Finding Diverse Solutions Parameterized by Cliquewidth

Karolina Drabik, Tomáš Masařík

University of Warsaw, Poland
karolina.drabik@mimuw.edu.pl, masarik@mimuw.edu.pl

Abstract

Finding a few solutions for a given problem that are diverse, as opposed to finding a single best solution to solve the problem, has recently become a notable topic in theoretical computer science. Recently, Baste, Fellows, Jaffke, Masařík, Oliveira, Philip, and Rosamond showed that under a standard structural parameterization by treewidth, one can find a set of diverse solutions for many problems with only a very small additional cost [Artificial Intelligence 2022]. In this paper, we investigate a much stronger graph parameter, the cliquewidth, which can additionally describe some dense graph classes. Broadly speaking, it describes graphs that can be recursively constructed by a few operations defined on graphs whose vertices are divided into a bounded number of groups, while each such group behaves uniformly with respect to any operation.

We show that for any vertex problem, if we are given a dynamic program solving that problem on cliquewidth decomposition, we can modify it to produce a few solutions that are as diverse as possible with as little overhead as in the above-mentioned treewidth paper. As a consequence, we prove that a diverse version of any MSO_1 expressible problem can be solved in linear FPT time parameterized by the cliquewidth, the number of sought solutions, and the number of quantifiers in the formula, which was a natural missing piece in the complexity landscape of structural graph parameters and logic for the diverse problems. We prove our results, allowing for a more general natural collection of diversity functions compared to only two mostly studied diversity functions previously. That might be of independent interest as a larger pool of different diversity functions can highlight various aspects of different solutions to a problem.

Extended version — https://arxiv.org/abs/2405.20931

1 Introduction

In a standard graph problem, one typically asks for a solution of the smallest size. The first difficulty that typically needs to be overcome is that a large proportion of important problems is hard to solve (for some notion of hardness). There is a vast branch of research dedicated to overcoming such a difficulty, like parameterized complexity, approximation algorithms, and heuristics, among others. Second, maybe a more pressing issue is that even once we are given a single optimal solution for the problem, we might not be able to learn as much about our original tasks as we had hoped. Why might it be so? Mathematically formulated problems are typically just an abstraction of real-world problems, where there might be other important factors that were perhaps just too complex to formalize or even not known in the first place. Such constraints might be visible only once an actual solution is presented. While the first difficulty stands, the second difficulty might be overcome or at least diminished by the following idea. Instead of searching for just one solution, we will search for various solutions that are as different as possible (for some notion of what different means, to be specified later). Moreover, the idea of looking for a diverse set of solutions may even further allow us to relax the original mathematical formulation of a problem, which in turn may allow for more efficient algorithms combating the first-mentioned difficulty. There are many examples of advantages producing numerous different solutions in the literature; see the famous architect problem (Galle 1989) as one of the earliest examples. Since then, there have been countless applications of this principle in various branches of computer science: constraint programming (Hébrard, O'Sullivan, and Walsh 2007; Hébrard et al. 2005; Petit and Trapp 2015; Ingmar et al. 2020), mixed integer optimization (Ahanor, Medal, and Trapp 2024; Danna and Woodruff 2009; Glover, Løkketangen, and Woodruff 2000), evolutionary computation (Do et al. 2022; Gabor et al. 2018; Wineberg and Oppacher 2003), social choice (Boehmer and Niedermeier 2021; Arrighi et al. 2021), and discrete geometry (Klute and van Kreveld 2022) among many others.

It was not until 2019 that this concept was brought to the attention from the perspective of theoretical graph algorithms and parameterized complexity toolbox (Baste et al. 2022). They showed that a dynamic program (DP) on a tree decomposition can be converted to a DP finding diverse solutions. Since then, this concept under the parameterized complexity examination started to play a more prominent role, and many follow-up papers have been published (Baste et al. 2019; Hanaka et al. 2021; de Berg, López Martínez, and Spieksma 2023; Fomin et al. 2024b; Hanaka et al. 2023; Arrighi et al. 2023; Do et al. 2023; Fomin et al. 2024a).

Now, let us define diverse problems formally. We start by explaining the two basic diversity measures that were within the central focus of the theoretical research on diverse prob-

lems. We define the *Hamming distance* between two sets S and S' as the size of their symmetric difference. That is

$$HamDist(S, S') \coloneqq |(S \setminus S') \cup (S' \setminus S)|.$$

We use it to define diversity measures of an (ordered) collection of sets. We define *Sum diversity* as

$$DivSum(S_1, \ldots, S_r) \coloneqq \sum_{1 \le i < j \le r} HamDist(S_i, S_j).$$

Then, we define *Min diversity* as

$$DivMin(S_1, \ldots, S_r) \coloneqq \min_{1 \le i < j \le r} HamDist(S_i, S_j).$$

Having defined the basic measures, we can state the problem formally. In the following, we use Div to denote an arbitrary diversity measure that takes r subsets of vertices and outputs an integer, for example, $DivSum$ or $DivMin$ functions defined above. Note that in this work, we are concerned only with vertex problems, but this template can be easily modified to take into account any list of subsets. Formally, a *vertex problem* is a set of pairs (G, S), where G is a graph and S is a subset of its vertices that satisfies some property.

Definition 1.1 (Diverse Problem)**.** *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems, Div be a diversity measure, $d \in \mathbb{N}$. Then*

$$Div^d(\mathcal{P}_1, \ldots, \mathcal{P}_r) \coloneqq \{(G, S_1, \ldots, S_r) | \\ (G, S_i) \in \mathcal{P}_i, Div(S_1, \ldots, S_r) \ge d\}.$$

In our paper, we focus on FPT algorithms for diverse vertex problems, parameterized by the *cliquewidth* of the graph, that means algorithms running in time $f(\omega) \cdot n^{O(1)}$, where ω is the parameter (the cliquewidth) and n is the graph size.

A graph G has cliquewidth ω if there is a rooted tree that describes the construction of G from single vertices via the listed simple operations using at most ω colors.

Definition 1.2 (Cliquewidth)**.** *Let $\omega \in \mathbb{N}$. A* cliquewidth decomposition *of* width ω *is a rooted tree $\mathcal{D}$ where each node $t \in V(\mathcal{D})$ is of one of the following types:*

- *$t = intro(v, a)$, where $a \in [\omega]$ and t is a leaf of $\mathcal{D}$. This node corresponds to construing a one-vertex colored graph with a vertex v of color a.*
- *$t = t_1 \oplus t_2$, where t_1, t_2 are children of t. This node corresponds to taking the disjoint union of colored graphs constructed by the subtrees of t_1, t_2.*
- *$t = recolor(t', a \to b)$, where t' is the only child of t, $a, b \in [\omega]$. This node corresponds to recoloring all the vertices of color a in a graph constructed by the subtree of t' to color b.*
- *$t = addEdges(t', a, b)$, where t' is the only child of t, $a, b \in [\omega]$. This node corresponds to adding an edge between each pair of vertices u, v of colors a, b respectively in a graph constructed by a subtree of t' if such an edge does not exist.*

For a node $t \in V(\mathcal{D})$ we denote by $\mathcal{D}_t$ the subtree of $\mathcal{D}$ rooted at t, by G_t the colored graph constructed by $\mathcal{D}_t$, and by $\delta(t)$ the number of children of t. Let G be a simple graph. $\mathcal{D}$ is a cliquewidth decomposition of G if G is isomorphic to G_t, where t is the root of $\mathcal{D}$.

