Exploring the Gap Between Treedepth and Vertex Cover Through Vertex Integrity*

Tatsuya Gima[1], Tesshu Hanaka[2], Masashi Kiyomi[3], Yasuaki Kobayashi[4], and Yota Otachi[1]

[1] Nagoya University, Nagoya, Japan
gima@nagoya-u.jp, otachi@nagoya-u.jp
[2] Chuo University, Bunkyo-ku, Tokyo, Japan
hanaka.91t@g.chuo-u.ac.jp
[3] Yokohama City University, Yokohama, Japan
masashi@yokohama-cu.ac.jp
[4] Kyoto University, Kyoto, Japan
kobayashi@iip.ist.i.kyoto-u.ac.jp

Abstract. For intractable problems on graphs of bounded treewidth, two graph parameters treedepth and vertex cover number have been used to obtain fine-grained complexity results. Although the studies in this direction are successful, we still need a systematic way for further investigations because the graphs of bounded vertex cover number form a rather small subclass of the graphs of bounded treedepth. To fill this gap, we use vertex integrity, which is placed between the two parameters mentioned above. For several graph problems, we generalize fixed-parameter tractability results parameterized by vertex cover number to the ones parameterized by vertex integrity. We also show some finer complexity contrasts by showing hardness with respect to vertex integrity or treedepth.

Keywords: vertex integrity, vertex cover number, treedepth.

1 Introduction

Treewidth, which measures how close a graph is to a tree, is arguably one of the most powerful tools for designing efficient algorithms for graph problems. The application of treewidth is quite wide and the general theory built there often gives a very efficient algorithm (e.g., [10,2,17]). However, still many problems are found to be intractable on graphs of bounded treewidth (e.g., [50]). To cope with such problems, one may use pathwidth, which is always larger than or equal to treewidth. Unfortunately, this approach did not quite work as no natural problem was known to change its complexity with respect to treewidth and pathwidth, until very recently [8]. Treedepth is a further restriction of pathwidth. However, still most of the problems do not change their complexity, except for some

* Partially supported by JSPS KAKENHI Grant Numbers JP18H04091, JP18K11168, JP18K11169, JP19K21537, JP20K19742, JP20H05793.

problems with hardness depending on the existence of long paths (e.g., [24,39]). One successful approach in this direction is parameterization by the vertex cover number, which is a strong restriction of treedepth. Many problems that are intractable parameterized by treewidth have been shown to become tractable when parameterized by vertex cover number [27,25,28,1,41,14].

One drawback of the vertex-cover parameterization is its limitation to a very small class of graphs. To overcome the drawback, we propose a new approach for parameterizing graph problems by vertex integrity [5]. The *vertex integrity* of a graph G, denoted $\mathsf{vi}(G)$, is the minimum integer k satisfying that there is $S \subseteq V(G)$ such that $|S|+|V(C)| \leq k$ for each component C of $G-S$. We call such S a $\mathsf{vi}(k)$-*set* of G. This parameter is bounded from above by vertex cover number + 1 and from below by treedepth. As a structural parameter in parameterized algorithms, vertex integrity (and its close variants) was used only in a couple of previous studies [23,33,12]. Our goal is to fill some gaps between treedepth and vertex cover number by presenting finer algorithmic and complexity results parameterized by vertex integrity. Note that the parameterization by vertex integrity is equivalent to the one by ℓ-component order connectivity $+ \ell$ [22].

Short preliminaries. For the basic terms and concepts in the parameterized complexity theory, we refer the readers to standard textbooks, e.g. [21,19].

For a graph G, we denote its treewidth by $\mathsf{tw}(G)$, pathwidth by $\mathsf{pw}(G)$, treedepth by $\mathsf{td}(G)$, and vertex cover number by $\mathsf{vc}(G)$. (See Section A for definitions.) It is known that $\mathsf{tw}(G) \leq \mathsf{pw}(G) \leq \mathsf{td}(G) - 1 \leq \mathsf{vi}(G) - 1 \leq \mathsf{vc}(G)$ for every graph G. We say informally that a problem is fixed-parameter tractable "parameterized by vi", which means "parameterized by the vertex integrity of the input graphs." We also say "graphs of $\mathsf{vi} = c$ (or $\mathsf{vi} \leq c$)".

Our results. The main contribution of this paper is to generalize several known FPT algorithms parameterized by vc to the ones by vi. We also show some results considering parameterizations by vc, vi, or td to tighten the complexity gaps between parameterizations by vc and by td. See Table 1 for the summary of results. Due to the space limitation, we had to move most of the results into the appendix. In the main text, we present full descriptions of selected results only. (Even for the selected results, we still have to omit some proofs. They are marked with ★.)

Extending FPT results parameterized by vc. We show that Imbalance, Maximum Common (Induced) Subgraph, Capacitated Vertex Cover, Capacitated Dominating Set, Precoloring Extension, Equitable Coloring, and Equitable Connected Partition are fixed-parameter tractable parameterized by vertex integrity. We present the algorithms for Imbalance as a simple but still powerful example that generalizes known results (Section 2) and for Maximum Common Subgraph as one of the most involved examples (Section 3). See Section E for the other problems. A commonly used trick is to reduce the problem instance to a number of instances of integer linear programming, while each problem requires a nontrivially tailored reduction depending on its structure. It was the same for parameterizations by vc, but the reductions here are more involved because of the generality of vi. Finding the similarity

Table 1. Summary. The results stated without references are shown in this paper.

Problem	Lower bounds	Upper bounds
Imbalance	NP-h [9]	FPT by $\mathsf{tw}+\Delta$ [42] FPT by vi
Max Common Subgraph Max Common Ind. Subgraph	NP-h for $\mathsf{vi}(G_2)=3$ NP-h for $\mathsf{vc}(G_2)=0$	FPT by $\mathsf{vi}(G_1)+\mathsf{vi}(G_2)$
Capacitated Vertex Cover	W[1]-h by td [20]	FPT by vi
Capacitated Dominating Set	W[1]-h by $\mathsf{td}+k$ [20]	FPT by vi
Precoloring Extension	W[1]-h by td [26]	FPT by vi
Equitable Coloring	W[1]-h by td [26]	FPT by vi
Equitable Connected Part.	W[1]-h by pw [25]	FPT by vi
Bandwidth	W[1]-h by td NP-h for $\mathsf{pw}=2$ [46]	FPT by vc [27] P for $\mathsf{pw}\leq 1$ [4]
Graph Motif	NP-h for $\mathsf{vi}=4$	FPT by vc [14] P for $\mathsf{vi}\leq 3$
Steiner Forest Unweighted Steiner Forest	NP-h for $\mathsf{vi}=5$ [35] NP-h for $\mathsf{tw}=3$ [35]	XP by vc FPT by vc
Unary Min Max Outdeg. Ori. Binary Min Max Outdeg. Ori.	W[1]-h by vc NP-h for $\mathsf{vc}=3$	XP by tw [51] P for $\mathsf{vc}\leq 2$
Metric Dimension	W[1]-h by pw [13]	FPT by $\mathsf{tw}+\Delta$ [6] FPT by td
Directed (p,q)-Edge Dom. Set	W[1]-h by pw [7]	FPT by $\mathsf{tw}+p+q$ [7] FPT by td
List Hamiltonian Path	W[1]-h by pw [43]	FPT by td

among the reductions and algorithms would be a good starting point to develop a general way for handling problems parameterized by vi (or vc). Additionally, we show that Bandwidth is W[1]-hard parameterized by td, while we were not able to extend the algorithm parameterized by vc to the one by vi.

Filling some complexity gaps. We observe that Graph Motif and Steiner Forest have different complexity with respect to vc and vi (Section F). In particular, we see that not all FPT algorithms parameterized by vc can be generalized to the ones by vi. Min Max Outdegree Orientation gives an example that a known hardness for td can be strengthened to the one for vc (Section 4). We additionally observe that some W[1]-hard problems parameterized by tw become tractable parameterized by td. Such problems include Metric Dimension, Directed (p,q)-Edge Dominating Set, and List Hamiltonian Path (Section G).

2 Imbalance

In this section, we show that Imbalance is fixed-parameter tractable parameterized by vi. Let $G=(V,E)$ be a graph. Given a linear ordering σ on V, the *imbalance* $\mathsf{im}_\sigma(v)$ of $v\in V$ is the absolute difference of the numbers of the neighbors of v that appear before v and after v in σ. The *imbalance* of G, de-

noted $\mathsf{im}(G)$, is defined as $\min_\sigma \sum_{v \in V} \mathsf{im}(v)$, where the minimum is taken over all linear orderings on V. Given a graph G and an integer b, Imbalance asks whether $\mathsf{im}(G) \leq b$.

Fellows et al. [27] showed that Imbalance is fixed-parameter tractable parameterized by vc. Recently, Misra and Mittal [44] have extended the result by showing that Imbalance is fixed-parameter tractable parameterized by the sum of the twin-cover number and the maximum twin-class size. Although twin-cover number is incomparable with vertex integrity, the combined parameter in [44] is always larger than or equal to the vertex integrity of the same graph. On the other hand, the combined parameter can be arbitrarily large for some graphs of constant vertex integrity (e.g., disjoint unions of P_3's). Hence, our result here properly extends the result in [44] as well.

Key concepts. Before proceeding to the algorithm, we need to introduce two important concepts that are common in our algorithms parameterized by vi.

1. *ILP parameterized by the number of variables.* It is known that the feasibility of an instance of integer linear programming (ILP) parameterized by the number of variables is fixed-parameter tractable [40]. Using the algorithm for the feasibility problem as a black box, one can show the same fact for the optimization version as well. (See Section B for the detail.) This fact has been used heavily for designing FPT algorithms parameterized by vc (see e.g. [27]). We are going to see that some of these algorithms can be generalized for the parameterization by vi, and Imbalance is the first such example.

2. *Equivalence relation among components.* For a vertex set S of G, we define an equivalence relation $\sim_{G,S}$ among components of $G - S$ by setting $C_1 \sim_{G,S} C_2$ if and only if there is an isomorphism g from $G[S \cup V(C_1)]$ to $G[S \cup V(C_2)]$ that fixes S; that is, $g|_S$ is the identity function. When $C_1 \sim_{G,S} C_2$, we say that C_1 and C_2 have the same (G, S)-*type* (or just the same *type* if G and S are clear from the context). See Fig. 1. We say that a component C of $G - S$ is of (G, S)-*type* t (or just *type* t) by using a canonical form t of the members of the (G, S)-type equivalence class of C. We can set the canonical form t in such a way that it can be computed from S and C in time depending only on $|S \cup V(C)|$.[5] Observe that if S is a $\mathsf{vi}(k)$-set of G, then the number of $\sim_{G,S}$ classes depends only on k since $|S \cup V(C)| \leq k$ for each component C of $G - S$. Hence, we can compute for all types t the number of type-t components of $G - S$ in $O(f(k) \cdot n)$ total running time, where $n = |V|$ and $f(k)$ is a computable function depending only on k. Note that this information (the numbers of type-t components for all t) completely characterizes the graph G up to isomorphism.

Theorem 2.1. Imbalance *is fixed-parameter tractable parameterized by* vi.

Proof. Let S be a $\mathsf{vi}(k)$-set of G. Such a set can be found in $O(k^{k+1}n)$ time [22]. We first guess and fix the relative ordering of S in an optimal ordering. There

[5] For example, by fixing the ordering of vertices in S as $v_1, \ldots, v_{|S|}$, we can set t to be the adjacency matrix of $G[S \cup V(C)]$ such that the ith row and column correspond to v_i for $1 \leq i \leq |S|$ and under this condition the string $t[1,1], \ldots, t[1,s], t[2,1], \ldots, t[s,s]$ is lexicographically minimal, where $s = |S \cup V(C)|$.

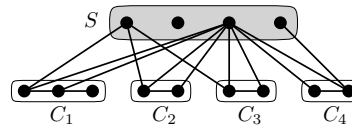

Fig. 1. The components C_2 and C_3 of $G - S$ have the same (G, S)-type.

are only $k!$ candidates for this guess. For each $v \in S$, let $\ell(v)$ and $r(v)$ be the numbers of vertices in $N(v) \cap S$ that appear before v and after v, respectively, in the guessed relative ordering of S.

