Diversity of Solutions: An Exploration Through the Lens of Fixed-Parameter Tractability Theory

Julien Baste[1] , **Michael R. Fellows**[2] , **Lars Jaffke**[2] , **Tomáš Masařík**[3,4] ,
Mateus de Oliveira Oliveira[2] , **Geevarghese Philip**[5,6] , **Frances A. Rosamond**[2]
[1]Ulm University, Germany
[2]University of Bergen, Norway
[3]University of Warsaw, Poland
[4]Charles University, Czech Republic
[5]Chennai Mathematical Institute, India
[6]UMI ReLaX
julien.baste@uni-ulm.de, {michael.fellows, lars.jaffke, mateus.oliveira, frances.rosamond}@uib.no,
masarik@kam.mff.cuni.cz, gphilip@cmi.ac.in

Abstract

When modeling an application of practical relevance as an instance of a combinatorial problem X, we are often interested not merely in finding *one* optimal solution for that instance, but in finding a *sufficiently diverse* collection of good solutions. In this work we initiate a systematic study of *diversity* from the point of view of fixed-parameter tractability theory. We consider an intuitive notion of *diversity* of a collection of solutions which suits a large variety of combinatorial problems of practical interest. Our main contribution is an algorithmic framework which –*automatically*– converts a tree-decomposition-based dynamic programming algorithm for a given combinatorial problem X into a dynamic programming algorithm for the diverse version of X. Surprisingly, our algorithm has a polynomial dependence on the diversity parameter.

1 Introduction

In a typical combinatorial optimization problem we are given a large space of potential solutions and an objective function. The task is to find a solution which maximizes or minimizes the objective function. In many situations of practical relevance, however, it does not really help to get just *one optimal* solution; it would be much better to have a small, but *sufficiently diverse* collection of *sufficiently good* solutions. Given such a small list of good solutions we can select one which is best for our purpose, perhaps by taking into account external factors—such as aesthetical, political, or environmental—which are difficult or even impossible to formalize. An early, illustrative example is the problem of generating floor plans for evaluation by an architect [Galle, 1989].

Solution diversity is already a fundamental concept in many computational tasks. Take for instance a web search. Here, we do not want to find *the one* website that 'optimally fits' the search term, neither a ranking of a small number of 'best fits', but what is desirable is a *diverse set* of websites that fit the search term *reasonably well*.

This notion has also been applied to solution sets of various types of combinatorial problems. For instance, the works [Glover *et al.*, 2000] and [Hebrard *et al.*, 2005; Hebrard *et al.*, 2007] seek to find solution sets to mixed integer programming problems and constraint satisfaction problems, respectively, that are diverse. In other words, the solutions are far apart from each other in some mathematical notion of distance. We refer to [Petit and Trapp, 2019] for a timely overview of the subject.

From a complexity-theoretic perspective, there are two immediate barriers to this approach. The first is that most combinatorial problems are already NP-hard when asking only for a single solution. The second is that the very basic MAXIMUM DIVERSITY problem which given a set of n elements in a metric space and an integer $k < n$, asks for a size-k subset of the elements such that the sum of the pairwise distances is maximized, is NP-hard as well [Kuo *et al.*, 1993]. The theory of *fixed-parameter tractability* [Downey and Fellows, 2013] provides a powerful framework to overcome these barriers. The key goal is to identify a secondary numerical measure of the inputs to an (NP-hard) computational problem, called the *parameter*, and to provide algorithms in whose runtime the combinatorial explosion is restricted to the parameter k. More formally, a problem is *fixed-parameter tractable (FPT)*, if it can be solved in time $f(k) \cdot n^c$, where f is a computable function, n the input size and c a fixed constant. On instances where the parameter value is relatively small, FPT-algorithms are efficient. In an application context, we are naturally concerned with finding *small* diverse sets of solutions, since the aim is to provide the user with a few alternatives that can then be compared manually. Therefore, the number of requested solutions is an ideal candidate for parameterization.

In this work, we propose to study the notion of solution diversity from the perspective of fixed-parameter tractability theory. We demonstrate the theoretical feasibility of this paradigm by showing that diverse variants of a large class of parameterized problems admit FPT-algorithms. Specifi-

cally, we consider *vertex-problems* on graphs, which are sets of pairs (G, S) of a graph G and a subset S of its vertices that satisfies some property. For instance, in the VERTEX COVER problem, we require the set S to be a vertex cover of G (i.e. S has to contain at least one endpoint of each edge of G). One consequence of our main result which we discuss below in more detail is that the diverse variant of VERTEX COVER, asking for r solutions, is FPT when parameterized by solution size plus r.

Before we proceed, we would like to point out promising future applications of the *Diverse FPT* paradigm in AI. The VERTEX COVER problem itself naturally models conflict-resolution: the entities are the vertices of the graph, and a conflict is represented by an edge. Now, a vertex cover of the resulting graph is a set of entities whose removal makes the model conflict-free. An example of a potential use of DIVERSE VERTEX COVER in a planning scenario is given in [Baste *et al.*, 2019b]. In general, in planning and scheduling problems, a large amount of *side information* is lost or intentionally omitted in the modelling process. Some side information could make the model too complex to be solved, and other information may even be impossible to model. Offering the user a small number of good solutions to a more easily computable 'base model', among which they can hand pick their favorite solution, is a feasible alternative.