Observe that in this definition, for each $v \in V(G)$, there is exactly one node $t = intro(v, a)$, where $a \in [\omega]$. We also write $t = Leaf_{\mathcal{D}}(v)$. The *cliquewidth* of G is minimal $\omega \in \mathbb{N}$ such that G admits a decomposition $\mathcal{D}$ of width ω.

The cliquewidth has played an influential role since its definition (Courcelle, Engelfriet, and Rozenberg 1993). This has only increased after a seminal result of Courcelle, Makowsky, and Rotics proving that any MSO_1 property can be checked in linear time on graphs of bounded cliquewidth (Courcelle, Makowsky, and Rotics 2000). This result essentially means that many difficult vertex problems on graphs are solvable in linear time, provided that the cliquewidth of that graph is small. The most prominent tool to solve problems on graphs of bounded cliquewidth is dynamic programming without any doubt. Besides the model-checking theorem, there have been countless examples of using dynamic programming to solve some particular problems, either because those problems are not expressible in MSO_1 or because they provide a much faster algorithm. For example, feedback vertex set (Bui-Xuan et al. 2013), triangle detection & girth (Coudert, Ducoffe, and Popa 2019), Alliances (Kiyomi and Otachi 2017), Hamiltonian cycle (Bergougnoux, Kanté, and Kwon 2019), Steiner tree (Bojikian and Kratsch 2024b), connected odd cycle transversal (Bojikian and Kratsch 2024a), and other connectivity problems (Bergougnoux and Kanté 2019; Hegerfeld and Kratsch 2023).

We would like to point out that for our purposes, there is very little difference between the case when the cliquewidth decomposition is given on the input and the case when it first needs to be computed. Indeed, despite being NP-hard to be computed optimally (Fellows et al. 2009), there is a $(2^{2\omega+1} - 1)$-approximation algorithm running in quadratic time, where ω is the cliquewidth of the graph on the input (Fomin and Korhonen 2022). Therefore, from now on, we will simply assume that the decomposition is given on the input, so we do not need to argue this in many places. Moreover, it allows us to state our results independently of potential improvements of the above-mentioned approximation algorithm.

1.1 Our Results

Our main contribution is the following. Given a cliquewidth decomposition of a graph G and (almost any) dynamic program solving a vertex problem on G, we are able to translate it to an algorithm solving a diverse version of the problem with only a very small expense on the running time. In fact, the running time of the diverse algorithm will be essentially the r-th power of the running time required to compute a single solution when we aim to find r diverse solutions.

More formally, we define what it means for dynamic programming to be monotone; see Definition 2.5. Despite its inherent technical nature, the idea behind this definition is very natural. Let us explain the intuition here. Take a dynamic programming table entry ε within some inner node of the DP. Such ε could originate from many different possible combinations of other entries evaluated earlier (in subtrees of the given node) in the DP. We call it the *derivation tree* of ε. To each such derivation tree, we assign a set of ver-

tices, called *partial solution*, based only on the entries in the leaves of the decomposition that combine to ε. Moreover, for the leaf node, we only allow for two entries: one where the vertex corresponding to a leaf node is in the partial solution and the other when it is not. We call the described property monotonicity, as the set of vertices assigned to the derivation tree is the union of the sets assigned to its subtrees.

With the intuition on what monotone DPs look like, we formulate our main theorem. In the following, we use Div to denote an arbitrary diversity measure from a collection of Venn measures that will be defined later; see Definition 1.9 for a formal definition. At this point, we keep in mind that $DivSum$ is one such measure, but $DivMin$ is not.

Theorem 1.3 (Diverse problems on cw)**.** *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems and given monotone dynamic programs solving them with the slowest running on a single node of the decomposition in time $t_{\mathcal{D}}$ for graph G with cliquewidth decomposition $\mathcal{D}$. Let Div be a Venn diversity function. The problem $Div^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on a graph G with given cliquewidth decomposition $\mathcal{D}$ can be solved in time $\mathcal{O}(|V(\mathcal{D})|t_{\mathcal{D}}^r)$.*

It is important to note that the running time of Theorem 1.3 does not depend on the actual value of the Div function. Our proof also provides the same statement for the $DivMin$ function, but with a worse running time dependent on the target value d of the $DivMin$ function.

Theorem 1.4 (Min Diverse problems on cw)**.** *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems and given monotone dynamic programs solving them with the slowest running on a single node of the decomposition in time $t_{\mathcal{D}}$ for graph G with cliquewidth decomposition $\mathcal{D}$. The problem $DivMin^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on a graph G with given cliquewidth decomposition $\mathcal{D}$ can be solved in time $\mathcal{O}(|V(\mathcal{D})|t_{\mathcal{D}}^r d^{r^2})$.*

We would like to compare the running time of Theorem 1.3 with an analogous theorem for the treewidth (Baste et al. 2022, Theorem 10). We present it with a minor improvement, where we omit the factor d^a (typically $a = 2$) stated in the referenced running time as it is not necessary once one aims to find a single diverse solution of the best possible value of $DivSum$. We discuss the change at the end of Section 2.

Theorem 1.5 (Reformulated (Baste et al. 2022, Theorem 10) with a slight improvement)**.** *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems and given dynamic program core (as defined in (Baste et al. 2022, Definition 6)) solving them with the slowest running on a single node of the decomposition in time $t_{\mathcal{T}}$ for graph G with tree decomposition $\mathcal{T}$. The problem $DivSum^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on a graph G with given treewidth decomposition $\mathcal{T}$ can be solved in time $\mathcal{O}(|V(\mathcal{T})|t_{\mathcal{T}}^r)$.*

We claim that a vast majority of dynamic programs solving vertex problems using the cliquewidth decomposition are either monotone or could be easily transformed to be monotone. To the best of our knowledge, we have not found any examples of non-monotone DPs in the literature. To support the intuition, we designed a non-monotone DP for the vertex cover problem. However, this algorithm can be easily turned into a monotone one; see (Drabik and Masařík 2024). In order to support our claim, we showed that using our framework, one can solve the diverse version of any vertex problem that can be expressed in MSO_1 logic, which is the fragment of second-order logic where the second-order quantification is limited to quantifying over sets of vertices. The MSO_1 captures many well-studied problems, including: Vertex Cover, Independent Set, Dominating Set, and Feedback Vertex Set problems. We show that any problem $\mathcal{P}$ for which there exists an MSO_1 formula $\Phi(S)$ such that $G \models \Phi(S)$ if and only if $(G, S) \in \mathcal{P}$ can be solved by a monotone dynamic program to which we apply Theorem 1.3. As the original result (Courcelle, Makowsky, and Rotics 2000) gives the DP only implicitly, we decided to design a new DP algorithm and show that it is indeed monotone. We based our approach on lecture notes (Fiala 2024, Section 5.4), which show how to use evaluation trees to obtain a similar result for treewidth decomposition.