Observe that the imbalance of a vertex v in a component C of $G-S$ depends only on the relative ordering of $S \cup V(C)$ since $N(v) \subseteq S \cup V(C)$. For each type t and for each relative ordering p of $S \cup V(C)$, where C is a type-t component of $G-S$, we denote by $\mathsf{im}(t,p)$ the sum of imbalance of the vertices in C. Similarly, the numbers of vertices in a type-t component C that appear before $v \in S$ and after v depend only on the relative ordering p of $S \cup V(C)$; we denote these numbers by $\ell(v,t,p)$ and $r(v,t,p)$, respectively. The numbers $\mathsf{im}(t,p)$, $\ell(v,t,p)$, and $r(v,t,p)$ can be computed from their arguments in time depending only on k, and thus they are treated as constants in the following ILP.

We represent by a nonnegative variable $x_{t,p}$ the number of type-t components that have relative ordering p with S. Note that the number of combinations of t and p depends only on k. For each $v \in S$, we represent (an upper bound of) the imbalance of v by an auxiliary variable y_v. This can be done by the following constraints:

$$y_v \geq (\ell(v) + \textstyle\sum_{t,p} \ell(v,t,p) \cdot x_{t,p}) - (r(v) + \sum_{t,p} r(v,t,p) \cdot x_{t,p}),$$
$$y_v \geq (r(v) + \textstyle\sum_{t,p} r(v,t,p) \cdot x_{t,p}) - (\ell(v) + \sum_{t,p} \ell(v,t,p) \cdot x_{t,p}).$$

Then the imbalance of the whole ordering, which is our objective function to minimize, can be expressed as

$$\textstyle\sum_{v \in S} y_v + \sum_{t,p} \mathsf{im}(t,p) \cdot x_{t,p}.$$

Now we need the following constraints to keep the total number of type-t components right:

$$\textstyle\sum_p x_{t,p} = c_t \quad \text{for each type } t,$$

where c_t is the number of components of type t in $G - S$.

By finding an optimal solution to the ILP above for each guess of the relative ordering of S, we can find an optimal ordering. Since the number of guesses and the number of variables depend only on k, the theorem follows. □

3 Maximum Common (Induced) Subgraph

In this section, we show that Maximum Common Subgraph (MCS) and Maximum Common Induced Subgraph (MCIS) are fixed-parameter tractable pa-

rameterized by vi of both graphs. (See Section C for the proof for MCIS.) The results extend known results and fill some complexity gaps as described below.

A graph Q is *subgraph-isomorphic* to G, denoted $Q \preceq G$, if there is an injection η from $V(Q)$ to $V(G)$ such that $\{\eta(u), \eta(v)\} \in E(G)$ for every $\{u, v\} \in E(Q)$. A graph Q is *induced subgraph-isomorphic* to G, denoted $Q \preceq_{\mathrm{I}} G$, if there is an injection η from $V(Q)$ to $V(G)$ such that $\{\eta(u), \eta(v)\} \in E(G)$ if and only if $\{u, v\} \in E(Q)$. Given two graphs G and Q, Subgraph Isomorphism (SI) asks whether $Q \preceq G$, and Induced Subgraph Isomorphism (ISI) asks whether $Q \preceq_{\mathrm{I}} G$. The results of this section are on their generalizations. Given two graphs G_1 and G_2, MCS asks to find a graph H with maximum $|E(H)|$ such that $H \preceq G_1$ and $H \preceq G_2$. Similarly, MCIS asks to find a graph H with maximum $|V(H)|$ such that $H \preceq_{\mathrm{I}} G_1$ and $H \preceq_{\mathrm{I}} G_2$.

If we restrict the structure of only one of the input graphs, then both problems remain quite hard. Since Partition Into Triangles [34] is a special case of SI where the graph Q is a disjoint union of triangles, MCS is NP-hard even if one of the input graphs has $\mathsf{vi} = 3$. Also, since Independent Set [34] is a special case of ISI where Q is an edge-less graph, MCIS is NP-hard even if one of the input graphs has $\mathsf{vc} = 0$. Furthermore, since SI and ISI generalize Clique [21], MCS and MCIS are W[1]-hard parameterized by the order of one of the input graphs. When parameterized by vc of one graph, an XP algorithm for (a generalization of) MCS is known [11].

For parameters restricting both input graphs, some partial results were known. It is known that SI is fixed-parameter tractable parameterized by vi of both graphs, while it is NP-complete when both graphs have $\mathsf{td} \leq 3$ [12]. The hardness proof in [12] can be easily adapted to ISI without increasing td. It is known that MCIS is fixed-parameter tractable parameterized by vc of both graphs [1].

Theorem 3.1. Maximum Common Subgraph *is fixed-parameter tractable parameterized by* vi *of both input graphs.*

Proof. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the input graphs of vertex integrity at most k. We will find isomorphic subgraphs $\Gamma_1 = (U_1, F_1)$ of G_1 and $\Gamma_2 = (U_2, F_2)$ of G_2 with maximum number of edges, and an isomorphism $\eta\colon U_1 \to U_2$ from Γ_1 to Γ_2.

Step 1. Guessing matched $\mathsf{vi}(2k)$*-sets* R_1 *and* R_2*.* Let S_1 and S_2 be $\mathsf{vi}(k)$-sets of G_1 and G_2, respectively. At this point, there is no guarantee that $S_i \subseteq U_i$ or $\eta(S_1) = S_2$. To have such assumptions, we make some guesses about η and find $\mathsf{vi}(2k)$-sets R_1 and R_2 of the graphs such that $\eta(R_1) = R_2$.

Step 1-1. Guessing subsets $X_i, Y_i \subseteq S_i$ *for* $i \in \{1, 2\}$*.* We guess disjoint subsets X_1 and Y_1 of S_1 such that $X_1 = S_1 \cap \eta^{-1}(U_2 \cap S_2)$ and $Y_1 = S_1 \cap \eta^{-1}(U_2 \setminus S_2)$. We also guess disjoint subsets X_2 and Y_2 of S_2 defined similarly as $X_2 = S_2 \cap \eta(U_1 \cap S_1)$ and $Y_2 = S_2 \cap \eta(U_1 \setminus S_1)$. Note that $\eta(X_1) = X_2$. There are $3^{|S_1|} \cdot 3^{|S_2|} \leq 3^{2k}$ candidates for the combinations of X_1, Y_1, X_2, and Y_2.

Observe that the vertices in $S_i \setminus (X_i \cup Y_i)$ do not contribute to the isomorphic subgraphs and can be safely removed. We denote the resultant graphs by H_i.

Step 1-2. Guessing η on $X_1 \cup Y_1$ and η^{-1} on $X_2 \cup Y_2$. Given the guessed subsets X_1, Y_1, X_2, and Y_2, we further guess how η maps these subsets. There are $|X_1|! \leq k!$ candidates for the bijection $\eta|_{X_1}$ (equivalently for $\eta^{-1}|_{X_2} = (\eta|_{X_1})^{-1}$).

Now we guess $\eta|_{Y_1}$ from at most 2^{k^3} non-isomorphic candidates as follows. Recall that $\eta(Y_1) \subseteq V_2 \setminus S_2$. Observe that each subset $A \subseteq V_2 \setminus S_2$ is completely characterized up to isomorphism by the numbers of ways A intersects type-t components for all (H_2, S_2)-types t. Since there are at most $2^{\binom{k}{2}}$ types and each component has order at most k, the total number of non-equivalent subsets of components is at most $2^{\binom{k}{2}} \cdot 2^k \leq 2^{k^2}$. Since $\eta(Y_1)$ is the union of at most $|Y_1|$ such subsets, the number of non-isomorphic candidates of $\eta(Y_1)$ is at most $(2^{k^2})^{|Y_1|} \leq 2^{k^3}$. In the analogous way, we can guess $\eta^{-1}|_{Y_2}$ from at most 2^{k^3} non-isomorphic candidates.

Now we set $Z_1 = \eta^{-1}(Y_2)$ and $Z_2 = \eta(Y_1)$. Let $R_1 = X_1 \cup Y_1 \cup Z_1$ and $R_2 = X_2 \cup Y_2 \cup Z_2$. Observe that each component C of $H_1 - R_1$ satisfies that $|C| \leq k - |S_1| \leq k$ and $|C| + |R_1| \leq (k - |S_1|) + (|S_1| + |\eta^{-1}(Y_2)|) \leq 2k$. Hence, R_1 is a $\mathsf{vi}(2k)$-set of H_1. Similarly, we can see that R_2 is a $\mathsf{vi}(2k)$-set of H_2. Furthermore, we know that $\eta(R_1) = R_2$.

Step 2. Extending the guessed parts of η. Assuming that the guesses we made so far are correct, we now find the entire η. Recall that we are seeking for isomorphic subgraphs $\Gamma_1 = (U_1, F_1)$ of G_1 and $\Gamma_2 = (U_2, F_2)$ of G_2 with maximum number of edges, and the isomorphism $\eta\colon U_1 \to U_2$ from Γ_1 to Γ_2. Since we already know the part $\eta|_{R_1}\colon R_1 \to R_2$, it suffices to find a bijective mapping from a subset of $V(H_1 - R_1)$ to a subset of $V(H_2 - R_1)$ that maximizes the number of matched edges where the connections to R_i are also taken into account.

As we describe below, the subproblem we consider here can be solved by formulating it as an ILP instance with $2^{O(k^3)}$ variables. The trick here is that instead of directly finding the mapping, we find which vertices and edges in $H_i - R_i$ are used in the common subgraph.

In the following, we are going to use a generalized version of *types* since the vertex set of a component of $H_i - R_i$ does not necessarily induce a connected subgraph of Γ_i. It is defined in a similar way as (H_i, R_i)-types except that it is defined for each pair (A, B) of a connected subgraph A of $H_i - R_i$ and a subset B of the edges between A and R_i. Let (A_1, B_1) and (A_2, B_2) be such pairs in $H_i - R_i$. We say that (A_1, B_1) and (A_2, B_2) have the same *g-(H_i, R_i)-type* (or just *g-type*) if there is an isomorphism from $H_i(A_1, B_1)$ to $H_i(A_2, B_2)$ that fixes R_i, where $H_i(A_j, B_j)$ is the subgraph of H_i formed by B_j and the edges in A_j. See Fig. 2. We say that a pair (A, B) is of *g-(H_i, R_i)-type t* (or just *g-type t*) by using a canonical form t of the g-(H_i, R_i)-type equivalence class of (A, B). Observe that all possible canonical forms of g-types can be computed in time depending only on k.

Step 2-1. Decomposing components of $H_i - R_i$ into smaller pieces. We say that an edge $\{u, v\}$ in H_1 is *used by η* if $u, v \in U_1$ and H_2 has the edge $\{\eta(u), \eta(v)\}$. Similarly, an edge $\{u, v\}$ in H_2 is *used by η* if $u, v \in U_2$ and H_1 has the edge $\{\eta^{-1}(u), \eta^{-1}(v)\}$.

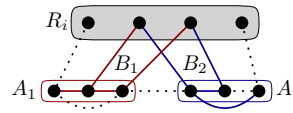

Fig. 2. The pairs (A_1, B_1) and (A_2, B_2) have the same g-(H_i, R_i)-type.

Let $i \in \{1, 2\}$, t be an (H_i, R_i)-type, and T be a multiset of g-(H_i, R_i)-types. Let C be a type t component of $H_i - R_i$, C' the subgraph of C formed by the edges used by η, and E' the subset of the edges between C' and R_i used by η. If T coincides with the multiset of g-types of the pairs (A, B) such that A is a component of C' and B is the subset of E' connecting A and R_i, then we say that η *decomposes* the type-t component C into T.

We represent by a nonnegative variable $x^{(i)}_{t,T}$ the number of type-t components of $H_i - R_i$ that are decomposed into T by η. We have the following constraint:

$$\textstyle\sum_T x^{(i)}_{t,T} = c^{(i)}_t \quad \text{for each } (H_i, R_i)\text{-type } t \text{ and } i \in \{1, 2\},$$

where the sum is taken over all possible multisets T of g-(H_i, R_i)-types, and $c^{(i)}_t$ is the number of components of type t in $H_i - R_i$. Additionally, if there is no way to decompose a type-t component into T, we add a constraint $x^{(i)}_{t,T} = 0$.

