Deciding first-order properties of locally tree-decomposable structures

Markus Frick[*] Martin Grohe[†]

August 9, 2001

Abstract

We introduce the concept of a class of graphs, or more generally, relational structures, being *locally tree-decomposable*. There are numerous examples of locally tree-decomposable classes, among them the class of planar graphs and all classes of bounded valence or of bounded tree-width. We also consider a slightly more general concept of a class of structures having *bounded local tree-width*.

We show that for each property φ of structures that is definable in first-order logic and for each locally tree-decomposable class C of structures, there is a linear time algorithm deciding whether a given structure $\mathcal{A} \in$ C has property φ. For classes C of bounded local tree-width, we show that for every $k \geq 1$ there is an algorithm solving the same problem in time $O(n^{1+(1/k)})$ (where n is the cardinality of the input structure).

1. Introduction

It is an important task in the theory of algorithms to find feasible instances of otherwise intractable algorithmic problems. A notion that has turned out to be extremely useful in this context is that of *tree-width* of a graph. 3-COLORABILITY, HAMILTONICITY, and many other NP-complete properties of graphs can be decided in linear time when restricted to graphs whose tree-width is bounded by a fixed constant (see [8] for a survey).

Courcelle [9] proved a meta-theorem, which easily implies numerous results of the abovementioned type: *Let $w \geq 1$ and φ be a property of graphs that is definable in monadic second-order logic. Then φ can be decided in linear time on graphs of tree-width at most w.* As a matter of fact, this result does not only hold for graphs, but for arbitrary relational structures. Although Courcelle's theorem does not give practical algorithms, because the hidden constants are too big, it is still useful since it gives a simple way to recognize a property as being linear time decidable on graphs of bounded tree-width. Once this has been done, a more refined analysis using the combinatorics of the particular property may yield a practical algorithm.

Though maybe the most successful, bounded tree-width is not the only restriction on graphs that makes algorithmic tasks easier. Other useful restrictions are *planarity* or *bounded valence*. For example, consider the problem k-DOMINATING SET for a fixed k. (Given a graph $\mathcal{G}$, is there a set D of at most k vertices of $\mathcal{G}$ such that every vertex of $\mathcal{G}$ is either equal or adjacent to a vertex in D?) To solve k-DOMINATING SET in general, we do not know much better than just trying all $O(n^k)$ candidate sets (n always denotes the number of vertices of the input graph). However, on planar graphs k-DOMINATING SET can be solved in time $O(c^{\sqrt{k}}n)$ for a suitable constant c [3], and on graphs of valence at most l, it can be solved in time $O((l+1)^k n)$ [13].

Unfortunately, the analogue of Courcelle's theorem does not hold for planar graphs or classes of bounded valence; 3-COLORABILITY is a monadic second-order definable property that remains NP-complete when restricted to the class of planar graphs of valence at most 4 [21]. Instead of monadic second-order, we study the complexity of first-order definable properties.

A preliminary version of this article appeared in *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science 1644, pp.331-340. © Springer-Verlag 1999

[*]Institut für Mathematische Logik, Albert-Ludwigs-Universität Freiburg, Eckerstr. 1, 79104 Freiburg, Germany.
Email: frick@logik.mathematik.uni-freiburg.de.

[†]Laboratory for Foundations of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK.
Email: grohe@dcs.ed.ac.uk.

Seese was the first to give a meta-theorem in the style of Courcelle's theorem for a more general class of structures; in [31] he proved that for every $l \geq 1$ and for every first-order definable property of structures there is a linear time algorithm that decides whether a given structure of valence at most l has this property.

An observation that has been used for various algorithms on planar graphs (essentially it goes back to Baker [6]) is that there is a bound on the tree-width of a planar graph only depending on its diameter. A different way to see this is that a local neighborhood of a vertex in a planar graph has tree-width bounded by a number only depending on the radius of this neighborhood. As a matter of fact, given a planar graph $\mathcal{G}$ we can compute in linear time a family of subgraphs of bounded tree-width such that a suitably big neighborhood of every vertex of $\mathcal{G}$ is completely contained in one of these subgraphs.