A Formal Notion of Diversity. We choose a very natural and general measure as our notion of diversity among solutions. Given two subsets S and S' of a set V the *Hamming distance* between S and S' is the number

$$\mathrm{HamDist}(S, S') = |S \backslash S'| + |S' \backslash S|.$$

We define the *diversity of a list* $S_1, \ldots, S_r$ of subsets of V to be

$$\mathrm{Div}(S_1, \ldots, S_r) = \sum_{1 \leq i < j \leq r} \mathrm{HamDist}(S_i, S_j).$$

We can now define the diverse version of vertex-problems:

Definition 1 (Diverse Problem)**.** *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex-problems, and let $d \in \mathbb{N}$. We let*

$$\mathrm{Div}^d(\mathcal{P}_1, \ldots, \mathcal{P}_r) = \{(G, X_1, \ldots, X_r) \mid (G, X_i) \in \mathcal{P}_i, \mathrm{Div}(X_1, \ldots, X_r) \geq d\}.$$

Intuitively, given vertex-problems $\mathcal{P}_1, \ldots, \mathcal{P}_r$ and a graph G, we want to find subsets $S_1, \ldots, S_r$ of vertices of G such that for each $i \in \{1, \ldots, r\}$, S_i is a solution for problem $\mathcal{P}_i$ on input G, and such that the list $S_1, \ldots, S_r$ has diversity at least d. If all vertex-problems $\mathcal{P}_1, \ldots, \mathcal{P}_r$ are the same problem $\mathcal{P}$, then we write $\mathrm{Div}^d_r(\mathcal{P})$ as a shortcut to $\mathrm{Div}^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$.

Diversity and Dynamic Programming. The treewidth of a graph is a structural parameter that quantifies how close the graph is to being a forest (i.e., a graph without cycles). The popularity of this parameter stems from the fact that many problems that are NP-complete on general graphs can be solved in polynomial time on graphs of constant treewidth. In particular, a celebrated theorem due to Courcelle [Courcelle, 1990] states that any problem expressible in the monadic second-order logic of graphs can be solved in polynomial time on graphs of constant treewidth. Besides this metatheorem, the notion of treewidth has found applications in several branches of Artificial Intelligence such as Answer Set Programs [Bliem *et al.*, 2017], checking the consistency of certain relational algebras in Qualitative Spacial Reasoning [Bodirsky and Wölfl, 2011], compiling Bayesian networks [Chavira and Darwiche, 2007], determining the winners of multiwinner voting systems [Yang and Wang, 2018], analyzing the dynamics of stochastic social networks [Barrett *et al.*, 2007], and solving constraint satisfaction problems [Jégou *et al.*, 2007]. A large number of these algorithms are in fact FPT-algorithms when treewidth is the parameter, i.e. they run in time $f(t) \cdot n^c$, where f is a computable function, t the treewidth of the input graph, n the number of its vertices, and c some fixed constant. Typically, such algorithms are dynamic programming algorithms which operate on a tree-decomposition in a bottom-up fashion by computing data from the leaves to the root.

Dynamic Programming Core Model. We introduce a formalism for dynamic programming based on a tree decomposition, which we call the *Dynamic Programming Core* model. This notion captures a large variety of dynamic programming algorithms on tree decompositions. We use the model to derive our main result (Theorem 10) which is a framework to efficiently—and automatically—transform treewidth-based dynamic programming algorithms for vertex-problems into algorithms for the diverse versions of these problems. More precisely, we show that if $\mathcal{P}_1, \ldots, \mathcal{P}_r$ are vertex-problems where for each $i \in \{1, \ldots, r\}$, $\mathcal{P}_i$ can be solved in time $f_i(t) \cdot n^{\mathcal{O}(1)}$, then $\mathrm{Div}^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ can be solved in time $(\prod_{i=1}^{r} f_i(t)) \cdot n^{\mathcal{O}(1)}$. In particular, if a vertex-problem $\mathcal{P}$ can be solved in time $f(t) \cdot n^{\mathcal{O}(1)}$, then its diverse version $\mathrm{Div}^d_r(\mathcal{P})$ can be solved in time $f(t)^r \cdot n^{\mathcal{O}(1)}$. The surprising aspect of this result is that the running time depends only *polynomially* on d (which is at most $r^2 n$), while a naïve dynamic programming algorithm would have a runtime of $d^{\mathcal{O}(r^2)} \cdot f(t)^r \cdot n^{\mathcal{O}(1)}$.

Discussion of the Diversity Measure. Various measures of diversity have been used, studied, and compared in different areas of computer science. We choose the *sum* of the Hamming distances over all pairs of elements for this work. This measure is commonly used for population diversity in genetic algorithms [Gabor *et al.*, 2018; Wineberg and Oppacher, 2003]. Nonetheless, we would like to point out that it has some weaknesses. For instance, taking many copies of two disjoint solutions yields a relatively high diversity value, and such a solution set is not 'diverse' from an intuitive point of view. We refer to [Baste *et al.*, 2019b] for a more detailed discussion. Another natural measure using the Hamming distance is the *minimum* Hamming distance over all pairs in a set, as it is done e.g. in [Hebrard *et al.*, 2005; Hebrard *et al.*, 2007]. We would like to point out that a straightforward adaptation of our algorithmic framework would result in a running time of $d^{\mathcal{O}(r^2)} \cdot f(t)^r \cdot n^{\mathcal{O}(1)}$, where d is the diversity, r the number of solutions, and t the treewidth. This remains FPT only when the diversity d is an additional component of the parameter, or when d is naturally upper

bounded by t and r. Consider for instance DIVERSE VERTEX COVER, asking for vertex covers of size at most k. In any nontrivial instance, t is at most k, and the Hamming distance between two solutions is at most $2k$, therefore we may assume that $d \leq 2k$. This implies that DIVERSE VERTEX COVER can be solved in time $2^{\mathcal{O}(r^2 \log k)+kr} \cdot n^{\mathcal{O}(1)}$ using the minimum Hamming distance as a diversity measure.