As each component of $H_i - R_i$ has order at most k, T contains at most k elements. Since there are at most $2^{\binom{2k}{2}}$ g-types, there are at most $(2^{\binom{2k}{2}})^k$ options for choosing T. Thus the number of variables $x^{(i)}_{t,T}$ is at most $2 \cdot 2^{\binom{2k}{2}} \cdot (2^{\binom{2k}{2}})^{k+1}$.

Now we introduce a nonnegative variable $y^{(i)}_t$ that represents the number of pairs (A, B) of g-type t obtained from the components of $H_i - R_i$ by decomposing them by η. The definition of $y^{(i)}_t$ gives the following constraint:

$$y^{(i)}_t = \textstyle\sum_{t',T} \mu(T, t) \cdot x^{(i)}_{t',T} \quad \text{for each g-}(H_i, R_i)\text{-type } t \text{ and } i \in \{1, 2\},$$

where $\mu(T, t)$ is the multiplicity of g-type t in T and the sum is taken over all possible (H_i, R_i)-types t' and multisets T of g-(H_i, R_i)-types. As in the previous case, we can see that the number of variables y_t depends only on k.

Step 2-2. Matching decomposed pieces. Observe that for each g-(H_1, R_1)-type t_1, there exists a unique g-(H_2, R_2)-type t_2 such that there is an isomorphism g from $H_1(A_1, B_1)$ to $H_2(A_2, B_2)$ with $g|_{R_1} = \eta|_{R_1}$, where (A_i, B_i) is a pair of g-(H_i, R_i)-type t_i for $i \in \{1, 2\}$. We say that such g-types t_1 and t_2 *match*. Since η is an isomorphism from Γ_1 to Γ_2, η maps each g-(H_1, R_1)-type t_1 pair to a g-(H_2, R_2)-type t_2 pair, where t_1 and t_2 match. This implies that $y^{(1)}_{t_1} = y^{(2)}_{t_2}$, which we add as a constraint. Now the total number of edges used by η can be computed from $y^{(1)}_t$. Let m_t be the number of edges in $H_1(A, B)$, where (A, B) is a pair of g-(H_1, R_1)-type t. Let r be the number of matched edges in R_1; that is, $r = |\{\{u, v\} \in E(H_1[R_1]) \mid \{\eta(u), \eta(v)\} \in E(G_2[R_2])\}|$. Then, the number of matched edges is $r + \sum_t m_t \cdot y^{(1)}_t$. On the other hand, given an assignment to the

variables, it is easy to find isomorphic subgraphs with that many edges. Since r is a constant here, we set $\sum_t m_t \cdot y_t^{(1)}$ to the objective function to be maximized.

Since the number of candidates in the guesses we made and the number of variables in the ILP instances depend only on k, the theorem follows. □

4 Min Max Outdegree Orientation

Given an undirected graph $G = (V, E)$, an edge weight function $w\colon E \to \mathbb{Z}^+$, and a positive integer r, Min Max Outdegree Orientation (MMOO) asks whether there exists an orientation Λ of G such that each vertex has outdegree at most r under Λ, where the outdegree of a vertex is the sum of the weights of outgoing edges. If each edge weight is given in binary, we call the problem Binary MMOO, and if it is given in unary, we call the problem Unary MMOO. Note that in the binary version, the weight of an edge can be exponential in the input size, whereas the unary version does not allow such weights.

Unary MMOO admits an $n^{O(\mathsf{tw})}$-time algorithm [51], but it is W[1]-hard parameterized by td [50].[6] In this section, we show a stronger hardness parameterized by vc. Binary MMOO is known to be NP-complete for graphs of $\mathsf{vi} = 4$ [3]. In Section D, we show a stronger hardness result that the binary version is NP-complete for graphs of $\mathsf{vc} = 3$. This result is tight as we can show that the binary version is polynomial-time solvable for graphs of $\mathsf{vc} \leq 2$.

Theorem 4.1. Unary MMOO *is W[1]-hard parameterized by* vc.

Proof. We give a parameterized reduction from Unary Bin Packing. Given a positive integer t and n positive integers $a_1, a_2, \ldots, a_n$ in unary, Unary Bin Packing asks the existence of a partition $S_1, \ldots, S_t$ of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in S_j} a_i = \frac{1}{t} \sum_{1 \leq i \leq n} a_i$ for $1 \leq j \leq t$. Unary Bin Packing is W[1]-hard parameterized by t [37].

We assume that $t \geq 3$ since otherwise the problem can be solved in polynomial time as the integers a_i are given in unary. Let $B = \frac{1}{t} \sum_{1 \leq i \leq n} a_i$ and $W = (t-1)B = \sum_{1 \leq i \leq n} a_i - B$. The assumption $t \geq 3$ implies that $B \leq W/2$. Observe that if $a_i \geq B$ for some i, then the instance is a trivial no instance (when $a_i > B$) or the element a_i is irrelevant (when $a_i = B$). Hence, we assume that $a_i < B$ (and thus $a_i < W/2$) for every i.

The reduction to Unary MMOO is depicted in Fig. 3. From the integers $a_1, a_2, \ldots, a_n$, we construct the graph obtained from a complete bipartite graph on the vertex set $\{u, s_1, s_2, \ldots, s_t\} \cup \{v_1, \ldots, v_n\}$ by adding the edge $\{u, s_1\}$. We set $w(\{v_i, s_j\}) = a_i$ for all i, j, $w(\{v_i, u\}) = W - a_i$ for all i, and $w(\{u, s_1\}) = W$. The vertices $s_1, s_2, \ldots, s_t, u$ form a vertex cover of size $t + 1$. We set the target maximum outdegree r to W. We show that this instance of Unary MMOO is a yes instance if and only if there exists a partition $S_1, \ldots, S_t$ of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in S_j} a_i = B$ for all j. Intuitively, we can translate the solutions of

[6] In [50], W[1]-hardness was stated for tw but the proof shows it for td as well.

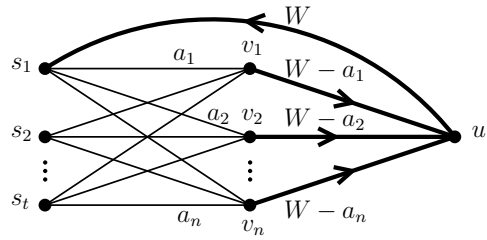

Fig. 3. Reduction from Unary Bin Packing to Unary MMOO.

the problems by picking a_i into S_j if $\{v_i, s_j\}$ is oriented from v_i to s_j, and vice versa.

Assume that there exists a partition $S_1, \ldots, S_t$ of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in S_j} a_i = B$ for all j. We first orient the edge $\{u, s_1\}$ from u to s_1 and each edge $\{v_i, u\}$ from v_i to u. (See the thick edges in Fig. 3.) Then, we orient $\{v_i, s_j\}$ from v_i to s_j if and only if $i \in S_j$. Under this orientation, all vertices have outdegree exactly W: $a_i + (W - a_i)$ for each v_i and $\sum_{i \notin S_j} a_i = \sum_{1 \leq i \leq n} a_i - B$ for each s_j.

Conversely, assume that there is an orientation such that each vertex has outdegree at most W. Since the sum of the edge weights is $(n + t + 1)W$ and the graph has $n + t + 1$ vertices, the outdegree of each vertex has to be exactly W. Since $a_i < W/2$ for all i, each edge $\{v_i, u\}$ has weight larger than $W/2$. Hence, for u, the only way to obtain outdegree exactly W is to orient $\{u, s_1\}$ from u to s_1 and $\{v_i, u\}$ from v_i to u for all i. Furthermore, for each i, there exists exactly one vertex s_j such that $\{v_i, s_j\}$ is oriented from v_i to s_j. Let $S_j \subseteq \{1, 2, \ldots, n\}$ be the set of indices i such that $\{v_i, s_j\}$ is oriented from v_i to s_j. The discussion above implies that $S_1, \ldots, S_t$ is a partition of $\{1, \ldots, n\}$. The outdegree of s_j is $\sum_{i \notin S_j} a_i$, which is equal to $W = \sum_{1 \leq i \leq n} a_i - B$. Thus, $\sum_{i \in S_j} a_i = \sum_{1 \leq i \leq n} a_i - W = B$. □

5 Bandwidth

Let $G = (V, E)$ be a graph. Given a linear ordering σ on V, the *stretch* of $\{u, v\} \in E$, denoted $\mathsf{str}_\sigma(\{u, v\})$, is $|\sigma(u) - \sigma(v)|$. The *bandwidth* of G, denoted $\mathsf{bw}(G)$, is defined as $\min_\sigma \max_{e \in E} \mathsf{str}_\sigma(e)$, where the minimum is taken over all linear orderings on V. Given a graph G and an integer w, Bandwidth asks whether $\mathsf{bw}(G) \leq w$. Bandwidth is NP-complete on trees of $\mathsf{pw} = 3$ [45] and on graphs of $\mathsf{pw} = 2$ [46]. Fellows et al. [27] presented an FPT algorithm for Bandwidth parameterized by vc . Here we show that Bandwidth is W[1]-hard parameterized by td on trees. The proof is inspired by the one by Muradian [46].

Theorem 5.1. Bandwidth *is W[1]-hard parameterized by* td *on trees.*

Proof. Let $(a_1, \ldots, a_n; t)$ be an instance of Unary Bin Packing with $t \geq 2$. Let $B = \frac{1}{t}\sum_{1 \leq i \leq n} a_i$ be the target weight. We construct an equivalent instance $(T = (V, E), w)$ of Bandwidth as follows (see Fig. 4). We start with a path $(z_0, x_1, y_1, z_1, \ldots, x_t, y_t, z_t)$ of length $3t$. For $1 \leq i \leq t-1$, we attach $12tnB$ leaves to z_i. To z_0 and z_t, we attach $12tnB + 4n + 1$ leaves. For $1 \leq i \leq n$, we take a star with $6tn \cdot a_i - 1$ leaves centered at v_i. Finally, we connect each v_i to x_1 with a path with $6t - 4$ inner vertices. We set the target width w to $6tnB + 2n + 1$. Note that $|V| = (3t + 2)w + 1$.

We can see an upper bound of $\mathsf{td}(T)$ as follows. We remove x_1 and all the leaves from T. This decreases treedepth by at most 2. The remaining graph is a disjoint union of paths and a longest path has order $6t - 3$. Since $\mathsf{td}(P_n) = \lceil \log_2(n+1) \rceil$ [47], we have $\mathsf{td}(T) \leq 2 + \lceil \log_2(6t-2) \rceil \leq \log_2 t + 6$.

Now we show that (T, w) is a yes instance of of Bandwidth if and only if $(a_1, \ldots, a_n; t)$ is a yes instance of Unary Bin Packing.

($\Longrightarrow$) First assume that $\mathsf{bw}(T) \leq w$ and that σ is a linear ordering on V such that $\max_{e \in E} \mathsf{str}_\sigma(e) \leq w$. Since $\deg(z_0) = 12tnB + 4n + 2 = 2w$, its closed neighborhood $N[z_0]$ has to appear in σ consecutively, where z_0 appears at the middle of this subordering. Furthermore, no edge can connect a vertex appearing before z_0 in σ and a vertex appearing after z_0 as such an edge has stretch larger than w. Since the edges not incident to z_0 form a connected subgraph, we can conclude that the vertices in $V - N[z_0]$ appear either all before $N[z_0]$ or all after $N[z_0]$ in σ. By symmetry, we can assume that those vertices appear after $N[z_0]$ in σ. This implies that $\sigma(z_0) = w + 1$. By the same argument, we can show that all vertices in $N[z_t]$ appear consecutively in the end of σ and $\sigma(z_t) = |V| - w = (3t + 1)w + 1$. Since $\sigma(z_t) - \sigma(z_0) = 3tw$ and the path $(z_0, x_1, y_1, z_1, \ldots, x_t, y_t, z_t)$ has length $3t$, each edge in this path has stretch exactly w in σ. Namely, $\sigma(x_i) = (3i-1)w + 1$, $\sigma(y_i) = 3iw + 1$, and $\sigma(z_i) = (3i+1)w + 1$.