We call classes of graphs admitting such a covering algorithm *locally tree-decomposable* (a precise definition is given in Section 6). Examples of locally tree-decomposable classes of graphs are all classes of bounded genus, bounded valence, and bounded tree-width. The concept easily generalizes to arbitrary relational structures.

Eppstein [16] considered a closely related, though slightly weaker concept he called the *diameter-treewidth property* (we call this property *bounded local tree-width* and refer the reader to Section 5 for the definition). Eppstein proved that the subgraph isomorphism problem for a fixed subgraph $\mathcal{H}$, asking whether a given graph $\mathcal{G}$ contains $\mathcal{H}$, is solvable in linear time when restricted to graphs $\mathcal{G}$ contained in a class of graphs that is closed under taking minors and has bounded local tree-width. It is not hard to see that every class C of graphs that is closed under taking minors and has bounded local tree-width is locally tree-decomposable (cf. Lemma 6.6).

Thus our main result goes much further:

Theorem 1.1. *Let* C *be a class of relational structures that is locally tree-decomposable and* φ *a property definable in first-order logic. Then there is a linear time algorithm deciding whether a given structure* $\mathcal{A} \in$ C *has property* φ.

It may be worth mentioning that in the terminology of [34], our result can be rephrased as follows: When restricted to a locally tree-decomposable class of structures, the *data complexity* of first-order logic is in linear time.

Examples of first-order definable properties are k-DOMINATING-SET and k-INDEPENDENT-SET for a fixed k, $\mathcal{H}$-SUBGRAPH-ISOMORPHISM (Given $\mathcal{G}$, is $\mathcal{H} \subseteq \mathcal{G}$?) and $\mathcal{H}$-HOMOMORPHISM (Given $\mathcal{G}$, is there a homomorphism $h : \mathcal{H} \to \mathcal{G}$?) for a fixed $\mathcal{H}$, $(\mathcal{H}, \mathcal{K})$-EXTENSION (Given $\mathcal{G}$, is every $\mathcal{H} \subseteq \mathcal{G}$ contained in some $\mathcal{K} \subseteq \mathcal{G}$?) for fixed $\mathcal{H} \subseteq \mathcal{K}$. Let us also give a few examples of a problems defined on other relational structures than graphs. For $k \geq 1$, k-SET-COVER is the problem of deciding whether a given family $\mathcal{F}$ of sets has a subfamily $\mathcal{S}$ of size at most k such that $\bigcup \mathcal{S} = \bigcup \mathcal{F}$. For $d \geq 1$, (k, d)-CIRCUIT-SATISFIABILITY is the problem of deciding whether a given Boolean circuit of depth at most d has a satisfying assignment in which at most k input gates are set to 'true'. Both k-SET-COVER and (k, d)-CIRCUIT-SATISFIABILITY can be seen as first-order definable problems on certain relational structures. Thus our theorem implies, for example, that k-SET-COVER can be solved in linear time for set systems where each element is only contained in a bounded number of sets and each set has bounded size, and that (k, d)-CIRCUIT-SATISFIABILITY can be solved in linear time for circuits whose underlying graph is planar. Of course problems like SUBGRAPH-ISOMORPHISM, HOMOMORPHISM, EXTENSION can be generalized to arbitrary relational structures.

As a last example, let us consider the problem of evaluating a (Boolean) database query formulated in the relational calculus against a relational database. Since relational calculus is the same as first-order logic, and relational databases are just finite relational structures, our theorem applies and shows, for example, that Boolean relational calculus queries can be evaluated in linear time on databases whose underlying graph is planar. As a matter of fact, this last example was one of our main motivation for starting this research. It seems that when storing geographical data such as road maps, planar structures come up quite naturally.