Related Work. The above mentioned MAXIMUM DIVERSITY problem has applications in the generation of diverse query results, see e.g. [Gollapudi and Sharma, 2009; Abbassi *et al.*, 2013]. Besides mixed integer programming [Glover *et al.*, 2000; Danna and Woodruff, 2009; Petit and Trapp, 2015], binary integer linear programming [Greistorfer *et al.*, 2008; Trapp and Konrad, 2015] and constraint programming [Hebrard *et al.*, 2005; Hebrard *et al.*, 2007], diverse solution sets have been considered in SAT solving [Nadel, 2011], recommender systems [Adomavicius and Kwon, 2014], routing problems [Schittekat and Sörensen, 2009], answer set programming [Eiter *et al.*, 2013], and decision support systems [Løkketangen and Woodruff, 2005; Hadžić *et al.*, 2009].

Several details that have been omitted due to space restrictions can be found in the full version [Baste *et al.*, 2019a].

2 Preliminaries

For positive integers a, b ; $a < b$ we use $[a, b]$ to denote the set $\{a, a + 1, \ldots, b\}$. We use $V(G)$ and $E(G)$, respectively, to denote the vertex and edge sets of a graph G. For a tree T rooted at q we use T_t to denote the subtree of T rooted at a vertex $t \in V(T)$. A *rooted tree decomposition* of a graph G is a tuple $\mathcal{D} = (T, q, \mathcal{X})$, where T is a tree rooted at $q \in V(T)$ and $\mathcal{X} = \{X_t \mid t \in V(T)\}$ is a collection of subsets of $V(G)$ such that:

- $\bigcup_{t \in V(T)} X_t = V(G)$,
- for every edge $\{u, v\} \in E(G)$, there is a $t \in V(T)$ such that $\{u, v\} \subseteq X_t$, and
- for each $\{x, y, z\} \subseteq V(T)$ such that z lies on the unique path between x and y in T, $X_x \cap X_y \subseteq X_z$.

We say that the vertices of T are the *nodes* of $\mathcal{D}$ and that the sets in $\mathcal{X}$ are the *bags* of $\mathcal{D}$. Given a node $t \in V(T)$, we denote by G_t the subgraph of G induced by the set of vertices $\bigcup_{s \in V(T_t)} X_s$. The *width* of a tree decomposition $\mathcal{D} = (T, q, \mathcal{X})$ is defined as $\max_{t \in V(T)} |X_t| - 1$. The *treewidth* of a graph G, denoted by $\mathsf{tw}(G)$, is the smallest integer w such that there exists a rooted tree decomposition of G of width at most w. The *rooted path decomposition* of a graph is a rooted tree decomposition $\mathcal{D} = (T, q, \mathcal{X})$ such that T is a path and q is a vertex of degree 1. The *pathwidth* of a graph G, denoted by $\mathsf{pw}(G)$, is the smallest integer w such that there exists a rooted path decomposition of G of width at most w. Note that in a rooted path decomposition, every node has at most one child.

For convenience we will always assume that the bag associated to the root of a rooted tree decomposition is empty. For a node $t \in V(T)$ we use $\delta_{\mathcal{D}}(t)$, or $\delta(t)$ when $\mathcal{D}$ is clear from the context, to denote the number of children of t in the tree T. For nodes t and t' of $V(T)$ where t' is the parent of t we use $\mathrm{forg}(t) = X_t \setminus X_{t'}$ to denote the set of vertices of G which are *forgotten* at t. By convention, for the root q of T, we let $\mathrm{forg}(q) = \emptyset$. For $t \in V(T)$ we denote by $\mathrm{new}(t)$ the set $X_t \setminus \bigcup_{i=1}^{\delta(t)} X_{t_i}$ where $t_1, \ldots, t_{\delta(t)}$ are the children of t. Given a rooted tree decomposition $\mathcal{D}$ of a graph G one can obtain, in linear time, a tree decomposition $(T, q, \mathcal{X})$ of G of the same width as $\mathcal{D}$ such that for each $t \in V(T)$, $\delta(t) \leq 2$ and $|\mathrm{new}(t)| \leq 1$ [Cygan *et al.*, 2015]. From now on we assume that every rooted tree decomposition is of this kind.

3 A First Example: Diverse Vertex Cover

The main result of this paper is a general framework to automatically translate tree-decomposition-based dynamic programming algorithms for vertex-problems into algorithms for the diverse versions of these problems. We develop this framework in Section 4. In this section we illustrate the main techniques used in this conversion process by showing how to translate a tree-decomposition-based dynamic programming algorithm for the VERTEX COVER problem into an algorithm for its diverse version DIVERSE VERTEX COVER. Given a graph G and three integers k, r, and d, the DIVERSE VERTEX COVER problem asks whether one can find r vertex covers in G, each of size at most k, such that their diversity is at least d. Our algorithm for this problem runs in $2^{\mathcal{O}(kr)}|V(G)|$ time.