For each leaf ℓ attached to z_i $(1 \leq i \leq t-1)$, $\sigma(y_i) < \sigma(\ell) < \sigma(x_{i+1})$ holds. Other than these leaves, there are $2(w-1) - 12tnB = 4n$ vertices placed between y_i and x_{i+1}. Let V_i be the set consisting of v_i and the leaves attached to it. For $j \in \{1, \ldots, t\}$, let I_j be the set of indices i such that v_i is put between z_{j-1} and

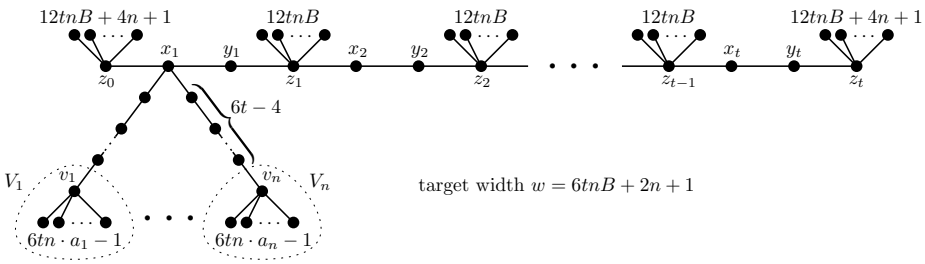

Fig. 4. Reductions from Unary Bin Packing to Bandwidth.

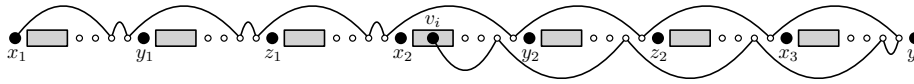

Fig. 5. Embedding the path from x_1 to v_i. The gray boxes are the occupied position and the white points are the vacant positions. ($n = 2$, $j = 2$, $t = 3$.)

z_j. If $i \in I_j$, then all $6tn \cdot a_i$ vertices in V_i are put between y_{j-1} and x_{j+1}. (We set $y_0 \coloneqq z_0$.)

If $\sum_{i \in I_j} a_i \geq B + 1$, then $|\bigcup_{i \in I_j} V_i| \geq 6tn(B+1) > w + 8n - 1$ as $t \geq 2$. This number of vertices cannot be put between y_{j-1} and x_{j+1} after putting the leaves attached to z_{j-1} and z_j: we can put at most $4n$ vertices between y_{j-1} and x_j, at most $4n$ vertices between y_j and x_{j+1}, and at most $w-1$ vertices between x_j and y_j. Since $I_1, \ldots, I_t$ form a partition of $\{1, \ldots, n\}$ and $\sum_{1 \leq i \leq n} a_i = tB$, we can conclude that $\sum_{i \in I_j} a_i = B$ for $1 \leq j \leq t$.

($\Longleftarrow$) Next assume that there exists a partition $S_1, \ldots, S_t$ of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in S_j} a_i = B$ for all $1 \leq j \leq t$.

We put $N[z_0]$ at the beginning of σ and $N[z_t]$ at the end. We set $\sigma(x_i) = (3i-1)w + 1$, $\sigma(y_i) = 3iw + 1$, and $\sigma(z_i) = (3i+1)w + 1$. For $1 \leq i \leq t-1$, we put the leaves attached to z_i so that a half of them have the first $6tnB$ positions between y_i and z_i and the other half have the first $6tnB$ positions between z_i and x_{i+1}. For each S_j, we put the vertices in $\bigcup_{i \in S_j} V_i$ so that they take the first $6tnB$ positions between x_j and y_j.

Now we have $2n$ vacant positions at the end of each interval between x_i and y_i for $1 \leq i \leq t$, between y_i and z_i for $1 \leq i \leq t-1$, and between z_i and x_{i+1} for $1 \leq i \leq t-1$. To these positions, we need to put the inner vertices of the paths connecting x_1 and $v_1, \ldots, v_n$. Let P_i be the inner part of x_1–v_i path. The path P_i uses the $(2i-1)$st and $(2i)$th vacant positions in each interval as follows (see Fig. 5).

Let $i \in S_j$. Starting from x_1, P_i proceeds from left to right and visits the two positions in each interval consecutively until it arrives the interval between x_j and y_j. At the interval between x_j and y_j, P_i switches to the phase where it only visits the $(2i)$th vacant position in each interval and still proceeds from left to right until it reaches the interval between x_t and y_t. Then P_i changes the direction and switches to the phase where it visits the $(2i-1)$st vacant position only in each interval until it reaches the interval between x_j and y_j.

Now all the vertices are put at distinct positions and it is easy to see that no edge has stretch more than w. This completes the proof. □

6 Conclusion

Using vertex integrity as a structural graph parameter, we presented finer analyses of the parameterized complexity of well-studied problems. Although we needed a case-by-case analysis depending on individual problems, the results in

this paper would be useful for obtaining a general method to deal with vertex integrity.

Although we succeeded to extend many fixed-parameter algorithms parameterized by vc to the ones parameterized by vi, we were not so successful on graph layout problems. Fellows et al. [27] showed that Imbalance, Bandwidth, Cutwidth, and Distortion are fixed-parameter tractable parameterized by vc. Lokshtanov [41] showed that Optimal Linear Arrangement is fixed-parameter tractable parameterized by vc. Are these problems fixed-parameter tractable parameterized by vi? We answered only for Imbalance in this paper.

References

1. Faisal N. Abu-Khzam. Maximum common induced subgraph parameterized by vertex cover. *Inf. Process. Lett.*, 114(3):99–103, 2014. doi:10.1016/j.ipl.2013.11.007.
2. Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991. doi:10.1016/0196-6774(91)90006-K.
3. Yuichi Asahiro, Eiji Miyano, and Hirotaka Ono. Graph classes and the complexity of the graph orientation minimizing the maximum weighted outdegree. *Discret. Appl. Math.*, 159(7):498–508, 2011. doi:10.1016/j.dam.2010.11.003.
4. S. F. Assmann, G. W. Peck, M. M. Sysło, and J. Zak. The bandwidth of caterpillars with hairs of length 1 and 2. *SIAM Journal on Algebraic Discrete Methods*, 2(4):387–393, 1981. doi:10.1137/0602041.
5. Curtis A. Barefoot, Roger C. Entringer, and Henda C. Swart. Vulnerability in graphs — a comparative survey. *J. Combin. Math. Combin. Comput.*, 1:13–22, 1987.
6. Rémy Belmonte, Fedor V. Fomin, Petr A. Golovach, and M. S. Ramanujan. Metric dimension of bounded tree-length graphs. *SIAM J. Discret. Math.*, 31(2):1217–1243, 2017. doi:10.1137/16M1057383.
7. Rémy Belmonte, Tesshu Hanaka, Ioannis Katsikarelis, Eun Jung Kim, and Michael Lampis. New results on directed edge dominating set. In *MFCS 2018*, volume 117 of *LIPIcs*, pages 67:1–67:16, 2018. doi:10.4230/LIPIcs.MFCS.2018.67.
8. Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. In *ESA 2020*, volume 173 of *LIPIcs*, pages 14:1–14:19, 2020. doi:10.4230/LIPIcs.ESA.2020.14.
9. Therese C. Biedl, Timothy M. Chan, Yashar Ganjali, Mohammad Taghi Hajiaghayi, and David R. Wood. Balanced vertex-orderings of graphs. *Discret. Appl. Math.*, 148(1):27–48, 2005. doi:10.1016/j.dam.2004.12.001.
10. Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *ICALP 1988*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118, 1988. doi:10.1007/3-540-19488-6_110.
11. Hans L. Bodlaender, Tesshu Hanaka, Lars Jaffke, Hirotaka Ono, Yota Otachi, and Tom C. van der Zanden. Hedonic seat arrangement problems (extended abstract). In *AAMAS 2020*, pages 1777–1779, 2020. URL: https://dl.acm.org/doi/abs/10.5555/3398761.3398979.
12. Hans L. Bodlaender, Tesshu Hanaka, Yoshio Okamoto, Yota Otachi, and Tom C. van der Zanden. Subgraph isomorphism on graph classes that exclude a substructure. In *CIAC 2019*, volume 11485 of *Lecture Notes in Computer Science*, pages 87–98, 2019. doi:10.1007/978-3-030-17402-6_8.

13. Édouard Bonnet and Nidhi Purohit. Metric dimension parameterized by treewidth. In *IPEC 2019*, volume 148 of *LIPIcs*, pages 5:1–5:15, 2019. doi:10.4230/LIPIcs.IPEC.2019.5.
14. Édouard Bonnet and Florian Sikora. The graph motif problem parameterized by the structure of the input graph. *Discret. Appl. Math.*, 231:78–94, 2017. doi:10.1016/j.dam.2016.11.016.
15. Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.
16. Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005. doi:10.1137/S0097539701385351.
17. Bruno Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *RAIRO Theor. Informatics Appl.*, 26:257–286, 1992. doi:10.1051/ita/1992260302571.
18. Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.
19. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
20. Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In *IWPEC 2008*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90, 2008. doi:10.1007/978-3-540-79723-4_9.
21. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999. doi:10.1007/978-1-4612-0515-9.
22. Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016. doi:10.1007/s00453-016-0127-x.
23. Pavel Dvořák, Eduard Eiben, Robert Ganian, Dušan Knop, and Sebastian Ordyniak. Solving integer linear programs with a small number of global variables and constraints. In *IJCAI 2017*, pages 607–613, 2017. doi:10.24963/ijcai.2017/85.
24. Pavel Dvořák and Dušan Knop. Parameterized complexity of length-bounded cuts and multicuts. *Algorithmica*, 80(12):3597–3617, 2018. doi:10.1007/s00453-018-0408-7.
25. Rosa Enciso, Michael R. Fellows, Jiong Guo, Iyad A. Kanj, Frances A. Rosamond, and Ondrej Suchý. What makes equitable connected partition easy. In *IWPEC 2009*, volume 5917 of *Lecture Notes in Computer Science*, pages 122–133, 2009. doi:10.1007/978-3-642-11269-0_10.
26. Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.*, 209(2):143–153, 2011. doi:10.1016/j.ic.2010.11.026.
27. Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph layout problems parameterized by vertex cover. In *ISAAC 2008*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305, 2008. doi:10.1007/978-3-540-92182-0_28.
28. Jiří Fiala, Petr A. Golovach, and Jan Kratochvíl. Parameterized complexity of coloring problems: Treewidth versus vertex cover. *Theor. Comput. Sci.*, 412(23):2513–2523, 2011. doi:10.1016/j.tcs.2010.10.043.

29. András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987. doi:10.1007/BF02579200.
30. Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *STOC 1983*, pages 448–456, 1983. doi:10.1145/800061.808776.
31. Robert Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In *IPEC 2011*, volume 7112 of *Lecture Notes in Computer Science*, pages 259–271, 2011. doi:10.1007/978-3-642-28050-4_21.
32. Robert Ganian. Using neighborhood diversity to solve hard problems. *CoRR*, abs/1201.3091, 2012. arXiv:1201.3091.
33. Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica*, 2020. doi:10.1007/s00453-020-00758-8.
34. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
35. Elisabeth Gassner. The steiner forest problem revisited. *J. Discrete Algorithms*, 8(2):154–163, 2010. doi:10.1016/j.jda.2009.05.002.
36. Petr Hlinený, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. doi:10.1093/comjnl/bxm052.
37. Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *J. Comput. Syst. Sci.*, 79(1):39–49, 2013. doi:10.1016/j.jcss.2012.04.004.
38. Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12:415–440, 1987. doi:10.1287/moor.12.3.415.
39. Leon Kellerhals and Tomohiro Koana. Parameterized complexity of geodetic set. In *IPEC 2020*, volume 180 of *LIPIcs*, pages 20:1–20:14, 2020. doi:10.4230/LIPIcs.IPEC.2020.20.
40. Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983. doi:10.1287/moor.8.4.538.
41. Daniel Lokshtanov. Parameterized integer quadratic programming: Variables and coefficients. *CoRR*, abs/1511.00310, 2015. arXiv:1511.00310.
42. Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Imbalance is fixed parameter tractable. *Inf. Process. Lett.*, 113(19-21):714–718, 2013. doi:10.1016/j.ipl.2013.06.010.
43. Kitty Meeks and Alexander Scott. The parameterised complexity of list problems on graphs of bounded treewidth. *Inf. Comput.*, 251:91–103, 2016. doi:10.1016/j.ic.2016.08.001.
44. Neeldhara Misra and Harshil Mittal. Imbalance parameterized by twin cover revisited. In *COCOON 2020*, volume 12273 of *Lecture Notes in Computer Science*, pages 162–173, 2020. doi:10.1007/978-3-030-58150-3_13.
45. Burkhard Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM J. Algebraic and Discrete Methods*, 7(4):505–512, 1986. doi:10.1137/0607057.
46. David Muradian. The bandwidth minimization problem for cyclic caterpillars with hair length 1 is NP-complete. *Theor. Comput. Sci.*, 307(3):567–572, 2003. doi:10.1016/S0304-3975(03)00238-X.
47. Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*. Algorithms and combinatorics. Springer, 2012. doi:10.1007/978-3-642-27875-4.

48. Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):10:1–10:20, 2008. doi:10.1145/1435375.1435385.
49. Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. doi:10.1007/978-3-662-43948-7_77.
50. Stefan Szeider. Not so easy problems for tree decomposable graphs. *Ramanujan Mathematical Society, Lecture Notes Series*, No. 13:179–190, 2010. arXiv:1107.1177.
51. Stefan Szeider. Monadic second order logic on graphs with local cardinality constraints. *ACM Trans. Comput. Log.*, 12(2):12:1–12:21, 2011. doi:10.1145/1877714.1877718.

A Graph parameters

We give formal definitions of $\mathsf{vc}(G)$, $\mathsf{td}(G)$, and $\mathsf{vi}(G)$ only. See [36] for the definitions of $\mathsf{pw}(G)$, $\mathsf{tw}(G)$, and clique-width $\mathsf{cw}(G)$.

A.1 Vertex cover

Let $G = (V, E)$ be a graph. A set $S \subseteq V$ is a *vertex cover* of G if each component of $G - S$ has exactly one vertex. The *vertex cover number* of G, denoted $\mathsf{vc}(G)$, is the size of a minimum vertex cover of G. It is known that a vertex cover of size k (if exists) can be found in time $O(2^k \cdot n)$ [19], where $n = |V|$ (see [15] for the currently fastest algorithm). Thus we can assume that a vertex cover of minimum size is given when designing an algorithm parameterized by vc.

A.2 Treedepth

The treedepth of a graph $G = (V, E)$, denoted $\mathsf{td}(G)$, is defined recursively as follows:

$$\mathsf{td}(G) = \begin{cases} 1 & |V| = 1, \\ \max_{1 \le i \le c} \mathsf{td}(C_i) & G \text{ has } c \ge 2 \text{ components } C_1, \ldots, C_c, \\ 1 + \min_{v \in V} \mathsf{td}(G - v) & \text{otherwise.} \end{cases}$$

In other words, a graph $G = (V, E)$ has treedepth at most d if there is a rooted forest F of height at most d on the same vertex set V such that two vertices are adjacent in G only if one is an ancestor of the other in F. It is known that such F, if exists, can be found in time $2^{O(d^2)} \cdot n$ [49], where $n = |V|$. So we assume that such a rooted forest of depth $\mathsf{td}(G)$ is given together with G when the parameter is td.

From the rooted forest F, one can easily construct a path decomposition of G with maximum bag size at most d: use leaves as bags and put all ancestors of a leaf into the bag corresponding to the leaf. This implies that $\mathsf{pw}(G) + 1 \le \mathsf{td}(G)$ for every graph G. On the other hand, td cannot be bounded by any function of

pw in general. For example, $\mathsf{pw}(P_n) = 1$ and $\mathsf{td}(P_n) = \lceil \log_2(n+1) \rceil$ [47], where P_n is the path of order n.

In general, we have the following upper bound of the length of paths.

Proposition A.1 ([47]). *The length of a longest path in G is less than $2^{\mathsf{td}(G)}$.*

A.3 Vertex integrity

The *vertex integrity* [5] of a graph G, denoted $\mathsf{vi}(G)$, is the minimum integer k satisfying that there is a vertex set $S \subseteq V(G)$ such that $|S| + |V(C)| \leq k$ for each component C of $G - S$. We call such S a $\mathsf{vi}(k)$-*set* of G. For an n-vertex graph of vertex integrity at most k, we can find a $\mathsf{vi}(k)$-*set* in $O(k^{k+1}n)$ time [22]. Hence, without loss of generality, we can assume that a $\mathsf{vi}(k)$-set is given as a part of input when designing an FPT algorithm parameterized by vi. Observe that $\mathsf{td}(G) \leq \mathsf{vi}(G)$ since we can first remove the $k' \leq k$ vertices in a $\mathsf{vi}(k)$-set and then each component has order at most $k - k'$ and thus treedepth at most $k - k'$. Also, since a vertex cover of size k is a $\mathsf{vi}(k+1)$-set, $\mathsf{vi}(G) \leq \mathsf{vc}(G) + 1$ holds.

Dvořák et al. [23] showed that Integer Linear Programming (ILP) is fixed-parameter tractable parameterized by the fracture number of the incidence, which is basically equivalent to the vertex integrity. Ganian et al. [33] showed that Bounded Degree Deletion is W[1]-hard parameterized by treedepth but fixed-parameter tractable parameterized by core fracture number, which can be seen as a generalization of vertex integrity. Bodlaender et al. [12] showed that Subgraph Isomorphism is fixed-parameter tractable parameterized by the vertex integrity of both graphs, while the problem is NP-complete for graphs of treedepth 3.

B ILP parameterized by the number of variables

Lenstra [40] showed that the feasibility of an integer linear programming (ILP) formula can be decided in FPT time when parameterized by the number of variables. The time and space complexity was later improved by Kannan [38] and by Frank and Tardos [29]. Their algorithms can be used also for the following ILP optimization problem (see e.g., [27]).

p-Opt-ILP
Input: A matrix $A \in \mathbb{Z}^{m \times p}$, vectors $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^p$.
Task: Find a vector $x \in \mathbb{Z}^p$ that minimizes $c^\top x$ and satisfies that $Ax \geq b$.

Proposition B.1 ([40,38,29]). p-Opt-ILP *can be solved using $O(p^{2.5p+o(p)} \cdot L \cdot \log(MN))$ arithmetic operations and space polynomial in L, where L is the number of bits in the input, N is the maximum absolute value any variable can take, and M is an upper bound on the absolute value of the minimum taken by the objective function.*

C Omitted proofs in Section 3

Theorem C.1. Maximum Common Induced Subgraph *is fixed-parameter tractable parameterized by the sum of the vertex integrity of input graphs.*

Proof. Since the proof is almost the same with the one for Theorem 3.1, here we only describe the differences for handling induced subgraphs.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the input graphs of vertex integrity at most k. We will find $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ with maximum size $|U_1| = |U_2|$ such that there is an isomorphism η from $G_1[U_1]$ to $G_2[U_2]$.

Step 1. Guessing matched $\mathsf{vi}(2k)$*-sets* R_1 *and* R_2*.* In the same way as before, we guess $\mathsf{vi}(2k)$-sets R_1 and R_2 of G_1 and G_2, respectively, and a bijection $\eta|_{R_1} \colon R_1 \to R_2$. The only difference here is that we reject the current guess if $\eta|_{R_1}$ is not an isomorphism from $G_1[R_1]$ to $G_2[R_2]$.

Step 2. Extending the guessed parts of η*.* To handle induced subgraphs, we need to modify the definition of "to decompose" as follows: for a type-t component C of $H_i - R_i$ and a multiset T of g-(H_i, R_i)-types, we say that $\eta \colon U_1 \to U_2$ *decomposes* C into T if T coincides with the multiset of g-(H_i, R_i)-types of the pairs (A, B) such that A is a component of $H_i[V(C) \cap U_i]$ and B is the set of all edges connecting A and R_i. Everything else works as before. □

D Omitted proofs in Section 4

Theorem D.1. Binary MMOO *is NP-complete for graphs of* $\mathsf{vc} = 3$.

Proof. Since the problem clearly belongs to NP, we show the NP-hardness by presenting a reduction from Partition, which is NP-complete [34]. Given an even number of positive integers $a_1, a_2, \ldots, a_n$ in binary, Partition asks the existence of a partition $\{S_1, S_2\}$ of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in S_j} a_i = \frac{1}{2} \sum_{1 \leq i \leq n} a_i$ for $j \in \{1, 2\}$. This problem remains NP-hard with an additional condition $|S_1| = |S_2| = n/2$ [34]. We assume that $n \geq 10$ since otherwise the problem can be solved in polynomial time. Let $B = W = \frac{1}{2} \sum_{1 \leq i \leq n} a_i$.

The proof is almost the same with the one of Theorem 4.1 except that $t = 2$. Observe that we assumed there that $t \geq 3$ only to guarantee that $a_i < W/2$ for all i. To have this assumption here, we start with an instance $a_1, a_2, \ldots, a_n$ of Partition with the restriction $|S_1| = |S_2| = n/2$. Let $a'_i = a_i + B$ for each i, and $B' = W' = \frac{1}{2} \sum_{1 \leq i \leq n} a'_i = (n/2 + 1)B$. Clearly, this is an equivalent instance as we added the same value to each number. Also, $a'_i < W'/2$ holds for all i since $n \geq 10$ and $a_i < 2B$ imply that $a'_i = a_i + B < 3B \leq (n/2+1)B/2 = W'/2$. Now we observe that the restriction $|S_1| = |S_2| = n/2$ is not a restriction anymore. That is, if $\sum_{i \in S} a'_i = B'$ for some $S \subseteq \{1, \ldots, n\}$, then $|S| = n/2$ holds. Suppose to the contrary that $S \neq n/2$. By swapping S and $\{1, \ldots, n\} \setminus S$ if necessary, we can assume that $S \leq n/2 - 1$. This gives $(n/2+1)B = \sum_{i \in S} a'_i \leq (n/2-1)B + \sum_{i \in S} a_i$, which implies $\sum_{i \in S} a_i \geq 2B = \sum_{1 \leq i \leq n} a_i$, a contradiction.

We construct an instance of Binary MMOO as exactly we did in the proof of Theorem 4.1 by setting $t = 2$ and using a'_i, B', and W' instead of a_i, B, and W. The equivalence of the instances can be shown in the same way. □

Theorem D.2. Binary MMOO *can be solved in polynomial time for graphs of* $\mathsf{vc} \leq 2$.

Proof. Let $G = (V, E)$, $w\colon E \to \mathbb{Z}^+$, $r \in \mathbb{Z}^+$ be an instance of Binary MMOO. We assume that $w(e) \leq r$ for each $e \in E$ since otherwise the problem is trivial. If there is a vertex of degree at most 1, we can safely remove it from the graph since we can always orient the edge incident to the vertex (if exists) from the vertex to the other endpoint. Hence, we assume that G has minimum degree at least 2.

Let $\{p, q\} \subseteq V$ be a vertex cover of G. By the assumption on the minimum degree, every vertex $v \in V \setminus \{p, q\}$ is adjacent to both p and q. If $w(\{v, p\}) + w(\{v, q\}) \leq r$, then we can safely orient the edges from v to p and q. Thus we remove such vertices from the graph. Now it holds that $w(\{v, p\}) + w(\{v, q\}) > r$ for all $v \in V \setminus \{p, q\}$. In particular, $\max\{w(\{v, p\}), w(\{v, q\})\} > r/2$ for all $v \in V \setminus \{p, q\}$.