Corollary 1.6. *Diverse MSO_1 problems can be solved in linear FPT time parameterized by the cliquewidth, the number of solutions, and the number of quantifiers in the formula.*

Corollary 1.6 is interesting on its own as it provides a new result for graph vertex problems expressible in certain logic. We can compare it with the original treewidth result (Baste et al. 2022), which encompasses problems that can be defined in MSO_2 logic and then provides the diverse variant in the running time comparable to Corollary 1.6. That is a stronger logic, but a much weaker graph class. Note that model checking of MSO_2 problems on cliques is not even in XP unless E = NE (Courcelle, Makowsky, and Rotics 2000). Later, Hanaka, Kobayashi, Kurita, and Otachi (Hanaka et al. 2021, Theorem 13) proved that problems that can be defined in FO logic on a nowhere dense graph class can have their Diverse variant solved in FPT time. However, this time the result relies heavily on the model-checking machinery (Grohe, Kreutzer, and Siebertz 2017) and the claimed running time is much worse than linear. Again, this result is incomparable with Corollary 1.6 as it acts on a larger class of graphs, but only on problems defined in a weaker logic. (Bergougnoux, Dreier, and Jaffke 2023) worked on an A&C DN logic (distance neighborhood logic with acyclicity and connectivity constraints). Informally speaking, DN logic allows for size measurement of neighborhood terms as well as inclusion-wise comparison of neighborhoods within the *existential* fragment of MSO_1 logic, while A&C further add two more constraints which allow us to express acyclicity and connectivity of the neighborhood term. They showed that if a vertex problem can be expressed in A&C DN logic, then one can express $DivMin$ or $DivSum$ variant of the problem in that logic as well. Then they present a cubic FPT algorithm solving A&C DN encoded problems on graphs of bounded cliquewidth parameterized by the size of the formula and the cliquewidth. Additionally, they present an XP algorithm for the same problem on graphs of bounded mimwidth which is a much more general structural graph parameter. Indeed, as this covers

only an existential fragment of MSO_1, it is incomparable with Corollary 1.6. Let us stress that all the related work is stated only for $DivSum$ or possibly for $DivMin$ measures.

New Venn Diversity Measures. In our work, we propose a whole collection of diversity measures. We believe they will be of independent interest. Despite being very natural, to the best of our knowledge, they have not been studied in this context before. In the previous theoretical works, research was only focused on the two measures already defined: $DivMin$ and $DivSum$. However, despite them being the first natural examples, there is nothing so special about them. One can find examples where those measures do not behave exactly as expected. In the case of the $DivSum$ measure, a profound example is taking many copies of two disjoint sets, which results in a relatively high diversity value. Still, intuitively, such a tuple of sets is not diverse. In particular, it does not display a potentially rich structure of various different solutions as we might have hoped. This was already observed and discussed in (Baste et al. 2019, Section 5, Figure 3). Such problems are not limited to $DivSum$; see the full version of this article (Drabik and Masařík 2024) for a problematic example for the $DivMin$ measure.

Therefore, we are convinced that our approach, which gives access to the whole collection of diversity measures, much better supports the nature of the diverse problems. Indeed, it seems in line with the diversity paradigm, where one seeks a collection of solutions rather than a single one. In the same spirit, we advocate trying different diversity measures and then comparing their results. However, we can be more subtle as we have access to a large collection of measures. We suggest that one might pick the measures dynamically. This means we can choose the measures not only based on the instance but also based on the diverse collection(s) we have obtained so far. If we obtain one of the degenerate examples, or simply, whenever we do not like the solution given by one diversity function, we can adjust the diversity function to penalize such an outcome and rerun the algorithm.

We will shortly give an example of a measure that penalizes an unwanted outcome returned using the $DivSum$ measure, but first, we define Venn diversity measures formally. We start by defining the membership vector, which is basically a characteristic vector of a vertex v being an element of set S_i in the list of sets $S_1, \ldots, S_r$.

Definition 1.7 (Membership vector)**.** *For $v \in V$ and a list $S_1, \ldots, S_r$ of subsets of V a* membership vector *of v in $S_1, \ldots, S_r$ is a vector $m(S_1, \ldots, S_r, v) \in \{0,1\}^r$ such that for each $i \in [r]$,*

$$m(S_1, \ldots, S_r, v)[i] = \begin{cases} 1 & \text{if } v \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

Then we define the influence of a vertex provided with a function f where the value of f is determined by the membership vector value.

Definition 1.8 (f-influence)**.** *Let $f : \{0,1\}^r \to \mathbb{N}$. For $v \in V$ and a list $S_1, \ldots, S_r$ of subsets of V the* f-influence *of v on the list $S_1, \ldots, S_r$ is the number*

$$I_f(S_1, \ldots, S_r, v) \coloneqq f(m(S_1, \ldots, S_r, v)).$$

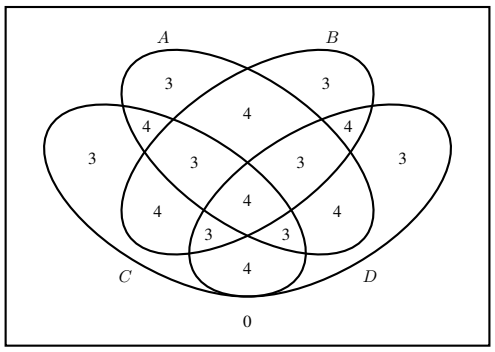

Figure 1: A Venn diagram for four solutions: A, B, C, D. The influence of a vertex corresponding to the $DivSum$ measure is marked. With Venn diversity measures, one could assign an arbitrary influence value to each part of this diagram.

Observe that in this definition, the influence depends only on the presence of v in each of $S_1, \ldots, S_r$. Finally, we define the Venn function; see Figure 1.