3.1 Incremental Computation of Diversity

Recall that we defined the diversity of a list $S_1, S_2, \ldots, S_r$ of subsets of a set V to be

$$\mathrm{Div}(S_1, \ldots, S_r) = \sum_{1 \leq i < j \leq r} \mathrm{HamDist}(S_i, S_j).$$

We will now describe a way to compute the diversity $\mathrm{Div}(S_1, \ldots, S_r)$ in an incremental fashion, by incorporating the influence of each element of V in turn. For each element $v \in V$ and each pair of subsets S, S' of V, we define $\gamma(S, S', v)$ to be 1 if $v \in (S \setminus S') \cup (S' \setminus S)$, and to be 0 otherwise. Intuitively, $\gamma(S, S', v)$ is 1 if and only if the element v contributes to the Hamming distance between S and S'. Given this definition we can rewrite $\mathrm{HamDist}(S, S')$ as

$$\mathrm{HamDist}(S, S') = \sum_{v \in V} \gamma(S, S', v),$$

and the diversity of a list $S_1, \ldots, S_r$ of subsets of V as

$$\begin{aligned}\mathrm{Div}(S_1, \ldots, S_r) &= \sum_{1 \leq i < j \leq r} \sum_{v \in V} \gamma(S_i, S_j, v) \\ &= \sum_{v \in V} |\{\ell \ : \ v \in S_\ell\}| \cdot |\{\ell \ : \ v \notin S_\ell\}|.\end{aligned}$$

Now, if we define the *influence* of v on the list $S_1, \ldots, S_r$ as $I(S_1, \ldots, S_r, v) = |\{\ell \ : \ v \in S_\ell\}| \cdot |\{\ell \ : \ v \notin S_\ell\}|$, then we have that $\mathrm{Div}(S_1, \ldots, S_r) = \sum_{v \in V} I(S_1, \ldots, S_r, v)$.

3.2 From Vertex Cover to Diverse Vertex Cover

We now solve DIVERSE VERTEX COVER using dynamic programming over a tree decomposition of the input graph.

Let (G, k, r, d) be an instance of DIVERSE VERTEX COVER and let $\mathcal{D} = (T, q, \mathcal{X})$ be a rooted tree decomposition of G. For each node $t \in V(T)$, we define the set

$$\mathcal{I}_t = \{((S_1, s_1), \ldots, (S_r, s_r), \ell) \mid \ell \in [0, d], \forall i \in [1, r], S_i \subseteq X_t, s_i \in [0, k]\}.$$

This set $\mathcal{I}_t$, $t \in V(T)$, is such that the partial solutions we will construct for the node t will always be a subset of $\mathcal{I}_t$. Note that for each $t \in V(T)$, $|\mathcal{I}_t| \leq (2^{|X_t|} \cdot (k+1))^r \cdot (d+1)$. Now, our dynamic programming algorithm for DIVERSE VERTEX COVER consists in constructing for each $t \in V(T)$ a subset $\mathcal{R}_t \subseteq \mathcal{I}_t$ as follows. Let t be a node in $V(T)$ with children $t_1, \ldots, t_{\delta(t)}$. We recall that, by convention, this set of children is of size 0, 1, or 2. We let $\mathcal{R}_t$ be the set of all tuples $((S_1, s_1), \ldots, (S_r, s_r), \ell) \in \mathcal{I}_t$ satisfying the following additional properties:

1. For each $j \in [1, r]$, $E(G[X_t \setminus S_j]) = \emptyset$.
2. For each $i \in [1, \delta(t)]$ there exists a tuple $((S_1^i, s_1^i), \ldots, (S_r^i, s_r^i), \ell_i)$ in $\mathcal{R}_{t_i}$ such that
   (a) $S_j \cap X_{t_i} = S_j^i \cap X_t$ for each $i \in [1, \delta(t)]$ and each $j \in [1, r]$,
   (b) For each $j \in [1, r]$, $s_j = |\mathrm{forg}(t) \cap S_j| + \sum_{i=1}^{\delta(t)} s_j^i$,
   (c) and $\ell = \min(d, m)$ where $m = \sum_{v \in \mathrm{forg}(t)} I(S_1, \ldots, S_r, v) + \sum_{i=1}^{\delta(t)} \ell_i$.

Lemma 2. *(G, k, r, d) is a* YES*-instance of* DIVERSE VERTEX COVER *if and only if there is a tuple $((S_1, s_1), \ldots, (S_r, s_r), \ell)$ in $\mathcal{R}_q$ such that $\ell = d$.*

Proof. 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 $Y_t = X_t \setminus \mathrm{forg}(t)$, 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
- $\min(d, \mathrm{Div}(\widehat{S}_1 \setminus Y_t, \ldots, \widehat{S}_r \setminus Y_t)) = \ell$.

As the root q of the tree decomposition $\mathcal{D}$ is such that $X_q = \emptyset$, we obtain that the elements 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
- $\min(d, \mathrm{Div}(\widehat{S}_1, \ldots, \widehat{S}_r)) = \ell$.

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

Theorem 3. *Given a graph G, integers k, r, d, and a rooted tree decomposition $\mathcal{D} = (T, q, \mathcal{X})$ of G of width w, one can determine whether (G, k, r, d) is a* YES*-instance of* DIVERSE VERTEX COVER *in time*

$$\mathcal{O}(2^r \cdot (2^{w+1} \cdot (k+1))^{a \cdot r} \cdot d^a \cdot w \cdot r \cdot n),$$

where $a = \max_{t \in V(T)} \delta(t) \leq 2$ and $n = |V(T)|$.

