A Proof of Lemma 2

Using induction, one can see that for each $t \in V(T)$, $\mathcal{R}_t$ is the set of every element of $\mathcal{I}_t$ such that, with $Y_t = X_t \setminus \text{forg}(t)$, there exists $(\widehat{S}_1, \ldots, \widehat{S}_r) \in V(G_t)^r$, that satisfies:

- for each $i \in [1, r]$, $\widehat{S}_i$ is a vertex cover of G_t,
- for each $i \in [1, r]$, $\widehat{S}_i \cap X_t = S_i$,
- for each $i \in [1, r]$, $|\widehat{S}_i \setminus Y_t| = s_i$, and
- $\min(d, \text{Div}(\widehat{S}_1 \setminus Y_t, \ldots, \widehat{S}_r \setminus Y_t)) = \ell$.

As the root q of the tree decomposition $\mathcal{D}$ is such that $X_q = \emptyset$, we obtain that the element in $\mathcal{R}_q$ are the elements $((\emptyset, s_1), \ldots, (\emptyset, s_r), \ell)$ of $\mathcal{I}_q$ such that there exists $(\widehat{S}_1, \ldots, \widehat{S}_r) \in V(G)^r$, that satisfy,

- for each $i \in [1, r]$, $\widehat{S}_i$ is a vertex cover of G_t,
- for each $i \in [1, r]$, $|\widehat{S}_i| = s_i \leq k$, and
- $\min(d, \text{Div}(\widehat{S}_1, \ldots, \widehat{S}_r)) = \ell$.

As such $(\widehat{S}_1, \ldots, \widehat{S}_r) \in V(G)^r$ is a solution of DIVERSE VERTEX COVER if and only if $\ell \geq d$, the lemma follows.

B Proof of Theorem 3

Let us analyze the time needed to compute $\mathcal{R}_q$. We have that, for each $t \in V(\mathcal{D})$, $|\mathcal{I}_t| \leq (2^{\cdot|X_t|} \cdot (k+1))^r \cdot (d+1)$. Note that given $I_1, \ldots, I_{\delta(t)}$ be elements of $\mathcal{R}_{t_1}, \ldots, \mathcal{R}_{t_{\delta(t)}}$, there are at most $2^{|\text{new}(t)| \cdot r} \leq 2^r$ way to create an element I of $\mathcal{R}_t$ by selecting, or not the (potential) new element of X_t for each set S_i, $i \in [1, r]$. The remaining is indeed fixed by $I_1, \ldots, I_{\delta(t)}$. Thus, $\mathcal{R}_t$ can be computed in time $\mathcal{O}(r \cdot |X_t| \cdot 2^r \cdot \prod_{i=1}^{\delta(t)} |\mathcal{R}_{t_i}|)$, where the factor $r \cdot |X_t|$ appears when verifying that the element we construct satisfy $\forall j \in [1, r]\,, E(G[X_j \setminus S_j]) = \emptyset$. As we need to compute $\mathcal{R}_t$ for each $t \in V(\mathcal{D})$ and that $|V(\mathcal{D})| = \mathcal{O}(n)$ and we can assume that $\delta(t) \leq 2$ for each $t \in V(\mathcal{D})$, the theorem follows.

C Extra problem definitions

In the d-HITTING SET problem, we are given a hypergraph H, each of whose hyperedges contains at most d elements, and an integer k, and the goal is to find a set $S \subseteq V(H)$ of vertices of H of size at most k such that each hyperedge contains at least one element from S. In the POINT LINE COVER problem, we are given a set of points in the plane and an integer k and we want to find a set of at most k lines such that each point lies on at least one of the lines. A directed graph D is called a *tournament*, if for each pair of vertices $u, v \in E(D)$, either the edge directed from u to v or the edge directed from v to u is contained in the set of arcs of D. In the FEEDBACK ARC SET IN TOURNAMENTS problem we are given a tournament and an integer k and the goal is to find a set of at most k arcs such that after removing this set, the resulting directed graph does not contain any directed cycles.