Thus our theorem gives a unifying framework for various results solving concrete problems on specific locally tree-decomposable classes such as the class of planar graphs. In addition, it yields a number of new results of this type.

Using the same techniques, we prove another theorem that applies to the even more general context of classes of structures of bounded local tree-width:

Theorem 1.2. *Let* C *be a class of relational structures of bounded local tree-width and* φ *a first-order definable property. Then for every* $k \geq 1$ *there is an algorithm deciding whether a given structure* $\mathcal{A} \in$ C *has property* φ *in time* $O(n^{1+(1/k)})$.

The complexity of first-order properties of relational structures has been studied under various aspects. It is long known that every first-order property of graphs can be decided in polynomial time, actually in AC_0 [2, 24]. A question closer to our theorem is whether deciding first-order properties is *fixed-parameter tractable*, that is, whether there is a fixed c such that every first-order property of finite relational structures can be decided in time $O(n^c)$. This question has been brought up by Yannakakis [35]. The theory of fixed-parameter tractability gives some evidence that the answer is no, as has been independently proved by Downey, Fellows, Taylor [14] and Papadimitriou, Yannakakis [26] (deciding first-order properties is AW[1]-complete). Theorem 1.2 shows that deciding first-order properties of structures in a class of bounded local tree-width is fixed-parameter tractable. Furthermore, it has been used in [19] to show that for every class C of graphs such that there is some graph that is not a minor of any graph in C, deciding first-order properties of graphs in C is fixed-parameter tractable.

The proofs of our results combine three main ingredients: a refinement of Courcelle's Theorem [9] mentioned above, Gaifman's Theorem [20] stating that first-order properties are local, and algorithmic techniques based on ideas of Baker [6] and Eppstein [16]. To prove Theorem 1.2, we also use covering techniques due to Awerbuch and Peleg [5, 27].

2. Preliminaries

A *vocabulary* is a finite set of relation symbols. Associated with every relation symbol R is a positive integer called the *arity* of R. In the following, E always denotes a binary relation symbol and τ a vocabulary.

A τ*-structure* $\mathcal{A}$ consists of a non-empty set A, called the *universe* of $\mathcal{A}$, and a relation $R^{\mathcal{A}} \subseteq A^r$ for each r-ary relation symbol $R \in \tau$. If $\mathcal{A}$ is a τ-structure and $B \subseteq A$, then $\langle B\rangle^{\mathcal{A}}$ denotes the substructure induced by $\mathcal{A}$ on B, that is, the τ-structure $\mathcal{B}$ with universe B and $R^{\mathcal{B}} := R^{\mathcal{A}} \cap B^r$ for every r-ary $R \in \tau$.

For instance, we consider *graphs* as $\{E\}$-structures $\mathcal{G} = (G, E^{\mathcal{G}})$, where the binary relation $E^{\mathcal{G}}$ is symmetric and anti-reflexive (that is, graphs are undirected and loop-free and they do not have multiple edges). As another example, we can view hypergraphs as $\{E, P\}$-structures, where E is binary and P unary. A *hypergraph* with vertices V and hyperedges $\mathcal{H} \subseteq \mathrm{Pow}(V)$ is modeled by the $\{E, P\}$-structure $\big(V \cup \mathcal{H}, \{(v, H) \mid v \in H\}, V\big)$. Occasionally, it is useful to consider graphs as hypergraphs and model them by the corresponding structures, then a graph $(G, E^{\mathcal{G}})$ corresponds to the $\{E, P\}$-structure $\big(G \cup E^{\mathcal{G}}, \{(v,(x,y)) \mid (x,y) \in E^{\mathcal{G}}, v \in \{x,y\}, G\big)$. This also enables us to include *multigraphs* (graphs with multiple edges) in our consideration.

In this paper we only consider finite structures. Let us remark that all the results of this paper remain true if we also admit constants in our structures. We restrict our attention to the relational case because constants would not give us additional insights.