Proof. 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 $2^{|\mathrm{new}(t)| \cdot r} \leq 2^r$ ways 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. □

Remark 4. *Given a graph G and a vertex cover Z of G of size k, one can find a rooted path decomposition $\mathcal{D} = (T, q, \mathcal{X})$ of G of width k, in linear time.*

This can be done by considering the bags $Z \cup \{v\}$ for each $v \in V(G)$ in any fixed order. Thus, from Theorem 3, we get the following corollary, which establishes an upper bound for the running time of our dynamic programming algorithm for DIVERSE VERTEX COVER solely in terms of the size k of the vertex cover, the number r of requested solutions, and the diversity d.

Corollary 5. DIVERSE VERTEX COVER *can be solved on an input (G, k, r, d) in time $\mathcal{O}((2^{k+2} \cdot (k+1))^r \cdot d \cdot k \cdot r \cdot |V(G)|)$.*

4 Computing Diverse Solutions using the Dynamic Programming Core model

In this section we show that the process illustrated in Section 3, of lifting a dynamic programming algorithm for a combinatorial problem to an algorithm for its diverse version, can be generalized to a much broader context. As a first step, we introduce the notion of *dynamic programming core*, a suitable formalization of the *intuitive* notion of tree-width based dynamic programming that satisfies three essential properties. First, this formalization is general enough to be applicable to a large class of combinatorial optimization problems. Second, this formalization is compatible with the notion of diversity, in the sense that the lifting of an algorithm for a problem to an algorithm for the diverse version of this problem can be done automatically, without requiring human ingenuity. Third, the resulting lifted algorithm is fast when compared with the original one. In particular, the running time of the resulting algorithm is polynomial on the diversity parameter. This is a highly desired property, since this allows our framework to be applied in the context where the sizes of the considered solution sets are not bounded.

Below, we let $\mathcal{G}$ be the set of simple, undirected graphs whose vertex set is a finite subset of $\mathbb{N}$. We say that a subset $\mathcal{P} \subseteq \mathcal{G}$ is a graph problem. Intuitively, a dynamic programming algorithm working on tree decompositions may be understood as a procedure that takes a graph $G \in \mathcal{G}$ and a rooted tree decomposition $\mathcal{D}$ of G as input, and constructs a certain amount of data for each node of $\mathcal{D}$. The data at node t is constructed by induction on the height of t, and in general, this data is used to encode the existence of a partial solution on the graph induced by bags in the sub-tree of $\mathcal{D}$

rooted at t. In the below definition, this is captured in the relation $\mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t)$. Such an algorithm accepts the input graph G if the data associated with the root node contains a string belonging to a set of *accepting strings*, captured below in the set $\mathrm{Accept}_{\mathfrak{C},G,\mathcal{D}}$. We formalize this intuitive notion in the following concept of *dynamic programming core*.

Definition 6 (Dynamic Programming Core)**.** *A* dynamic programming core *is an algorithm $\mathfrak{C}$ that takes a graph $G \in \mathcal{G}$ and a rooted tree decomposition $\mathcal{D}$ of G as input, and produces the following data.*

- *A finite set* $\mathrm{Accept}_{\mathfrak{C},G,\mathcal{D}} \subseteq 2^{\{0,1\}^*}$.
- *A finite set* $\mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t) \subseteq \left(2^{\{0,1\}^*}\right)^{\delta(t)+1}$ *for each* $t \in V(\mathcal{D})$.

We let $\tau(\mathfrak{C}, G, \mathcal{D})$ be the overall time necessary to construct the data associated with all nodes of $\mathcal{D}$. The size of $\mathfrak{C}$ on a pair $(G, \mathcal{D})$ is defined as

$$\mathrm{Size}(\mathfrak{C}, G, \mathcal{D}) = \max\{|\mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t)| \mid t \in V(\mathcal{D})\}.$$

Next, we define the notion of a *witness* for a dynamic programming core. Intuitively such witnesses are certificates of existence of a solution.

Definition 7. *Let $\mathfrak{C}$ be a dynamic programming core, G be a graph in $\mathcal{G}$, and $\mathcal{D} = (T, q, \mathcal{X})$ be a rooted tree decomposition of G. A* $(\mathfrak{C}, G, \mathcal{D})$-witness *is a function $\alpha : V(T) \to \{0,1\}^*$ such that the following conditions are satisfied.*

1. *For each $t \in V(T)$, with children $t_1, \ldots, t_{\delta(t)}$, $(\alpha(t), \alpha(t_1), \ldots, \alpha(t_{\delta(t)})) \in \mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t)$.*
2. *$\alpha(q) \in \mathrm{Accept}_{\mathfrak{C}}$.*

Using the notion of witness, we define formally what it means for a dynamic programming core to solve a combinatorial problem.

Definition 8. *We say that a dynamic programming core $\mathfrak{C}$ solves a problem $\mathcal{P}$ if for each graph $G \in \mathcal{G}$, and each rooted tree decomposition $\mathcal{D}$ of G, $G \in \mathcal{P}$ if and only if a $(\mathfrak{C}, G, \mathcal{D})$-witness exists.*

Theorem 9. *Let $\mathcal{P}$ be a graph problem and $\mathfrak{C}$ be a dynamic programming core that solves $\mathcal{P}$. Given a graph $G \in \mathcal{G}$ and a rooted tree decomposition $\mathcal{D}$ of G, one can determine whether $G \in \mathcal{P}$ in time $\mathcal{O}\left(\sum_{t \in V(T)} |\mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t)| + \tau(\mathfrak{C}, G, \mathcal{D})\right)$.*

