parameter. *Discrete Mathematics and Theoretical Computer Science*, 17(2):77–100, 2015. URL: http://dmtcs.episciences.org/2136.

[19] S. Gaspers and K. Najeebullah. Optimal surveillance of covert networks by minimizing inverse geodesic length. In *AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 533–540, 2019.

[20] D. Gonçalves. On the $L(p,1)$-labelling of graphs. *Discrete Mathematics*, 308(8):1405 – 1414, 2008. doi:10.1016/j.disc.2007.07.075.

[21] J. R. Griggs and R. K. Yeh. Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595, 1992. doi:10.1137/0405048.

[22] F. Gurski and E. Wanke. The tree-width of clique-width bounded graphs without $k_{n,n}$. In *Graph-Theoretic Concepts in Computer Science*, pages 196–205. Springer, 2000. doi:10.1007/3-540-40064-8_19.

[23] W. K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[24] M. M. Halldórsson. Approximating the $L(h,k)$-labelling problem. *International Journal of Mobile Network Design and Innovation*, 1(2):113–117, 2006.

[25] T. Hanaka, K. Kawai, and H. Ono. Computing $L(p,1)$-labeling with combined parameters. In *International Conference and Workshops on Algorithms and Computation (WALCOM 2021)*, pages 208–220, 2021. doi:10.1007/978-3-030-68211-8_17.

[26] T. Hasunuma, T. Ishii, H. Ono, and Y. Uno. A linear time algorithm for $L(2,1)$-labeling of trees. *Algorithmica*, 66(3):654–681, 2013. doi:10.1007/s00453-012-9657-z.

[27] T. Hasunuma, T. Ishii, H. Ono, and Y. Uno. Algorithmic aspects of distance constrained labeling: a survey. *International Journal of Networking and Computing*, 4(2):251–259, 2014.

[28] B. M. P. Jansen and A. Pieterse. Optimal data reduction for graph coloring using low-degree polynomials. *Algorithmica*, 81(10):3865–3889, 2019. doi:10.1007/s00453-019-00578-5.

[29] D. Knop, T. Masarík, and T. Toufar. Parameterized Complexity of Fair Vertex Evaluation Problems. In *International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138, pages 33:1–33:16, 2019. doi:10.4230/LIPIcs.MFCS.2019.33.

[30] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

[31] N. Misra and H. Mittal. Imbalance parameterized by twin cover revisited. In *Computing and Combinatorics (COCOON 2020)*, pages 162–173. Springer, 2020. doi:10.1016/j.tcs.2021.09.017.

[32] F. S. Roberts. T-colorings of graphs: recent results and open problems. *Discrete Mathematics*, 93(2):229 – 245, 1991. doi:10.1016/0012-365X(91)90258-4.

[33] I. Todinca. Coloring powers of graphs of bounded clique-width. In *Graph-Theoretic Concepts in Computer Science (WG 2003)*, pages 370–382. Springer, 2003. doi:10.1007/978-3-540-39890-5_32.

[34] X. Zhou, Y. Kanari, and T. Nishizeki. Generalized vertex-colorings of partial k-trees. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E83-A(4):671–678, 2000.