D Proof of Corollary 15

(i)[3] The classical kernelization for VERTEX COVER due to [Buss and Goldsmith, 1993] consists of the following two reduction rules. Let (G, k) be an instance of VERTEX COVER. First, we remove isolated vertices from G; since they do not cover any edges of the graph, we do not need them to construct a vertex cover. To obtain the loss-less kernel, we put these vertices into the set A. Second, if there is a vertex of degree more than k, this vertex has to be included in any solution; otherwise we would have to include its more than k neighbors, resulting in a vertex cover that exceeds the size bound. We add this vertex to F, remove it from G and decrease the parameter value by 1. This second reduction rule finishes the description of the kernel. It is not difficult to argue that after an exhaustive application of these two rules, the resulting kernelized instance (G', k') is such that either $k' < 0$, in which case we are dealing with a NO-instance, or $|V(G')| = \mathcal{O}(k^2)$. For the domain recovery algorithm, we can use a trivial algorithm that adds some isolated vertices to the graph G'.

We now argue that this is indeed a loss-less kernel. Consider Definition 13. Item (iv) follows immediately from the fact that each vertex cover of G of size at most k has to contain all vertices in F and that each vertex in $V(G) \setminus F$ is isolated. (ii) follows from the fact that A is a set of isolated vertices in $G - F$, and (iii) from the fact that for any $A' \subseteq A$ and $a \in A'$, a is an isolated vertex in the graph of the instance $(G', k') \hookleftarrow_{A'} (G, k)$. The result now follows from Theorem 14.

(ii) We show that the kernel on $\mathcal{O}(k^d)$ vertices presented in [Cygan *et al.*, 2015, Section 2.6.1] is a loss-less kernel. This kernel is essentially a generalization of the one presented in the proof of (i), so we will skip some of the details. It is based on the following reduction rule: If there are $k+1$ hyperedges $e_1, \ldots, e_{k+1}$ with $Y := \bigcap_{i=1}^{k+1} e_i$ such that for each $i \in [k+1]$, $e_i \setminus Y \neq \emptyset$, then any solution has to contain Y; otherwise, to hit the hyperedges $e_1, \ldots, e_{k+1}$, we would have to include at least $k+1$ elements in the hitting set. Moreover, if $Y = \emptyset$, we can immediately conclude that we are dealing with a NO-instance. If Y is nonempty, then we add all elements of Y to F and decrease the parameter value by $|Y|$. The set A consists of all vertices that are isolated (i.e. not contained in any hyperedge) after exhaustively applying the previous reduction rule. Following the same argumentation above (and using the same domain recovery algorithm), we can conclude that this procedure is a loss-less kernel on $\mathcal{O}(k^d)$ vertices, and the result follows from Theorem 14.

(iii) Let (P, k) be an instance of POINT LINE COVER. We consider the set of the lines defined by all pairs of points of P as the domain of (P, k), and we denote this set by $L(P)$. All solutions to (P, k) can be considered a subset of $L(P)$. We obtain a kernel on $\mathcal{O}(k^2)$ points as follows, cf. [Cygan *et al.*, 2015, Exercise 2.4]. The idea is again similar to the kernel presented in (i). If there are $k+1$ points on a line, then we have to include this line in any solution; we add such lines to the set F and remove all points on them from P, and decrease the parameter value by 1. We finally add to A all lines that have no points on them. We can argue in the same way as

[3] This was also observed in [Carbonnel and Hebrard, 2016].

above that this gives a kernel with at most $\mathcal{O}(k^2)$ points and with Theorem 14, the result follows.

(iv) We observe that the kernel given in [Cygan *et al.*, 2015, Section 2.2.2] is a loss-less kernel. Its first reduction rule states that if there is an arc that is contained in at least $k+1$ triangles, then we reverse this arc and decrease the parameter value by 1, and the second reduction rule states that any vertex that is not contained in a triangle can be removed. Any arc affected by the former rule will be put in the set F and any arc affected by the latter rule will be put in the set A. We now describe the domain recovery algorithm. Let (T, k) be the original instance and (T', k') the kernelized instance, and let $(u, v) = a \in A$ be an arc. Then, we add a to T' and to ensure that the resulting directed graph is a tournament, for any $x \in \{u, v\} \setminus V(T')$, we add all arcs $(x, y) \in E(T)$ and $(y, x) \in E(T)$ to T'. Since $a \in A$, we know that one of its endpoints was not contained in any triangle, and hence adding the endpoints of a and all their incident arcs does not add any triangles to the tournament.