4.1 Dynamic Programming Cores for Vertex Problems

Let $\mathfrak{C}$ be a dynamic programming core. A *$\mathfrak{C}$-vertex-membership function* is a function $\rho : \mathbb{N} \times \{0,1\}^* \to \{0,1\}$ such that for each graph G, each rooted tree decomposition $\mathcal{D} = (T, q, \mathcal{X})$ of G and each $(\mathfrak{C}, G, \mathcal{D})$-witness α, it holds that $\rho(v, \alpha(t)) = \rho(v, \alpha(t'))$ for each edge $(t, t') \in E(T)$ and each vertex $v \in X_t \cap X_{t'}$. Intuitively, if G is a graph and $\mathcal{D}$ is a rooted tree decomposition of G, then a $\mathfrak{C}$-vertex-membership together with a $(\mathfrak{C}, G, \mathcal{D})$-witness, provide an encoding of a subset of vertices of the graph. More precisely, we let

$$S_\rho(G, \mathcal{D}, \alpha) = \{v \mid \exists t \in V(T_{\mathcal{D}}), \rho(v, \alpha(t)) = 1\}$$

be this encoded vertex set. Given a $\mathfrak{C}$-vertex-membership function ρ, we let $\hat{\rho} : \{0,1\}^* \to 2^{\mathbb{N}}$ be the function that sets $\hat{\rho}(w) = \{v \in \mathbb{N} \mid \rho(v, w) = 1\}$ for each $w \in \{0,1\}^*$.

Let $\mathcal{P}$ be a vertex-problem, $\mathfrak{C}$ be a dynamic programming core, and ρ be a $\mathfrak{C}$-vertex-membership function. We say that $(\mathfrak{C}, \rho)$ *solves* $\mathcal{P}$ if for each graph $G \in \mathcal{G}$, each subset $S \subseteq V(G)$, and each rooted tree decomposition $\mathcal{D}$, $(G, S) \in \mathcal{P}$ if and only if there exists a $(\mathfrak{C}, G, \mathcal{D})$-witness α such that $S = S_\rho(G, \mathcal{D}, \alpha)$.

The following theorem is the main result of this section. It shows how to transform dynamic programming cores for problems $\mathcal{P}_1, \ldots, \mathcal{P}_r$ into a dynamic programming core for the problem $\mathrm{Div}^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$.

Theorem 10. *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex-problems, let $(\mathfrak{C}_i, \rho_i)$ be a dynamic programming core for $\mathcal{P}_i$, and let d be an integer. The problem $\mathrm{Div}^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$, on graph G with rooted tree decomposition $\mathcal{D} = (T, q, \mathcal{X})$, can be solved in time*

$$\mathcal{O}(d^a \cdot |V(T)| \cdot \prod_{i=1}^{r} \mathrm{Size}(\mathfrak{C}_i, G, \mathcal{D}) + \sum_{i=1}^{r} \tau(\mathfrak{C}_i, G, \mathcal{D})),$$

where $a = \max_{t \in V(T)} \delta(t) \leq 2$.

4.2 An Illustrative Application of Theorem 10

In this subsection we show how to apply Theorem 10 in the construction of an improved dynamic programming algorithm for DIVERSE VERTEX COVER. The first thing to do is to describe a dynamic programming core $\mathfrak{C}_{\mathrm{VC}}$ for k-VERTEX COVER. Given a graph G and a rooted tree decomposition $\mathcal{D} = (T, q, \mathcal{X})$, this dynamic programming core $\mathfrak{C}_{\mathrm{VC}}$ produces:

$$\mathrm{Accept}_{\mathfrak{C},G,\mathcal{D}} = \{(S, s) \mid S \subseteq X_q, s \leq k\}$$

$$\mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t) = \{((S, s), (S^1, s^1), \ldots, (S^{\delta(t)}, s^{\delta(t)})) \mid E(G[X_t \setminus S]) = \emptyset,\ \forall i \in [1, \delta(t)] : S^i \cap X_t = S \cap X_{t_i},\ s = |\mathrm{forg}(t) \cap S| + \sum_{t=1}^{\delta(t)} s^i\}$$

Provided the width of the decomposition is at most k, this can be done in time $\mathcal{O}((2^{k+1} \cdot (k+1))^{\delta(t)} \cdot k \cdot \delta(t))$ for each $t \in V(T)$, where the factor $k \cdot \delta(t)$ appears as we need the conditions $E(G[X_t \setminus S]) = \emptyset$ and $\forall i \in [1, \delta(t)], S^i \cap X_t = S \cap X_{t_i}$ to be verified. It is easy to verify that $\mathfrak{C}_{\mathrm{VC}}$ is a dynamic programming core for the VERTEX COVER problem. As described in Remark 4, we know that we can construct a rooted path decomposition of G of width k. We are now considering this rooted path decomposition. Thus, for each $t \in V(T)$, $|\mathrm{Process}_{\mathfrak{C},G,\mathcal{D}}(t)| \leq 2 \cdot 2^{k+1} \cdot (k+1)$. By Theorem 10, we obtain the following corollary, improving Corollary 5.

Corollary 11. DIVERSE VERTEX COVER *can be solved on an input (G, k, r, d) in time*

$$\mathcal{O}(d \cdot |V(G)| \cdot (2^{k+2} \cdot (k+1))^r + |V(G)| \cdot 2^{k+1} \cdot (k+1) \cdot k).$$

Note that we obtain a slightly better running time than for Corollary 5. This is due to the fact that verifying the properties $E(G[X_t \setminus S]) = \emptyset$ and $\forall i \in [1, \delta(t)], S^i \cap X_t = S \cap X_{t_i}$

is done when constructing $\mathfrak{C}_{\mathrm{VC}}$ and not when constructing $\mathfrak{C}$. Note also that, formally, we need to construct $\mathfrak{C}_{\mathrm{VC}}$ r times but as it is r times the same, we do the operation only once.