Observe that for each vertex, at most one edge of weight more than $r/2$ can be oriented from the vertex to one of its neighbors. For p and q, we guess such edges. That is, we guess one edge of weight more than $r/2$ incident to p (q, resp.) and orient it from p (q, resp.) to the other endpoint; or guess that there is no such edge. These guesses determine almost a complete orientation. For a non-guessed edge $\{v, p\}$ with $w(\{v, p\}) > r/2$, we orient it from v to p. Since $w(\{v, p\}) + w(\{v, q\}) > r$, we then have to orient $\{v, q\}$ from q to v. The other case of $w(\{v, q\}) > r/2$ is symmetric. Now the only edge with undetermined orientation is $\{p, q\}$ (if they are adjacent). We just try both directions of $\{p, q\}$ and check if the whole orientation is of maximum outdegree at most r. □

E Extending algorithms known for vc parameterizations

E.1 Capacitated problems

Let $G = (V, E)$ be a graph with a capacity function $c\colon V \to \mathbb{Z}^+$ such that $c(v) \leq \deg(v)$ for each $v \in V$. A set $C \subseteq V$ is a *capacitated vertex cover* if there exists a mapping $f\colon E \to C$ such that $f(e)$ is an endpoint of e for each $e \in E$ and $|\{e \in E \mid f(e) = v\}| \leq c(v)$ for each $v \in C$. A set $D \subseteq V$ is a *capacitated dominating set* if there exists a mapping $f\colon V \setminus D \to D$ such that $f(v) \in N(v) \cap D$ for each $v \in V \setminus D$ and $|\{v \in V \setminus D \mid f(v) = u\}| \leq c(u)$ for each $u \in D$. Now the problems studied in this section are defined as follows.

Capacitated Vertex Cover
Input: A graph G, a capacity function $c\colon V \to \mathbb{Z}^+$, a positive integer k.
Question: Is there a capacitated vertex cover X of G with $|X| \leq k$?

Capacitated Dominating Set
Input: A graph G, a capacity function $c\colon V \to \mathbb{Z}^+$, a positive integer k.
Question: Is there a capacitated dominating set D of G with $|D| \leq k$?

It is known that Capacitated Vertex Cover is W[1]-hard parameterized by td, and Capacitated Dominating Set is W[1]-hard parameterized by $\mathsf{td} + k$ [20].[7]

For a vertex set S of G, we say that components C_1 and C_2 of $G - S$ have the same *c-type* if C_1 and C_2 have the same (G, S)-type and furthermore there is an isomorphism g from $G[S \cup V(C_1)]$ to $G[S \cup V(C_2)]$ such that $g|_S$ is the identity and $c(v) = c(g(v))$ for each $v \in S \cup V(C_1)$. We say that a component C of $G - S$ is of *c-type* t by using a canonical form t of the members of the c-type equivalence class of C. If S is a $\mathsf{vi}(k)$-set of G, then every vertex in $G - S$ has degree less than k in G, and thus its capacity is also less than k. This implies that the number of different c-types depends only on k.

Theorem E.1. Capacitated Vertex Cover *is fixed-parameter tractable parameterized by* vi.

Proof. We are going to find a minimum capacitated vertex cover X of G. Let S be a $\mathsf{vi}(k)$-set of the input graph $G = (V, E)$. We first guess the subset $X_S = X \cap S$ and the partial mapping $f_S\colon E(G[S]) \to X_S$ with $f_S(e) \in e$ for each $e \in E(G[S])$. The numbers of candidates for X_S and f_S depend only on k. For each $v \in X_S$, we set $c'(v) = c(v) - |\{e \in E(G[S]) \mid f_S(e) = v\}|$. Each $v \in X_S$ can cover $c'(v)$ edges between S and $V - S$.

Let C be a c-type t component of $G - S$. We say that a pair (W, f) of a subset $W \subseteq V(C)$ and a mapping $f\colon E(C) \cup E(V(C), S) \to W \cup X_S$ is *feasible* if $|\{e \mid f(e) = v\}| \leq c(v)$ for each $v \in W$ and $f(e) \in e$ for each $e \in E(C) \cup E(C, S)$.[8] The number of feasible pairs depends only on k. A feasible pair gives a cover of all edges in C and some edges between $V(C)$ and S, and it asks X_S to cover the remaining edges between $V(C)$ and S in a certain way. Now it suffices to find an assignment of feasible pairs to components of $G - S$ that minimizes the number of vertices used by the feasible pairs and does not exceed the capacity of any vertex in X_S.

We represent by a nonnegative variable $x_{t,W,f}$ the number of c-type t components C of $G - S$ such that $V(C) \cap X = W$ and (W, f) is a feasible pair. The number of such variables depends only on k. Let d_t be the number of components of c-type t in $G - S$. Since each component of $G - S$ has to be assigned a feasible pair, we have the following constraints:

$$\textstyle\sum_{W,\, f} x_{t,W,f} = d_t \quad \text{for each } c\text{-type } t.$$

The capacity constraints for X_S can be expressed as follows:

$$\textstyle\sum_{t,\, W,\, f} \#(f, v) \cdot x_{t,W,f} \leq c'(v) \quad \text{for each } v \in X_S,$$

[7] The W[1]-hardness results are stated only for tw and $\mathsf{tw} + k$ but the proofs actually show them for td and $\mathsf{td} + k$, respectively.
[8] For vertex sets A and B, $E(A, B)$ denotes the set of edges between A and B.

where $\#(f, v)$ is the number of edges that f maps to v. Finally, our objective function to minimize is $|X_S| + \sum_{t,\, W,\, f} |W| \cdot x_{t,W,f}$.

By finding an optimal solution to the ILP above for each guess of X_S and f_S, we can find the minimum capacitated vertex cover of G. Since the number of guesses and the number of variables depend only on k, the theorem follows by Proposition B.1. □

Theorem E.2. Capacitated Dominating Set *is fixed-parameter tractable parameterized by* vi

Proof. We are going to find a minimum capacitated dominating set D of G. Let S be a $\mathsf{vi}(k)$-set of the input graph $G = (V, E)$.

We first guess the partition (D_S, A_S, B_S) of S such that $D_S = D \cap S$, A_S is the set of vertices dominated by D_S, and B_S is the set of vertices dominated by $D \setminus S$. Next we guess the partial mapping $f_S \colon A_S \to D_S$ with $f_S(v) \in N(v) \cap D_S$ for each $v \in A_S$. The numbers of candidates for (D_S, A_S, B_S) and f_S depend only on k. For each $v \in D_S$, we set $c'(v) = c(v) - |\{u \in A_S \mid f_S(u) = v\}|$. Each $v \in D_S$ can dominate $c'(v)$ vertices in $V - S$.

Let C be a c-type t component of $G - S$. Let (D_C, A_C, B_C) be a partition of $V(C)$, $B'_S \subseteq B_S$, $f \colon A_C \cup B'_S \to D_C$, and $g \colon B_C \to D_S$. We say that $(D_C, A_C, B_C, B'_S, f, g)$ is *feasible* if $f(v) \in N(v) \cap D_C$ for each $v \in A_C \cup B'_S$, $g(v) \in N(v) \cap D_S$ for each $v \in B_C$, and $|\{u \in A_C \cup B'_S \mid f(u) = v\}| \leq c(v)$ for each $v \in D_C$. The number of feasible tuples depends only on k. A feasible tuple gives a domination of all vertices in $V(C) \setminus B_C$ and B'_S, and it asks D_S to dominate B_C in a certain way.

As before, it suffices to find an assignment of feasible tuples to components of $G - S$ that minimizes the number of vertices used by the feasible tuples and does not exceed the capacity of any vertex in D_S.

We represent by a nonnegative variable $x_{t,D_C,A_C,B_C,B'_S,f,g}$ the number of c-type t components C of $G-S$ such that $V(C) \cap D = D_C$ and $(D_C, A_C, B_C, B'_S, f, g)$ is a feasible tuple. The number of such variables depends only on k. Let d_t be the number of components of c-type t in $G - S$. Since each component of $G - S$ has to be assigned a feasible tuple, we have the following constraints:

$$\sum_{D_C,\, A_C,\, B_C,\, B'_S,\, f,\, g} x_{t,D_C,A_C,B_C,B'_S,f,g} = d_t \quad \text{for each } c\text{-type } t.$$

The capacity constraints for D_S can be expressed as follows:

$$\sum_{D_C,\, A_C,\, B_C,\, B'_S,\, f,\, g} \#(g, v) \cdot x_{t,D_C,A_C,B_C,B'_S,f,g} \leq c'(v) \quad \text{for each } v \in D_S,$$

where $\#(g, v)$ is the number of vertices that g maps to v. We also have to guarantee that each vertex in B_S is dominated by a vertex in $V - S$. This can be done by the following constraints:

$$\sum_{D_C,\, A_C,\, B_C,\, B'_S \ni v,\, f,\, g} x_{t,D_C,A_C,B_C,B'_S,f,g} \geq 1 \quad \text{for each } v \in B_S.$$

Finally, our objective function to minimize is

$$|D_S| + \sum_{D_C,\, A_C,\, B_C,\, B'_S,\, f,\, g} |D_C| \cdot x_{t,D_C,A_C,B_C,B'_S,f,g}.$$

As before the discussion so far implies the theorem. □

E.2 Coloring and partitioning problems

Precoloring Extension, Equitable Coloring, and Equitable Connected Partition form a first set of problems studied under the "treewidth versus vertex cover" perspective [25,26,28]. Equitable Coloring and Precoloring Extension are fixed-parameter tractable parameterized by vc [28] and W[1]-hard parameterized by td [26].[9] Equitable Connected Partition is fixed-parameter tractable parameterized by vc and W[1]-hard parameterized by pw [25].

In this section, we show that all the three problems are fixed-parameter tractable parameterized by vi.

Precoloring Extension Given a graph $G = (V, E)$, a precoloring $c_U \colon U \to \{1, \ldots, r\}$ for some $U \subseteq V$, and a positive integer r, Precoloring Extension asks whether G admits a proper r-coloring c such that $c(v) = c_U(v)$ for every $v \in U$.

Theorem E.3. Precoloring Extension *is fixed-parameter tractable parameterized by* vi.

Proof. Let $(G = (V, E), c_U, r)$ be an instance of Precoloring Extension. Let S be a vi(k)-set of G. For each $v \in V$, let $L(v)$ be the following set (the list of allowed colors):

$$L(v) = \begin{cases} \{c_U(v)\} & v \in U, \\ \{1, \ldots, r\} \setminus \{c_U(u) \mid u \in N(v) \cap U\} & v \in S \setminus U, \\ \{1, \ldots, \min\{r, k\}\} \setminus \{c_U(u) \mid u \in N(v) \cap U\} & v \in V \setminus (S \cup U). \end{cases}$$

Observe that there exists a proper r-coloring c of G with $c(v) = c_U(v)$ for all $v \in U$ if and only if there is a proper coloring c' of G with $c'(v) \in L(v)$. This is almost trivial except for the case of $v \in V \setminus (S \cup U)$, where we restrict the domain to $\{1, \ldots, k\}$ when $k < r$. This can be justified by considering the degree of v. Since S is a vi(k)-set and $v \notin S$, we have $\deg(v) < k$. Thus, after coloring $G - v$, v can always be colored with a color not used in its neighborhood.

Now in the list coloring setting, we can remove the vertices in U unless the instance is a trivial no instance with $\{u, v\} \in E$ such that $c_U(u) = c_U(v)$. In

[9] The W[1]-hardness results are stated only for tw but the proofs actually show them for td.

the following, we consider the graph where U is removed and still use the same symbols G and S.

Let v be a vertex with $|L(v)| \geq 2k$. By the definition of L, $v \in S$. Such a vertex can be safely removed: the vertices in $V \setminus S$ use colors only in $\{1, \ldots, k\}$; and the vertices in $S - v$ use at most $k - 1$ colors in $L(v)$. We now assume that $L(u) < 2k$ for all vertices in the graph.

Now we guess the coloring of S and then check independently for each component C of $G - S$ whether $G[S \cup V(C)]$ has a coloring consistent with L and the guessed coloring of S. The number of possible colorings of S is at most $(2k)^k$. Since $|S \cup V(C)| \leq k$, checking the existence of a consistent coloring can be done in time depending only on k. □

Equitable Coloring Given an n-vertex graph $G = (V, E)$ and a positive integer r, Equitable Coloring asks whether G admits a proper r-coloring c such that $|\{v \in V \mid c(v) = i\}| \in \{\lfloor n/r \rfloor, \lceil n/r \rceil\}$ for each $i \in \{1, \ldots, r\}$. We call such a coloring an *equitable r-coloring*.