Definition 1.9 (Venn f-diversity)**.** *Let $f : \{0,1\}^r \to \mathbb{N}$. We define the* Venn f-diversity *of a list $S_1, \ldots, S_r$ of subsets of V to be*

$$Div_f(S_1, \ldots, S_r) \coloneqq \sum_{v \in V} I_f(S_1, \ldots, S_r, v).$$

Whenever f is fixed, we will write $I(S_1, \ldots, S_r, v)$ instead of $I_f(S_1, \ldots, S_r, v)$ and $Div(S_1, \ldots, S_r)$ instead of $Div_f(S_1, \ldots, S_r)$, and we say Venn diversity instead of Venn f-diversity. We now give an example of a Venn f-diversity measure other than $DivSum$.

Example 1.10.

$$Div^*(S_1, \ldots, S_r) \coloneqq \sum_{v \in V} (r^2 - |\{i \in [r] : v \in S_i\}|^2).$$

Here, the influence of v on $S_1, \ldots, S_r$ depends on the intersection of how many sets from $S_1, \ldots, S_r$ v belongs to.

Returning to our example with only two distinct solutions copied multiple times, achieving the best $DivSum$ value, we suggest that in that case, one might try to use the Div^* measure defined in Example 1.10. Contrary to the sum of pairwise hamming distances, this measure penalizes taking many copies of two disjoint sets; see Figure 2.

Organization of the paper. We give the proof of our main theorem (Theorem 1.3) in Section 2. In Section 3, we provide a monotone DP for MSO_1 problems that implies Corollary 1.6. We conclude with a discussion about open problems and further extensions in Section 4.

2 Monotone DP for Vertex Problems

A very natural property of dynamic programming algorithms for vertex problems on cliquewidth decompositions is that each partial solution for a node t is a disjoint union

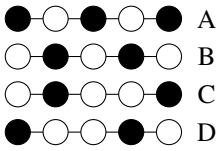

Figure 2: Consider the dominating set problem on a path of five vertices. If the number of required solutions is four, then a solution maximizing $DivSum$ will consist of twice A and twice B, having $DivSum$ equal to 20. However, for the Div^* function, the value for this set of solutions is equal to 60, which is less than for an arguably more diverse set of solutions consisting of all A, B, C, and D, with the Div^* being equal to 63.

of partial solutions for its children. We call algorithms with this property *monotone*.

In this section, we give the formalism of dynamic programming and the monotone property. We prove our main result, that having a monotone dynamic program for each of the vertex problems $\mathcal{P}_1, \ldots, \mathcal{P}_r$ and a cliquewidth decomposition $\mathcal{D}$ of G, we can solve $Div^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on G. Recall that $\delta(t)$ denotes the number of children of a node $t \in V(D)$.

Definition 2.1 (Dynamic programming core)**.** *A* dynamic programming core *is an algorithm $\mathfrak{C}$ that takes cliquewidth decomposition $\mathcal{D}$ of a graph G and produces a set $Accept_{\mathfrak{C},\mathcal{D}} \subseteq \{0,1\}^*$ and for each $t \in V(\mathcal{D})$ a set $Process_{\mathfrak{C},\mathcal{D}}(t) \subseteq (\{0,1\}^*)^{\delta(t)+1}$.*

Let $\tau(\mathfrak{C}, \mathcal{D})$ be the time necessary to construct these data and $Size_{\mathfrak{C},\mathcal{D}} := max_{t \in V(\mathcal{D})} |Process_{\mathfrak{C},\mathcal{D}}(t)|$.

Intuitively, $Process_{\mathfrak{C},\mathcal{D}}$ corresponds to the transitions of the dynamic programming. If a partial solution represented by w at node t can be obtained from partial solutions represented by $w_1 \ldots w_{\delta(t)}$ at its children $t_1 \ldots t_{\delta(t)}$ then $ww_1 \ldots w_{\delta(t)} \in Process_{\mathfrak{C},\mathcal{D}}(t)$.

Definition 2.2 (Witness)**.** *Let $\mathcal{D}$ be a cliquewidth decomposition of a graph G with a root q and $\mathfrak{C}$ be a dynamic programming core. For $w \in \{0,1\}^*$ a $(\mathfrak{C}, \mathcal{D}, w)$-witness is a function $\alpha : V(\mathcal{D}) \to \{0,1\}^*$ such that for each $t \in V(\mathcal{D})$ $\alpha(t)\alpha(t_1) \ldots \alpha(t_{\delta(t)}) \in Process_{\mathfrak{C},\mathcal{D}}(t)$ and $\alpha(q) = w$.*

If $w \in Accept_{\mathfrak{C},\mathcal{D}}$, we say α is a $(\mathfrak{C}, \mathcal{D})$-witness.

Intuitively, if w is a dynamic programming entry in the root t of $\mathcal{D}$, it can originate from many combinations of entries in the children of t. A $(\mathfrak{C}, \mathcal{D}, w)$-witness describes one of many derivation trees for w.

Observe that if $\mathcal{P}$ is a vertex problem, given a graph G, we cannot hope to list all S such that $(G, S) \in \mathcal{P}$ in FPT time, as the number of such sets may be exponential in $|G|$. Therefore, we focus on deciding whether for a given graph G there exists any such S. We call this decision problem a *graph problem* and write $G \in \mathcal{P}$ if the answer is YES.

Definition 2.3. *We say that a dynamic programming core $\mathfrak{C}$* solves *a graph problem $\mathcal{P}$ if for each graph G and for each cliquewidth decomposition $\mathcal{D}$ of G, $G \in \mathcal{P}$ if and only if a $(\mathfrak{C}, \mathcal{D})$-witness exists.*

That means G is a YES-instance of $\mathcal{P}$ if and only if there exists a derivation tree for some accepting entry w.

Theorem 2.4. *Let $\mathcal{P}$ be a graph problem and $\mathfrak{C}$ be a dynamic programming core that solves $\mathcal{P}$. Given a decomposition $\mathcal{D}$ of a graph G we can determine whether $G \in \mathcal{P}$ in time $\mathcal{O}\left(\tau(\mathfrak{C}, \mathcal{D}) + |V(\mathcal{D})| \cdot Size_{\mathfrak{C},\mathcal{D}}\right)$.*

Proof. Given $\mathfrak{C}, \mathcal{D}$, we can construct $Accept_{\mathfrak{C},\mathcal{D}}$ and $Process_{\mathfrak{C},\mathcal{D}}(t)$ for each $t \in V(\mathcal{D})$ in time $\tau(\mathfrak{C}, \mathcal{D})$. Let $\Pi(t)$ be the set of $w \in \{0,1\}^*$ such that a $(\mathfrak{C}, \mathcal{D}_t, w)$-witness exists. We can construct $\Pi(t)$ for each $t \in V(\mathcal{D})$ by induction. If $\delta(t) = 0$, $\Pi(t) = Process_{\mathfrak{C},\mathcal{D}}(t)$. Otherwise, $\Pi(t) = \{w | \exists_{(w, w_1, \ldots w_{\delta(t)}) \in Process_{\mathfrak{C},\mathcal{D}}(t)} \forall_{i \in [\delta(t)]} w_i \in \Pi(t_i)\}$. $G \in \mathcal{P}$ if and only if $\Pi(q) \cap Accept_{\mathfrak{C},\mathcal{D}} \neq \emptyset$, where q is the root of $\mathcal{D}$. □