5 Diversity in Kernelization

Another key concept in the field of parameterized complexity is that of a *kernelization* algorithm [Fomin *et al.*, 2019]. We have obtained some parallel results about the kernelization complexity of diverse problems as well that we want to briefly sketch in this section. A polynomial kernel of a parameterized problem is a polynomial-time algorithm that given any instance either solves it, or constructs in polynomial time an equivalent[1] instance whose size is polynomial in the parameter. It is known that a parameterized problem is FPT if and only if it has a (not necessarily polynomial) kernel, and a natural step after proving a parameterized problem to be FPT is to decide whether or not it has a polynomial kernel.

We show that the diverse variants of several basic problems parameterized by the number of requested solutions plus solution size admit polynomial kernels as well. This is done via a variant of the recently introduced notion of *loss-less kernels* [Carbonnel and Hebrard, 2016] which are a special class of kernelizations that - very roughly speaking - for each but polynomially many bits of the input can either decide whether it has to be part of every solution or if it may be added to a solution without 'destroying' it.

For instance, consider the famous Buss kernel for VERTEX COVER [Buss and Goldsmith, 1993]: Given a graph G and an integer k, we want to decide if G has a vertex cover of size k. Each vertex of degree at least $k+1$ must be in each solution. Otherwise, we have to include its (at least) $k+1$ neighbors, exceeding the size constraint. On the other hand, each isolated (degree-0) vertex can be included in a vertex cover without destroying it, but it does not cover any edge. In the 'non-diverse' variant, we may remove these isolated vertices, and in the diverse variant, we have to keep some of them - they may be used to increase diversity - however, polynomially (in k and r) many such vertices suffice. Via such arguments, we obtain the following result about polynomial kernels of diverse problems.

Theorem 12. *The following diverse subset minimization problems parameterized by $k+r$ admit polynomial kernels:*

- DIVERSE VERTEX COVER*, on $\mathcal{O}(k(k+r))$ vertices;*
- DIVERSE d-HITTING SET *for fixed d, on $\mathcal{O}(k^d + kr)$ vertices;*
- DIVERSE POINT LINE COVER*, on $\mathcal{O}(k(k+r))$ points;*
- DIVERSE FEEDBACK ARC SET IN TOURNAMENTS*, on $\mathcal{O}(k(k+r))$ vertices.*

6 Conclusion

In this work, we considered a formal notion of diversity of a set of solutions to combinatorial problems in the setting of parameterized algorithms. We showed how to emulate treewidth based dynamic programming algorithms in order to solve diverse problems in FPT time, with the number r of requested solutions being an additional parameter.

This line of research is now wide open, with many natural questions to address. As all our results are of a positive nature, we ask: when can diversity be a source of hardness? Concretely, a natural target in parameterized complexity would be to identify a parameterized problem Π that is FPT, however DIVERSE Π being W[1]-hard when r is an additional parameter. For positive results, an interesting research direction would be to generalize our framework for diverse problems to other well studied width measures for graphs, as well as to other structures, such as matroids.

In this work, we considered the *sum* of all pairwise Hamming distances of a set as a measure of diversity. As pointed out, this measure has some weaknesses, and another widely used measure is the *minimum* Hamming distance. In this setting, we only obtain FPT-results when the diversity is bounded by a function of the treewidth and the number of solutions, but not in general. So, a natural follow-up question is whether or not we can obtain FPT-results under the minimum Hamming distance, even if the diversity is unbounded.

[1]Meaning that the constructed instance is a YES-instance if and only if the original instance was.

Acknowledgements

M. Fellows, F. Rosamond, and G. Philip acknowledge support from the Research Council of Norway grant "Parameterized Complexity for Practical Computing (PCPC)" (NFR, no. 274526). M. Fellows, F. Rosamond, L. Jaffke, M. de Oliveira Oliveira, and G. Philip acknowledge support from the Trond Mohn Foundation (TMS). J. Baste acknowledges support from the German Research Foundation (DFG, no. 388217545). T. Masařík acknowledges support from the European Research Council (ERC, no. 714704), and from Charles University's grants GAUK 1514217 and SVV-2017-260452. He recently started a postdoc at Simon Fraser University, Canada. M. de Oliveira Oliveira acknowledges support from the Research Council of Norway (NFR, no. 288761). G. Philip acknowledges support from the NFR grants "MULTIVAL" and "CLASSIS", and from the European Research Council (ERC, no. 819416).

References

[Abbassi *et al.*, 2013] Zeinab Abbassi, Vahab S. Mirrokni, and Mayur Thakur. Diversity maximization under matroid constraints. 19th KDD, pages 32–40, 2013.

[Adomavicius and Kwon, 2014] Gediminas Adomavicius and YoungOk Kwon. Optimization-based approaches for maximizing aggregate recommendation diversity. *INFORMS Journal on Computing*, 26(2):351–369, 2014.

[Barrett *et al.*, 2007] Chris Barrett, Harry B Hunt, Madhav V Marathe, SS Ravi, Daniel J Rosenkrantz, Richard E Stearns, and Mayur Thakur. Computational aspects of analyzing social network dynamics. In *20th IJCAI*, pages 2268–2273, 2007.

[Baste *et al.*, 2019a] Julien Baste, Michael R. Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A. Rosamond. Diversity

of solutions: An exploration through the lens of fixed-parameter tractability theory, 2019. arXiv:1903.07410.