Theorem E.4. Equitable Coloring *is fixed-parameter tractable parameterized by* vi.

Proof. Let $(G = (V, E), r)$ be an instance of Equitable Coloring, and S be a $\mathsf{vi}(k)$-set of G. We split the proof into two cases: $r \leq 2k$ and $r > 2k$. We reduce both cases to the feasibility test of the ILP defined as follows. By Proposition B.1, the theorem will follow.

Case 1: $r \leq 2k$. We guess a partition $S_1, \ldots, S_r$ of S such that each of them is an independent set and some of them may be empty. Since $|S| \leq k$ and $r \leq 2k$, the number of such partitions depends only on k. We interpret this partition as a coloring of $G[S]$ and try to extend this to the whole graph.

For a (G, S)-type t, a coloring $\mu \colon V(C) \to \{1, \ldots, r\}$ of a type-t component C of $G - S$ is *feasible* if $S_i \cup \{v \in V(C) \mid \mu(v) = i\}$ is an independent set for each i. We set $\mu_i = |\{v \in V(C) \mid \mu(v) = i\}|$.

We represent by a nonnegative variable $x_{t,\mu}$ the number of type-t components colored with a feasible μ. Since each component of $G - S$ has to be colored, we have the following constraints:

$$\textstyle\sum_{\mu} x_{t,\mu} = d_t \quad \text{for each } (G, S)\text{-type } t,$$

where d_t is the number of type-t components in $G - S$. The equitable constraints can be expressed as follows:

$$\textstyle\sum_{t,\mu} \mu_i \cdot x_{t,\mu} = \lceil n/r \rceil - |S_i| \quad \text{for each } i \in \{1, \ldots, b\},$$
$$\textstyle\sum_{t,\mu} \mu_i \cdot x_{t,\mu} = \lfloor n/r \rfloor - |S_i| \quad \text{for each } i \in \{b + 1, \ldots, r\},$$

where b is the remainder of n/r.

Case 2: $r > 2k$. In this case, we do not have an upper bound of r. The first trick is that we can still guess the coloring of S since we use at most k colors there. The second trick is that after checking the extendability of the k colors, the rest of the problem becomes trivial.

We guess a partition $S_1, \ldots, S_k$ of S and an integer a such that: each S_i is a possibly-empty independent set; and there are disjoint independent sets $W_1, \ldots, W_k$ such that $S_i \subseteq W_i$ for all i, $|W_i| = \lceil n/r \rceil$ for $1 \leq i \leq a$, and $|W_i| = \lfloor n/r \rfloor$ for $a+1 \leq i \leq k$. Then, $G' \coloneqq G - \bigcup_{1 \leq i \leq k} W_i$ has an equitable $r-k$ coloring if and only if G has an equitable r coloring having $W_1, \ldots, W_k$ as color classes.

We can show that actually G' always has an equitable $r-k$ coloring. Observe that each component in G' has order at most $k < r-k$ as G' is a subgraph of $G-S$. We now linearly order the vertices of G' in such a way that the vertices of a component appear consecutively. Then we color the first vertex in this ordering with color 1, the second one with color 2, and so on. Formally, we color the ith vertex in this ordering with color $(i \bmod (r-k)) + 1$. Since each component has order less than $r-k$, we never repeat a color in a component. Thus, this is an equitable $r-k$ coloring of G'. Therefore, it suffices to decide whether there exists the super sets $W_1, \ldots, W_k$.

Since we are searching for a partial coloring, we use a special character $*$ to indicate "not colored." We need to change the definition of feasibility. For a (G,S)-type t, a coloring $\mu \colon V(C) \to \{*\} \cup \{1, \ldots, k\}$ of a type-t component C of $G-S$ is *feasible*, if $S_i \cup \{v \in V(C) \mid \mu(v) = i\}$ is an independent set for each $i \neq *$. We set $\mu_i = |\{v \in V(C) \mid \mu(v) = i\}|$. Now the rest of the proof is exactly the same as before.

We represent by a nonnegative variable $x_{t,\mu}$ the number of type-t components colored with a feasible μ. Since each component of $G-S$ has to be colored, we have the following constraints: $\sum_{\mu} x_{t,\mu} = d_t$ for each (G,S)-type t, where d_t is the number of type-t components in $G-S$. The equitable constraints can be expressed as follows: $\sum_{t,\mu} \mu_i \cdot x_{t,\mu} = \lceil n/r \rceil - |S_i|$ for $1 \leq i \leq a$, and $\sum_{t,\mu} \mu_i \cdot x_{t,\mu} = \lfloor n/r \rfloor - |S_i|$ for $a+1 \leq i \leq k$. □

Equitable Connected Partition Given an n-vertex graph $G = (V,E)$ and a positive integer r, Equitable Connected Partition asks whether there is a partition of $V_1, \ldots, V_r$ of V such that $G[V_i]$ is connected and $|V_i| \in \{\lfloor n/r \rfloor, \lceil n/r \rceil\}$ for all i. We call such a partition an *equitable connected r-partition.*

Theorem E.5. Equitable Connected Partition *is fixed-parameter tractable parameterized by* vi.

Proof. Let $(G = (V,E), r)$ be an instance of Equitable Connected Partition, and S be a $\mathsf{vi}(k)$-set of G. Observe that at most k of $V_1, \ldots, V_r$ can intersect S. We split the proof into two cases $r \leq k$ and $r > k$.

Case 1: $r \leq k$. If additionally $\lfloor n/r \rfloor \leq k$ holds in this case, then $n \in O(k^2)$. Thus we assume that $\lfloor n/r \rfloor > k$. This implies that every V_i intersects S. We first guess the partition $S_1, \ldots, S_r$ of S.

Let C_1 and C_2 be components of $G-S$ with the same type, and $\mu_j\colon V(C_j) \to \{1,\dots,r\}$ for each $j \in \{1,2\}$. Then, we say that (C_1,μ_1) and (C_2,μ_2) are *equivalent* if there is an isomorphism η from $G[S\cup C_1]$ and $G[S\cup C_2]$ such that η fixes S and $\mu_1(v) = \mu_2(\eta(v))$ for all $v \in V(C_1)$. A set $\mathcal{M} = \{(C_1,\mu_1),\dots,(C_p,\mu_p)\}$ is *feasible* if C_j is a component of $G-S$ for each j, $\mu_j\colon V(C_j) \to \{1,\dots,r\}$ for each j, and the subgraph of G induced by $S_i \cup \bigcup_{1\le j\le p}\{v \in V(C_j) \mid \mu_j(v) = i\}$ is connected for all $1 \le i \le r$. Let $\mathcal{M}' = \{(C'_1,\mu'_1),\dots,(C'_q,\mu'_q)\}$ be a subset of $\mathcal{M}$ obtained by removing all but one of each equivalent class. It is easy to see that $\mathcal{M}'$ is feasible if and only if so is $\mathcal{M}$. Let t'_j be the (G,S)-type of C'_j. We call the set $\{(t'_1,\mu'_1),\dots,(t'_q,\mu'_q)\}$ a *type-color representation* of $\mathcal{M}$.

Now we guess the type-color representation $\mathcal{T} = \{(t_1,\mu_1),\dots,(t_q,\mu_q)\}$ of a solution. That is, we find a partition such that at least one component of type t_1 is partitioned by μ_1, and no component is partitioned in a way not included in $\mathcal{T}$. The number of candidates depends only on k, and the feasibility of each candidate can be checked in polynomial time.

By a nonnegative variable $x_{t,\mu}$ for $(t,\mu) \in \mathcal{T}$, we represent the number of type-t components that we partition by μ or an equivalent mapping. Since we take at least one such partition of type-t components, we set the constraint $x_{t,\mu} \ge 1$ for each $(t,\mu) \in \mathcal{T}$. Now the connectivity has been handled, and we only need to force the equitable partition. Since each component has to be partitioned, we need the following constraints:

$$\textstyle\sum_{(t,\mu)\in\mathcal{T}} x_{t,\mu} = d_t \quad \text{for each type } t,$$

where d_t is the number of type-t components in $G-S$. The equitable constraints can be expressed as follows:

$$\textstyle\sum_{(t,\mu)\in\mathcal{T}} \mu^{(i)}\cdot x_{t,\mu} = \lceil n/r\rceil - |S_i| \quad \text{for each } i \in \{1,\dots,a\},$$
$$\textstyle\sum_{(t,\mu)\in\mathcal{T}} \mu^{(i)}\cdot x_{t,\mu} = \lfloor n/r\rfloor - |S_i| \quad \text{for each } i \in \{a+1,\dots,r\},$$

where a is the remainder of n/r and $\mu^{(i)}$ is the number of vertices μ maps to i.

Since the number of variables depends only on k, Proposition B.1 implies that the feasibility test of the ILP defined above is fixed-parameter tractable parameterized by k.

Case 2: $r > k$. In this case, some V_i does not intersect S, and thus it is contained in a component of $G-S$. This implies that $\lfloor n/r\rfloor \le k$, and thus $\max_i |V_i| \le k+1$. We first guess the number $k' < k$ of the V_i's intersecting S and the number $a \le k'$ of size $\lceil n/r\rceil$ sets among them. Now we guess V_1: guess at most $k+1$ types of $G-S$; guess the number of components we take from the chosen types, which is at most $k+1$; and for each component, guess the subset of the vertices taken to V_1. The number of candidates depends only on k. In general, when we guess V_i, $2 \le i \le k'$, we first remove the vertices chosen for $\bigcup_{1\le j\le i-1} V_j$ and recompute and redefine the types. Then, we can guess V_i in exactly the same way as the case of $i = 1$. The number of candidates for all $V_1,\dots,V_{k'}$ depends only on k. We reject the guess if some $G[V_i]$ is disconnected.

Let $W = \bigcup_{1 \leq j \leq k'} V_j$. Now it suffices to decide whether $G - W$ has an equitable connected $(r-k')$-partition. For each component C of $G-W$, we enumerate all the possible pairs (p, q) of nonnegative integers such that C admits an equitable connected $(p+q)$-partition such that p parts have size $\lceil n/r \rceil$ and q parts have size $\lfloor n/r \rfloor$. This can be done in FPT time parameterized by k in total, since each component has at most k vertices and the number of components in $G - W$ is at most $|V|$. We now check whether by picking one pair (p, q) for each component, it is possible to make the total number of components $r - k'$. This can be done in polynomial time by a standard dynamic programming algorithm since the number of components and $r - k'$ are at most $|V|$. □

F Hard problems parameterized by vi

F.1 Graph Motif

Given a graph $G = (V, E)$, a vertex coloring $c\colon V \to \mathcal{C}$, and a multiset M of colors in $\mathcal{C}$, the problem Graph Motif is to decide if there is a set $S \subseteq V$ such that $G[S]$ is connected and $c(S) = M$, where $c(S)$ is the multiset of colors appearing in S. If the motif M is a set (i.e., no element in M has multiplicity more than 1), then the restricted problem is called Colorful Graph Motif. It is known that Graph Motif is fixed-parameter tractable parameterized by vc [14] (actually by more general parameters *neighborhood diversity* [32] and *twin-cover number* [31]). The proof of Theorem 20 in [14] implies that Colorful Graph Motif is NP-complete for graphs of vi = 6. By a similar proof, we will show that Colorful Graph Motif is NP-complete for graphs of vi = 4. We then complement this by showing that Graph Motif is polynomial-time solvable for graphs of vi ≤ 3.

Theorem F.1. Colorful Graph Motif *is NP-complete on trees of vertex integrity* 4.

Proof. The problem is clearly in NP. We present a reduction from an NP-complete problem 3-Dimensional Matching [34]. The input of 3-Dimensional Matching consists of three disjoint sets $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, $Z = \{z_1, \ldots, z_n\}$, and a set of triples $T \subseteq X \times Y \times Z$. The task is to decide whether there is a subset S of T such that $|S| = n$ and each element of $X \cup Y \cup Z$ appears in a triple included in S.