Note that from the above algorithm we can also easily recover some $(\mathfrak{C}, \mathcal{D}, w)$-witness for each $w \in \Pi(q)$

Definition 2.5 (Monotonicity)**.** *Let $\mathcal{P}$ be a vertex problem. We call a dynamic programming core $\mathfrak{C}$* monotone *if there exists a function $\rho : \{0,1\}^* \to \{0,1\}$ such that for each graph G, for each cliquewidth decomposition $\mathcal{D}$ of G, for each $S \subseteq V(G)$, $(G, S) \in \mathcal{P}$ if and only if there exists a $(\mathfrak{C}, \mathcal{D})$-witness α such that $S = \{v \in V(G) : \rho(\alpha(Leaf_{\mathcal{D}}(v))) = 1\}$. We call such ρ a* $\mathfrak{C}$-vertex-membership function *and say pair $(\mathfrak{C}, \rho)$* solves *$\mathcal{P}$.*

Intuitively, a vertex membership function ρ together with a cliquewidth decomposition $\mathcal{D}$ of a graph G and a $(\mathfrak{C}, \mathcal{D})$-witness α provide an encoding of a subset of vertices of G using only the values of DP entries in leaves of $\mathcal{D}$. Let $S_\rho(\mathcal{D}, \alpha) = \{v \in V(G) : \rho(\alpha(Leaf_{\mathcal{D}}(v))) = 1\}$. Note that in this definition, we determine whether a vertex is in the subset or not based only on the value of the witness on its introduce node.

The following theorem is a formal reformulation of Theorem 1.3 formulated in the introduction. Indeed, if $t_{\mathcal{D}} = \mathcal{O}(Size_{\mathfrak{C}_i,\mathcal{D}})$ for $\mathcal{P}_i$ being the one with the slowest maximal running time on a single node of the decomposition out of $\mathcal{P}_1, \ldots, \mathcal{P}_r$, and assuming we are given $Accept_{\mathfrak{C}_i,D}$ and $Process_{\mathfrak{C}_i,D}(t)$ for each $t \in V(\mathcal{D})$ precomputed, the time needed to solve $Div^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ is $\mathcal{O}\left(|V(\mathcal{D})| \cdot \prod_{i=1}^{r} Size_{\mathfrak{C}_i,\mathcal{D}}\right) \leq \mathcal{O}(|V(\mathcal{D})| t_{\mathcal{D}}^r)$.

Theorem 2.6. *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems, $(\mathfrak{C}_i, \rho_i)$ be a pair of a monotone dynamic programming core and a vertex membership function that solves $\mathcal{P}_i$, Div be a Venn f-diversity measure for some $f : \{0,1\}^r \to \mathbb{N}$, $d \in \mathbb{N}$. The problem $Div^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on a graph G with given cliquewidth decomposition $\mathcal{D}$ can be solved in time*

$$\mathcal{O}\left(\sum_{i=1}^{r} \tau(\mathfrak{C}_i, \mathcal{D}) + |V(\mathcal{D})| \cdot \prod_{i=1}^{r} Size_{\mathfrak{C}_i,\mathcal{D}}\right).$$

Before proving it, we will also state the following theorem, which is a formal reformulation of Theorem 1.4 from the introduction. The proof of Theorem 2.6 can be easily modified to work for Theorem 2.7.

Theorem 2.7. *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems, $(\mathfrak{C}_i, \rho_i)$ be a pair of a monotone dynamic programming core and*

a vertex membership function that solves $\mathcal{P}_i$, $d \in \mathbb{N}$. The problem $DivMin^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on a graph G with given cliquewidth decomposition $\mathcal{D}$ can be solved in time

$$\mathcal{O}\left(\sum_{i=1}^{r} \tau(\mathfrak{C}_i, \mathcal{D}) + |V(\mathcal{D})| \cdot \prod_{i=1}^{r} Size_{\mathfrak{C}_i,\mathcal{D}} \cdot d^{r^2}\right).$$

Keep in mind that the following proof can be modified to work also for $DivMin$ as stated in Theorem 2.7. Instead of one diversity value ℓ for a tuple of partial solutions, we need to store a vector $L \in \{0, \ldots, d\}^{\binom{r}{2}}$ of pairwise Hamming distances between the solutions. Since for $S_1, S_2 \subseteq S$, $P_1, P_2 \subseteq P$, where S and P are disjoint, $HamDist(S_1 \cup P_1, S_2 \cup P_2) = HamDist(S_1, S_2) + HamDist(P_1, P_2)$, we can sum these vectors point-wise as we sum single diversity values. As the Hamming distance only grows, and we are interested in having the minimum distance of at least d, we can treat all the values greater than d as equal to d. However, we cannot take advantage of keeping only the best diversity value for a tuple of entries because such vectors may be incomparable, so for one tuple of entries that represents multiple tuples of partial solutions, there may be up to $(d+1)^{\binom{r}{2}}$ different diversity vectors, which contributes to d^{r^2} factor in Theorem 2.7.

Sketch of proof of Theorem 2.6. Full formal proof is in (Drabik and Masařík 2024). The correctness of the algorithm is based on the following lemma and the fact that for each node $t = t_1 \oplus t_2$ of the decomposition $V(G_{t_1}) \cap V(G_{t_2}) = \emptyset$.

Lemma 2.8. [1] *Let $S, P \subseteq V$ be disjoint sets, $S_1, \ldots, S_r \subseteq S$, $P_1, \ldots, P_r \subseteq P$ and let Div be a Venn f-diversity measure for some $f : \{0,1\}^r \to \mathbb{N}$. Then*

$$Div(S_1 \cup P_1, \ldots, S_r \cup P_r) = Div(S_1, \ldots, S_r) + Div(P_1, \ldots, P_r) - |V(G)| \cdot f(0^r).$$