[Baste *et al.*, 2019b] Julien Baste, Lars Jaffke, Tomáš Masařík, Geevarghese Philip, and Günter Rote. FPT algorithms for diverse collections of hitting sets. *Algorithms*, 12:254, 2019.

[Bliem *et al.*, 2017] Bernhard Bliem, Marius Moldovan, Michael Morak, and Stefan Woltran. The impact of treewidth on ASP grounding and solving. In *26th IJCAI*, pages 852–858, 2017.

[Bodirsky and Wölfl, 2011] Manuel Bodirsky and Stefan Wölfl. RCC8 is polynomial on networks of bounded treewidth. In *22nd IJCAI, Vol. 2*, pages 756–761, 2011.

[Buss and Goldsmith, 1993] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993.

[Carbonnel and Hebrard, 2016] Clément Carbonnel and Emmanuel Hebrard. Propagation via kernelization: The vertex cover constraint. In *22nd CP*, pages 147–156, 2016.

[Chavira and Darwiche, 2007] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks using variable elimination. In *20th IJCAI*, pages 2443–2449, 2007.

[Courcelle, 1990] Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inform. Comput.*, 85(1):12–75, 1990.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Danna and Woodruff, 2009] Emilie Danna and David L. Woodruff. How to select a small set of diverse solutions to mixed integer programming problems. *Operations Research Letters*, 37(4):255–260, 2009.

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[Eiter *et al.*, 2013] Thomas Eiter, Esra Erdem, Halit Erdogan, and Michael Fink. Finding similar/diverse solutions in answer set programming. *Theory and Practice of Logic Programming*, 13(3):303–359, 2013.

[Fomin *et al.*, 2019] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization*. Cambridge University Press, 2019.

[Gabor *et al.*, 2018] Thomas Gabor, Lenz Belzner, Thomy Phan, and Kyrill Schmid. Preparing for the unexpected: Diversity improves planning resilience in evolutionary algorithms. In *15th ICAC*, pages 131–140, 2018.

[Galle, 1989] Per Galle. Branch & sample: A simple strategy for constraint satisfaction. *BIT Numer. Math.*, 29(3):395–408, 1989.

[Glover *et al.*, 2000] Fred Glover, Arne Løkketangen, and David L. Woodruff. Scatter search to generate diverse mip solutions. In *Computing Tools for Modeling, Optimization and Simulation*, pages 299–317. Springer, 2000.

[Gollapudi and Sharma, 2009] Sreenivas Gollapudi and Aneesh Sharma. An axiomatic approach for result diversification. In *18th WWW*, pages 381–390, 2009.

[Greistorfer *et al.*, 2008] Peter Greistorfer, Arne Løkketangen, Stefan Voß, and David L. Woodruff. Experiments concerning sequential versus simultaneous maximization of objective function and distance. *Journal of Heuristics*, 14(6):613–625, 2008.

[Hadžić *et al.*, 2009] Tarik Hadžić, Alan Holland, and Barry O'Sullivan. Reasoning about optimal collections of solutions. In *15th CP*, pages 409–423. Springer, 2009.

[Hebrard *et al.*, 2005] Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In *20th AAAI*, pages 372–377, 2005.

[Hebrard *et al.*, 2007] Emmanuel Hebrard, Barry O'Sullivan, and Toby Walsh. Distance constraints in constraint satisfaction. In *IJCAI*, pages 106–111, 2007.

[Jégou *et al.*, 2007] Philippe Jégou, Samba Ndojh Ndiaye, and Cyril Terrioux. Dynamic heuristics for backtrack search on tree-decomposition of CSPs. In *20th IJCAI*, pages 112–117, 2007.

[Kuo *et al.*, 1993] Ching-Chung Kuo, Fred Glover, and Krishna S. Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming*. *Decis. Sci.*, 24(6):1171–1185, 1993.

[Løkketangen and Woodruff, 2005] Arne Løkketangen and David L. Woodruff. A distance function to support optimized selection decisions. *Decision Support Systems*, 39(3):345–354, 2005.

[Nadel, 2011] Alexander Nadel. Generating diverse solutions in sat. In *14th SAT*, pages 287–301. Springer, 2011.

[Petit and Trapp, 2015] Thierry Petit and Andrew C. Trapp. Finding diverse solutions of high quality to constraint optimization problems. In *24th IJCAI*, pages 260–267, 2015.

[Petit and Trapp, 2019] Thierry Petit and Andrew C. Trapp. Enriching solutions to combinatorial problems via solution engineering. *INFORMS Journal on Computing*, 31(3):429–444, 2019.

[Schittekat and Sörensen, 2009] Patrick Schittekat and Kenneth Sörensen. OR practice — supporting 3PL decisions in the automotive industry by generating diverse solutions to a large-scale location-routing problem. *Operations Research*, 57(5):1058–1067, 2009.

[Trapp and Konrad, 2015] Andrew C. Trapp and Renata A. Konrad. Finding diverse optima and near-optima to binary integer programs. *IIE Trans.*, 47(11):1300–1312, 2015.

[Wineberg and Oppacher, 2003] Mark Wineberg and Franz Oppacher. The underlying similarity of diversity measures used in evolutionary computation. In *GECCO '03, Part II*, pages 1493–1504, 2003.

[Yang and Wang, 2018] Yongjie Yang and Jianxin Wang. Multiwinner voting with restricted admissible sets: complexity and strategyproofness. In *27th IJCAI*, pages 576–582, 2018.