We construct a graph G with a coloring c as follows. The graph G contains a special root vertex r with unique color $c(r) = r$. For each triple $t = (x_i, y_j, z_k) \in T$, take three new vertices t_i, t_j, t_k with $c(t_i) = x_i$, $c(t_j) = y_j$, and $c(t_k) = z_k$, and add three new edges $\{r, t_i\}$, $\{t_i, t_j\}$, and $\{t_j, t_k\}$. We set $M = \{r\} \cup X \cup Y \cup Z$. This completes the construction. Note that G is a tree and $\{r\}$ is a vi(4)-set of G as each component of $G - \{r\}$ is a path of order 3.

Assume that (X, Y, Z, T) is a yes instance of 3-Dimensional Matching with $S \subseteq T$ as a certificate. We set $L = \{r\} \cup \{t_i, t_j, t_k \mid t = (x_i, y_j, z_k) \in S\}$. Clearly, $c(L) = M$. Since $G[L]$ is connected, (G, M) is a yes instance of Colorful Graph Motif.

To show the other direction, assume that a vertex subset L of G induces a connected graph and $c(L) = M$. Observe that L has to include r. Since $X \cup Y \cup Z = c(L) \setminus \{r\}$, L includes exactly n vertices of distance i from r for each $i \in \{1, 2, 3\}$. This fact and the connectivity of $G[L]$ imply that for each $t = (x_i, y_j, z_k) \in T$, L contains either all vertices t_i, t_j, t_k or none of them. Let $S \subseteq T$ be the set of triples such that L contains all three vertices corresponding to each $t \in S$. By the discussion above, $|S| = n$ and each element of $X \cup Y \cup Z$ appears in a triple included in S. □

Theorem F.2. Graph Motif *can be solved in polynomial time on graphs of vertex integrity at most* 3.

Proof. Let $G = (V, E)$ be the input graph with a coloring $c\colon V \to \mathcal{C}$ and M be the input multiset of colors. Let R be a $\mathsf{vi}(3)$-set of G. If $|R| \geq 2$, then R is a vertex cover of G with $|R| \leq 3$, and thus we can apply an FPT algorithm parameterized by the vertex cover number [31,14]. If $R = \emptyset$, then each connected component of G is of order at most 3, and thus the problem is trivial. In the following we assume that $R = \{r\}$ for some $r \in V$. Furthermore, we assume that r is included in the solution S as otherwise $|S| \leq 2$. Let D_i be the vertices of distance i from r. Note that $V = \{r\} \cup D_1 \cup D_2$.

We construct an auxiliary bipartite multi-graph H as follows. For each color $x \in \mathcal{C}$, take new vertices x_1 and x_2. For each component C of $G - r$, if C has two vertices u of color x and v of color y, where only u is adjacent to r, then add one edge between x_1 and y_2. For each color $x \in \mathcal{C}$, we define degree constraints of x_1 and x_2 in H as follows: the degree constraint of x_1 is "at most $M(x)$", and the degree constraint of x_2 is "exactly $\max\{M(x) - q(x), 0\}$", where $M(x)$ is the multiplicity of x in $M \setminus \{c(r)\}$ and $q(x)$ is the number of color-x vertices in D_1. We will show that H has a subgraph F satisfying the degree constraints if and only if there is a set $S \subseteq V$ such that $G[S]$ is connected and $c(S) = M$. Since finding a subgraph of such degree constraints can be done in polynomial time [30], this equivalence implies the theorem.

First assume that there is a set $S \subseteq V$ such that $G[S]$ is connected and $c(S) = M$. We choose S among such sets so that $|S \cap D_1|$ is maximized. This implies in particular that if there is a vertex $v \in D_1 \setminus S$ of color x, then no vertex of color x in D_2 belongs to S. For each edge $\{u, v\}$ in $G[S - r]$, if $u \in D_1$, $v \in D_2$, $c(u) = x$, and $c(v) = y$, then add one edge between x_1 and y_2 into F. Now for each color x, the degree of x_1 in F is at most $M(x)$. Since S takes color-x vertices in D_2 only when it is necessary after including all color-x vertices in D_1, the degree of x_2 in F is exactly $\max\{M(x) - q(x), 0\}$.

Next assume that H has a subgraph F that satisfies the degree constraints. For each edge between x_1 and y_2 in F, we add into S the endpoints of an arbitrary edge $\{u, v\}$ in $G - r$ such that $c(u) = x$, $u \in D_1$, $c(v) = y$, and $v \in D_2$. Let $\mathcal{S} = c(S)$, the multiset of colors appear in S. From the construction of S, it holds for each color x that $\mathcal{S}(x) = \deg_F(x_1) + \deg_F(x_2) = \deg_F(x_1) + \max\{M(x) - q(x), 0\}$. If $M(x) \leq q(x)$, then $\mathcal{S}(x) = \deg_F(x_1) \leq M(x)$. We add, into S, arbitrary $M(x) - \mathcal{S}(x)$ of color-x vertices in $D_1 \setminus S$. This is possible

since the number of color-x vertices in $D_1 \setminus S$ is $q(x) - \mathcal{S}(x) \geq M(x) - \mathcal{S}(x)$. If $M(x) > q(x)$, then $\mathcal{S}(x) = \deg_F(x_1) + M(x) - q(x)$. In this case, we add all color-x vertices in D_1 into S, and then the multiplicity of x in the resultant set becomes $q(x) + M(x) - q(x) = M(x)$. □

F.2 Steiner Forest

Steiner Forest is a generalization of Steiner Tree and defined as follows: Given a graph $G = (V, E)$ with edge weighting $w\colon E \to \mathbb{Z}^+$, a positive integer k, and disjoint terminal sets $T_1, \ldots, T_t \subseteq V$ with $|T_i| \geq 2$ for all i, decide whether there is a subgraph F of G with $\sum_{e \in F} w(e) \leq k$ such that each T_i is contained in some connected component of F. Note that we can assume that F is a forest.

It is known that Steiner Forest is strongly NP-complete (that is, NP-complete even if the weights are given in unary) on graphs of vertex integrity 5 [35]. We show that for graphs of small vertex cover number, the problem becomes easier.

Let $G = (V, E)$ be a graph, F a subgraph of G, and S a vertex cover of G. We assume without loss of generality that F is a forest. The following observations follow from this assumption and the fact that $V - S$ is an independent set.

Observation F.3 *At most $|S| - 1$ vertices in $V - S$ have degree 2 or more in F.*

Observation F.4 *F has at most $|S|$ connected components.*

Theorem F.5. Steiner Forest *can be solved in time $n^{O(\mathsf{vc})}$, on n-vertex graphs of vertex cover number at most* vc.

Proof. Let $G = (V, E)$ be a graph and $S \subseteq V$ be a vertex cover of G. We first guess the set $D \subseteq V - S$ of vertices that have degree at least 2 in F. By Observation F.3, we know that $|D| \leq |S| - 1$. The vertices in $V - (S \cup D)$ have degree at most 1 in F: if such a vertex appears in some T_i, then it has degree 1 in F; otherwise, it has degree 0 in F, and thus can be safely removed from the graph. We then guess the edges in $F[S \cup D]$. The number of candidates for D is at most $n^{|S|}$, and the number of candidates for the edge set is at most $|S|^{2|S|}$.

We reject the guess $F[S \cup D]$ if there are two components of $F[S \cup D]$ that contain elements of T_i for some i. Now for each i, there is at most one component C_i of $F[S \cup D]$ such that $C \cap T_i \neq \emptyset$. If there is no such component, we guess one component of $F[S \cup D]$ from $O(|S|)$ candidates and call it C_i. Note that C_i and C_j may be the same for $i \neq j$. Now for each i and for each vertex $u \in T_i \setminus (S \cup D)$, we find an edge $\{u, v\}$ of the minimum weight such that $v \in V(C_i)$ and add $\{u, v\}$ into F. We output a minimum weight forest obtained in this way. □

We denote by Unweighted Steiner Forest the special case of Steiner Forest such that each edge has weight 1. By subdividing the edges in the proof in [35], we can show that Unweighted Steiner Forest is NP-complete for graphs of $\mathsf{tw} = 3$.

Theorem F.6. Unweighted Steiner Forest *is fixed-parameter tractable parameterized by* vc.

Proof. Let $G = (V, E)$ be the input graph and S be a vertex cover of G. Let $s = |S|$. We reduce the instance by applying the following reduction rules exhaustively.

1. If T_i contains s or more vertices v in $V - S$ that have the same neighborhood $N(v)$, then remove one of them from T_i and decrease k by 1.
2. Let $T_{i(1)}, \dots T_{i(s+1)}$ be $s+1$ distinct terminal sets such that $T_{i(j)} \cap S = \emptyset$ for all j and for each $X \subseteq S$ and $j \neq j'$, it holds that $|\{v \in T_{i(j)} \mid N(v) = X\}| = |\{v \in T_{i(j')} \mid N(v) = X\}|$. Then replace $T_{i(1)}$ and $T_{i(2)}$ with their union $T_{i(1)} \cup T_{i(2)}$.
3. If there are two non-terminal vertices of the same neighborhood, then remove one of them from the graph.

The first rule is safe by Observation F.3. At least one of the s vertices of the same neighborhood is a leaf. Since there are other vertices of the same neighborhood in T_i, this leaf can be joined to the component containing the other vertices of T_i with an edge. The safeness of the second rule follows by Observation F.4, as at least two of them belong to the same connected component. The third rule is safe because at most one of them is used in an optimal solution.

We can see that if no reduction rule above applies, then the numbers of vertices and of terminal sets depend only on s. By the first rule, each terminal set T_i contains at most $s \cdot 2^s$ vertices. By the first and second rules, there are at most $s \cdot s^{2^s} + s$ terminal sets. By Reduction rule 3, there are at most $2^s + s$ non-terminal vertices. □

G Easy problems parameterized by td

Here we show the following.

Observation G.1 List Hamiltonian Path, Directed (p, q)-Edge Dominating Set, *and* Metric Dimension *are fixed-parameter tractable parameterized by* td.

List Hamiltonian Path is a generalization of Hamiltonian Path such that each vertex has a set of permitted positions where it can be put in a Hamiltonian path. This problem is W[1]-hard parameterized by pw [43]. By Proposition A.1, this problem parameterized by td admits a trivial FPT algorithm: if a graph G has at least $2^{\mathsf{td}(G)}$ vertices, then G does not have a Hamiltonian path; otherwise, we can try all $n! \leq (2^{\mathsf{td}(G)})!$ permutations of vertices.

For integers $p, q \geq 0$, the edges (p, q)-*dominated* by an arc $e = (u, v)$ are e itself and all arcs that are on a directed path of length at most p to u or on a directed path of length at most q from v. Then, Directed (p, q)-Edge Dominating Set asks whether there exists a set K of arcs with $|K| \leq k$ such that every arc is (p, q)-dominated by K. This problem is W[1]-hard parameterized by

pw but fixed-parameter tractable parameterized by $\mathsf{tw} + p + q$ [7]. Since we can assume that p and q are smaller than the longest path length, we can also assume that they are bounded by a function of td. Thus the fixed-parameter tractability with $\mathsf{tw} + p + q$ implies the fixed-parameter tractability solely with td. (Here the parameters are defined on the undirected graph obtained by ignoring the directions of arcs.)

The argument for Metric Dimension is slightly more involved. In Metric Dimension, we are given a graph $G = (V, E)$ and an integer k and asked whether there exists $S \subseteq V$ such that $|S| \leq k$ and for each pair $u, v \in V$ there exists $w \in S$ with $\mathrm{dist}(u, w) \neq \mathrm{dist}(v, w)$, where $\mathrm{dist}(\cdot, \cdot)$ is the distance between two vertices in G. We call such a set S a *resolving set*. Recently, this problem is shown to be W[1]-hard parameterized by pw [13]. Observe that there is an MSO_1 formula $\varphi(S)$ such that its length depends only on the diameter of the underlying graph and it is evaluated to be `true` if and only if the vertex set S is a resolving set of the underlying graph. It is known that finding a minimum vertex set S satisfying $\varphi(S)$ is fixed-parameter tractable parameterized by clique-width + $|\varphi|$ [18,48]. Therefore, Metric Dimension is fixed-parameter tractable parameterized by clique-width+the diameter. Since the clique-width and the diameter are bounded from above by functions of its treedepth, Metric Dimension is fixed-parameter tractable parameterized by td. (See [16] for an upper bound of clique-width.)