We start by computing the tables $Process_{\mathfrak{C}_i,\mathcal{D}}$ and $Accept_{\mathfrak{C}_i,\mathcal{D}}$ corresponding to each core in total time $\sum_{i=1}^{r} \tau(\mathfrak{C}_i, \mathcal{D})$. Then for each node t of the decomposition, we construct by induction set $\Pi_{Div}(t)$ of tuples $(w^1, \ldots, w^r, \ell)$ such that for each i there exists $(\mathfrak{C}_i, \mathcal{D}_t, w^i)$-witness α_i, where ℓ is the diversity of the list of partial solutions defined by this tuple witnesses. If for some $(w^1, \ldots, w^r)$, there are multiple such tuples of witnesses, we keep only those that give the maximum value of the diversity. If t is a leaf of $\mathcal{D}$, each witness α_i consists only of $w^i \in Process_{\mathfrak{C}_i,\mathcal{D}}$. Observe that for any tuple $w = (w^1, \ldots, w^r)$, the only vertex whose membership in the list of partial solutions defined by this tuple can be nonzero is the vertex v introduced at t. Moreover, we can compute this vector m_w by simply applying the membership function. Then $\Pi_{Div}(t)$ consists of all such tuples w, together with the diversity value $\ell = f(m_w) + (|V(G)| - 1)f(0^r)$. If t has one child t_1, let $\Pi_{Div}(t)$ be the set of tuples $(w^1, \ldots, w^r, \ell)$ such that ℓ is the maximal value of ℓ_1 among all $(w_1^1, \ldots, w_1^r, \ell_1) \in \Pi_{Div}(t_1)$ where each $(w^i, w_1^i) \in Process_{\mathfrak{C}_i,\mathcal{D}}(t)$. If t has children t_1, t_2, let $\Pi_{Div}(t)$ be the set of tuples $(w^1, \ldots, w^r, \ell)$ such that ℓ is the maximal value of $\ell_1 + \ell_2 - |V(G)|f(0^r)$ among all pairs of $(w_1^1, \ldots, w_1^r, \ell_1) \in \Pi_{Div}(t_1), (w_2^1, \ldots, w_2^r, \ell_2) \in \Pi_{Div}(t_2)$ where each $(w^i, w_1^i, w_2^i) \in Process_{\mathfrak{C}_i,\mathcal{D}}(t)$. Note that we are using the monotonicity assumption in this argument. As we are joining pairs of partial solutions belonging to disjoint sets $V(G_{t_1}), V(G_{t_2})$, we can compute the diversity using Lemma 2.8. Observe that for each node t set $\Pi_{Div}(t)$ can be computed in time $\prod_{i=1}^{r} |Process_{\mathfrak{C}_i,\mathcal{D}}(t)|$. The last step is checking whether there exists $(w^1, \ldots, w^r, \ell) \in \Pi_{Div}(q)$ such that $\ell \geq d$ and each $w^i \in Accept_{\mathfrak{C}_i,\mathcal{D}}$ for q being the root of $\mathcal{D}$. This can be done in $\prod_{i=1}^{r} |Accept_{\mathfrak{C}_i,\mathcal{D}}|$ time.

We prove the correctness by combining the two lemmas formally stated and proven in (Drabik and Masařík 2024). Firstly, we show that for each $S_1, \ldots, S_r$ such that $(G, S_i) \in \mathcal{P}_i$ there exists $(w^1, \ldots, w^r, \ell) \in \Pi_{Div}(q)$ for some $\ell \geq Div(S_1, \ldots, S_r)$. Secondly, we show that for each $(w^1, \ldots, w^r, \ell) \in \Pi_{Div}(q)$ there exists $S_1, \ldots, S_r$ such that $(G, S_i) \in \mathcal{P}_i$ and $Div(S_1, \ldots, S_r) = \ell$. □

We would like to briefly explain at this point why the factor d^a is stated in the running time of Theorem 1.5. Integer a stands for the maximum number of children of a node in the given tree decomposition. The reason why this improvement is possible is that during the DP composition, we need to remember for each entry only the best value of diversity and not all d of them, similarly to our proof of Theorem 1.3. Such a change requires only a single computation for each table entry compared to the d^a factor used originally.

3 MSO_1 Has Monotone DP

Theorem 3.1 ((Courcelle, Makowsky, and Rotics 2000))**.** *Given a cliquewidth decomposition $\mathcal{D}$ of a graph G, an MSO_1 formula Φ can be evaluated on G in time $c \cdot |V(\mathcal{D})|$, where c is a constant depending only on Φ and the width of $\mathcal{D}$.*

The proof of Theorem 3.1 will give us dynamic programming over the decomposition of G. In Section 3.2, we will show that for vertex problems, this dynamic programming is monotone, which, combined with Theorem 2.6, will give us Corollary 1.6. Full formal proofs are in (Drabik and Masařík 2024).

3.1 Proof Sketch of Courcelle's Theorem for Cliquewidth Using Reduced Evaluation Trees

We prove Theorem 3.1 using *partial evaluation trees*. Let Φ be an MSO_1 formula in the prenex normal form with $q = q_v + q_s$ free variables, q_v of them being *individual variables*, and q_s being *set variables*. An *evaluation tree* for Φ and a graph G is a tree F of height q such that each node of depth ℓ corresponds to the assignment of the first ℓ variables (and is *labelled* by the value of the ℓ-th of them). The answer to whether $G \vDash \Phi$ can be obtained by traversing F bottom-up, where the value of a subtree in the evaluation

[1] Note that in order to use this lemma to compute diversity, we need to know the number of vertices of the final graph; $|V(G)|$. However, $f(0^r)$ corresponds to the influence of a vertex that is in none of the partial solutions, which could be set to 0 in many convenient measures. In such cases, we can even ignore this mild constraint.

depends on the values of its children. The size of the evaluation tree is exponential in the size of G, but it turns out that it can be reduced. Based on the observation that the value of the leaf depends only on the assignments of the variables restricted to the subgraph of G induced by the values of the individual variables corresponding to this leaf, we can define isomorphism classes of the leaves as pairs of such induced subgraph and variable assignment (there are at most $2^{\mathcal{O}(q_v q)}$ of them). The isomorphism classes of higher subtrees of F are defined inductively, as sets of isomorphism classes of their children. Proceeding bottom-up and reducing all children from the same class under the same parent to one fixed representative of this class, we get a *reduced evaluation tree* F^- which has the same value as F in the evaluation, and its size is bounded by $\left.2^{\cdot^{\cdot^{2^{\mathcal{O}(q_v q)}}}}\right\} q+1$.

A *partial evaluation tree* F_t for Φ and a node t of the decomposition $\mathcal{D}$ of G is the evaluation tree for Φ and G_t in which For individual variables, we consider the additional case when the value does not belong to G_t (we call such branch *external*). Observe that if we prune from F_t all the external branches, we get the evaluation tree for Φ and G_t. We can reduce partial evaluation trees similarly as regular evaluation trees, defining the isomorphism classes of the leaves (also called *configurations*) as combinations of: the colored subgraph G'_t of G_t induced by the values of individual variables corresponding to this leaf, assignment of the individual variables and assignment of the set variables restricted to $V(G'_t)$ (there are at most $2^{\mathcal{O}(q_v(q+\log\omega))}$ such combinations, where ω is the number of colors). Proceeding with the same bottom-up reduction, we get a *reduced partial evaluation tree* F_t^- of size bounded by $\left.2^{\cdot^{\cdot^{2^{\mathcal{O}(q_v(q+\log\omega))}}}}\right\} q+1$ which is isomorphic to F_t, so after pruning the external branches, it evaluates to the same value as the regular evaluation tree for G_t. It remains to prove that we can compute F_t^- for t being the root of $\mathcal{D}$ without computing F_t. The idea is to construct F_t^- inductively from $F_{t_1}^-, \ldots, F_{t_\delta(t)}^-$. Observe that for t being the leaf, the size of F_t is bounded, so we can compute F_t and reduce it. For $t = recolor(t_1, a \to b)$ or $t = addEdges(t_1, a, b)$, it suffices to reduce $F_{t_1}^-$, taking into account the new colored graph. For $t = t_1 \oplus t_2$ we compute a *tree product* $F_{t_1}^- \otimes F_{t_2}^-$ and then reduce it. The tree product of two partial evaluation trees F_1, F_2 for the same formula Φ and graphs G_1, G_2 is recursively defined as follows. If F_1, F_2 are leaves, $F_1 \otimes F_2 = (F_1, F_2)$. If the first free variable of Φ is a set variable, F_1 has children $\{F_1^i : i \in [n]\}$ corresponding, respectively, to the assignments of sets $\{S_i : i \in [n]\}$ to this variable, and similarly F_2 has children $\{F_2^j : j \in [m]\}$ corresponding, respectively, to the assignments of $\{P_j : j \in [m]\}$, then $F_1 \otimes F_2$ has children $\{F_1^i \otimes F_2^j : i \in [n], j \in [m]\}$ corresponding to the assignments of $\{S_i \cup P_j : i \in [n], j \in [m]\}$. If the first free variable of Φ is an individual variable, F_1 has children F_1^0 and $\{F_1^i : i \in [n]\}$ corresponding, respectively, to the assignments of external value and $\{v_i : i \in [n]\}$ to this variable, and similarly F_2 has children F_2^0 and $\{F_2^j : j \in [m]\}$ corresponding, respectively, to the assignments of external value and $\{u_j : j \in [m]\}$, then $F_1 \otimes F_2$ has children $F_1^0 \otimes F_2^0$, $\{F_1^i \otimes F_2^0 : i \in [n]\}$ and $\{F_1^0 \otimes F_2^j : j \in [m]\}$ respectively, to the assignments of external value, $\{v_i : i \in [n]\}$ and $\{u_j : j \in [m]\}$ to this variable. Note that $|F_1 \otimes F_2| \leq |F_1| \cdot |F_2|$. It is easy to see that for $t = t_1 \oplus t_2$, $F_t = F_{t_1} \otimes F_{t_2}$. Then, in order to show the correctness of the inductive construction of reduced partial evaluation trees, it remains to prove the following lemma, whose proof is (Drabik and Masařík 2024).

Lemma 3.2. *Let $t_1, t_2 \in V(\mathcal{D})$ such that G_{t_1}, G_{t_2} are disjoint, $F_1 \sim F_1'$ be partial evaluation trees for Φ and t_1, $F_2 \sim F_2'$ be partial evaluation trees for Φ and t_2. Then $F_1 \otimes F_2 \sim F_1' \otimes F_2'$.*

3.2 Generalization to Diverse Vertex Problems

We show that for any MSO_1 formula Φ expressing a vertex problem $\mathcal{P}$ the dynamic programming from Section 3.1 yields a core $\mathfrak{C}$ which (together with a proper membership function) solves $\mathcal{P}$ and $\tau(\mathfrak{C}, \mathcal{D}) \leq |V(\mathcal{D})| \cdot c$, $Size_{\mathfrak{C},\mathcal{D}} \leq c$ for some constant c depending on Φ and the width of $\mathcal{D}$. The following is a more explicit version of Corollary 1.6.

Corollary 3.3. *Let $\mathcal{P}_1, \ldots, \mathcal{P}_r$ be vertex problems expressible in MSO_1, Φ_i be an MSO_1 formula for $\mathcal{P}_i$, and Div be a Venn diversity function. The problem $Div^d(\mathcal{P}_1, \ldots, \mathcal{P}_r)$ on a graph G with given cliquewidth decomposition $\mathcal{D}$ can be solved in time $\mathcal{O}\left(|V(\mathcal{D})| \cdot \prod_{i=1}^r c_i\right)$, where c_i is a constant depending only on Φ_i and the width of $\mathcal{D}$.*

4 Conclusions and Open Problems

Our main contribution is that we can convert a monotone DP into one that solves a diverse version of the problem. We think it would be interesting to study this direction past the vertex problems. For example, can we extend the definition of monotonicity to edge problems? However, there is a barrier preventing FPT algorithms for many edge problems, as MSO_2 model checking is most likely not even in XP on cliques (Courcelle, Makowsky, and Rotics 2000). Still, a transformation of the original DP algorithm might exist. Indeed, this is in line with our approach, which also works independently of the running time of the original algorithm (of course, the final running time is dependent on it). Take, for example, the Hamiltonian cycle problem. There is an XP dynamic programming algorithm parameterized by the cliquewidth (Bergougnoux, Kanté, and Kwon 2019). Is there an XP algorithm solving the diverse variant of the Hamiltonian cycle problem parameterized by the cliquewidth? Another intriguing direction would be to research whether d^{r^2} overhead in running time comparison of Theorem 1.3 and Theorem 1.4 is necessary.

We believe that studying the Venn f-diversity measures is of independent interest. It would be nice to see some systematic study of differences in such measures or perhaps some experimental study that would give us more understanding of the various measures. Likewise, there has been a study comparing different diversity measures from the perspective of evolutionary computation (Wineberg and Oppacher 2003).

Acknowledgements

Both authors of this research were supported by the Polish National Science Centre grant number 2021/43/D/ST6/03312 (SONATA-17).

References

Ahanor, I.; Medal, H.; and Trapp, A. C. 2024. DiversiTree: A new method to efficiently compute diverse sets of near-optimal solutions to mixed-integer optimization problems. *INFORMS J. Comput.*, 36(1): 61–77.

Arrighi, E.; Fernau, H.; Lokshtanov, D.; Oliveira, M. d. O.; and Wolf, P. 2021. Diversity in Kemeny Rank Aggregation: A Parameterized Approach. In Zhou, Z.-H., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 10–16. International Joint Conferences on Artificial Intelligence Organization.

Arrighi, E.; Fernau, H.; Oliveira, M. d. O.; and Wolf, P. 2023. Synchronization and Diversity of Solutions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11516–11524. Association for the Advancement of Artificial Intelligence (AAAI).

Baste, J.; Fellows, M. R.; Jaffke, L.; Masařík, T.; de Oliveira Oliveira, M.; Philip, G.; and Rosamond, F. A. 2022. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artificial Intelligence*, 303: 103644.

Baste, J.; Jaffke, L.; Masařík, T.; Philip, G.; and Rote, G. 2019. FPT Algorithms for Diverse Collections of Hitting Sets. *Algorithms*, 12(12): 254:1–254:18.

Bergougnoux, B.; Dreier, J.; and Jaffke, L. 2023. A logic-based algorithmic meta-theorem for mim-width. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 3282–3304. Society for Industrial and Applied Mathematics. ISBN 9781611977554.

Bergougnoux, B.; and Kanté, M. 2019. Fast exact algorithms for some connectivity problems parameterized by clique-width. *Theoretical Computer Science*, 782: 30–53.

Bergougnoux, B.; Kanté, M. M.; and Kwon, O.-j. 2019. An Optimal XP Algorithm for Hamiltonian Cycle on Graphs of Bounded Clique-Width. *Algorithmica*, 82(6): 1654–1674.

Boehmer, N.; and Niedermeier, R. 2021. Broadening the Research Agenda for Computational Social Choice: Multiple Preference Profiles and Multiple Solutions. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '21)*, 1–5. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Bojikian, N.; and Kratsch, S. 2024a. Tight Algorithm for Connected Odd Cycle Transversal Parameterized by Clique-width. arXiv:2402.08046.

Bojikian, N.; and Kratsch, S. 2024b. A tight Monte-Carlo algorithm for Steiner Tree parameterized by clique-width. In Bringmann, K.; Grohe, M.; Puppis, G.; and Svensson, O., eds., *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 29:1–29:18. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-322-5.

Bui-Xuan, B.-M.; Suchý, O.; Telle, J. A.; and Vatshelle, M. 2013. Feedback vertex set on graphs of low clique-width. *European Journal of Combinatorics*, 34(3): 666–679.

Coudert, D.; Ducoffe, G.; and Popa, A. 2019. Fully Polynomial FPT Algorithms for Some Classes of Bounded Clique-width Graphs. *ACM Transactions on Algorithms*, 15(3): 1–57.

Courcelle, B.; Engelfriet, J.; and Rozenberg, G. 1993. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2): 218–270.

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

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

de Berg, M.; López Martínez, A.; and Spieksma, F. 2023. Finding Diverse Minimum s-t Cuts. In Iwata, S.; and Kakimura, N., eds., *34th International Symposium on Algorithms and Computation (ISAAC 2023)*, volume 283 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 24:1–24:17. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-289-1.

Do, A. V.; Guo, M.; Neumann, A.; and Neumann, F. 2022. Niching-based evolutionary diversity optimization for the traveling salesperson problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2022)*, GECCO 2022, 684–693. ACM.

Do, A. V.; Guo, M.; Neumann, A.; and Neumann, F. 2023. Diverse Approximations for Monotone Submodular Maximization Problems with a Matroid Constraint. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI-2023, 5558–5566. International Joint Conferences on Artificial Intelligence Organization.

Drabik, K.; and Masařík, T. 2024. Finding Diverse Solutions Parameterized by Cliquewidth. arXiv:2405.20931v2.

Fellows, M. R.; Rosamond, F. A.; Rotics, U.; and Szeider, S. 2009. Clique-Width is NP-Complete. *SIAM Journal on Discrete Mathematics*, 23(2): 909–939.

Fiala, J. 2024. Graph minors, decompositions and algorithms. Updated lecture notes; originally published in IUUK-CE-ITI series in 2003 number 2003-132.

Fomin, F. V.; Golovach, P. A.; Jaffke, L.; Philip, G.; and Sagunov, D. 2024a. Diverse pairs of matchings. *Algorithmica*, 86(6): 2026–2040.

Fomin, F. V.; Golovach, P. A.; Panolan, F.; Philip, G.; and Saurabh, S. 2024b. Diverse collections in matroids and graphs. *Mathematical Programming*, 204(1–2): 415–447.

Fomin, F. V.; and Korhonen, T. 2022. Fast FPT-approximation of branchwidth. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, 886–899. ACM.

Gabor, T.; Belzner, L.; Phan, T.; and Schmid, K. 2018. Preparing for the Unexpected: Diversity Improves Planning Resilience in Evolutionary Algorithms. In *2018 IEEE International Conference on Autonomic Computing (ICAC)*, 131–140. IEEE.

Galle, P. 1989. Branch & sample: A simple strategy for constraint satisfaction. *BIT*, 29(3): 395–408.

Glover, F.; Løkketangen, A.; and Woodruff, D. L. 2000. *Scatter Search to Generate Diverse MIP Solutions*, 299–317. Springer US. ISBN 9781461545675.

Grohe, M.; Kreutzer, S.; and Siebertz, S. 2017. Deciding First-Order Properties of Nowhere Dense Graphs. *Journal of the ACM*, 64(3): 1–32.

Hanaka, T.; Kiyomi, M.; Kobayashi, Y.; Kobayashi, Y.; Kurita, K.; and Otachi, Y. 2023. A Framework to Design Approximation Algorithms for Finding Diverse Solutions in Combinatorial Problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 3968–3976. Association for the Advancement of Artificial Intelligence (AAAI).

Hanaka, T.; Kobayashi, Y.; Kurita, K.; and Otachi, Y. 2021. Finding Diverse Trees, Paths, and More. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 3778–3786. Association for the Advancement of Artificial Intelligence (AAAI).

Hébrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proceedings of the 20th National Conference on Artificial Intelligence, AAAI'05*, volume 1, 372–377. AAAI Press. ISBN 157735236x.

Hegerfeld, F.; and Kratsch, S. 2023. Tight Algorithms for Connectivity Problems Parameterized by Clique-Width. In Gørtz, I. L.; Farach-Colton, M.; Puglisi, S. J.; and Herman, G., eds., *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 59:1–59:19. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-295-2.

Hébrard, E.; O'Sullivan, B.; and Walsh, T. 2007. Distance constraints in constraint satisfaction. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 106–111. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ingmar, L.; Garcia de la Banda, M.; Stuckey, P. J.; and Tack, G. 2020. Modelling Diversity of Solutions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02): 1528–1535.

Kiyomi, M.; and Otachi, Y. 2017. Alliances in graphs of bounded clique-width. *Discrete Applied Mathematics*, 223: 91–97.

Klute, F.; and van Kreveld, M. 2022. On Fully Diverse Sets of Geometric Objects and Graphs. In *Graph-Theoretic Concepts in Computer Science: 48th International Workshop, WG 2022, Tübingen, Germany, June 22–24, 2022, Revised Selected Papers*, 328–341. Springer International Publishing. ISBN 9783031159145.

Petit, T.; and Trapp, A. C. 2015. Finding Diverse Solutions of High Quality to Constraint Optimization Problems. In *International Joint Conference on Artificial Intelligence*.

Wineberg, M.; and Oppacher, F. 2003. The underlying similarity of diversity measures used in evolutionary computation. In *Genetic and Evolutionary Computation GECCO 2003*, Lecture notes in computer science, 1493–1504. Springer Berlin Heidelberg.