The formulas of *first-order logic* FO are build up in the usual way from an infinite supply of variables denoted by $x, y, x_1, \dots$, the equality symbol $=$ and relation symbols of a vocabulary τ, the connectives $\wedge, \vee, \neg, \rightarrow$, and the quantifiers $\forall, \exists$ ranging over the universe of the structure. For example, the first-order sentence

$$\varphi := \forall x_1 \forall x_2 \forall x_3 \big((Ex_1x_2 \wedge Ex_1x_3 \wedge Ex_2x_3) \rightarrow \exists y(Ex_1y \wedge Ex_2y \wedge Ex_3y)\big)$$

says that every triangle of a graph is contained in a K_4 (a complete graph on four vertices). The formula

$$Px \wedge \neg\exists y \exists z(\neg y = z \wedge Exy \wedge Exz)$$

defines the set of all vertices x of a hypergraph that are contained in at most one hyperedge.

A *free variable* in a first-order formula is a variable x not in the scope of a quantifier $\exists x$ or $\forall x$. A *sentence* is a formula without free variables. The notation $\varphi(x_1, \dots, x_k)$ indicates that all free variables of the formula φ are among $x_1, \dots, x_k$; it does not necessarily mean that the variables $x_1, \dots, x_k$ all appear in φ. For a formula $\varphi(x_1, \dots, x_k)$, a structure $\mathcal{A}$, and $a_1, \dots, a_k \in A$ we write $\mathcal{A} \models \varphi(a_1, \dots, a_k)$ to say that $\mathcal{A}$ satisfies φ if the variables $x_1, \dots, x_k$ are interpreted by the vertices $a_1, \dots, a_k$, respectively.

Example 2.1. In this example we show how to model the k-SET-COVER problem mentioned in the introduction by a first-order definable problem. We can view a family $\mathcal{F}$ of a sets as the hypergraph with vertex set $\bigcup \mathcal{F}$ and edge set $\mathcal{F}$.

Let

$$\varphi_k := \exists x_1 \dots x_k \forall y \Big(Py \rightarrow \big(Eyx_1 \vee \dots \vee Eyx_k \big) \Big).$$

Then the hypergraph corresponding to the family $\mathcal{F}$ satisfies φ_k if and only if there exists an $\mathcal{S} \subseteq \mathcal{F}$ of cardinality $|\mathcal{S}| = k$ such that $\bigcup \mathcal{S} = \bigcup \mathcal{F}$.

We often denote tuples $(a_1, \dots, a_k)$ of elements of a set A by $\bar{a}$, and we write $\bar{a} \in A$ instead of $\bar{a} \in A^k$. Similarly, we denote tuples of variables by $\bar{x}$.

Our underlying model of computation is the standard RAM-model with addition and subtraction as arithmetic operations (cf. [1, 33]). In our complexity analysis we use the uniform cost measure. Structures are represented on a RAM in a straightforward way by listing all elements of the universe and then all tuples in the relations. For details we refer the reader to [18]. We define the *size* of a τ-structure $\mathcal{A}$ to be $||\mathcal{A}|| := |A| + \sum_{R \in \tau \text{ } r\text{-ary}} r \cdot |R^{\mathcal{A}}|$; this is the length of a reasonable representation of $\mathcal{A}$ (if we suppress details that are inessential for us).

3. Gaifman's Theorem

The *Gaifman graph* of a τ-structure $\mathcal{A}$ is the graph $\mathcal{G}_{\mathcal{A}}$ with vertex set $G_{\mathcal{A}} := A$ and an edge between two vertices $a, b \in A$ if $a \neq b$ and there exists an $R \in \tau$ and a tuple $(a_1, \dots, a_k) \in R^{\mathcal{A}}$ such that $a, b \in \{a_1, \dots, a_k\}$. The *distance* $d^{\mathcal{A}}(a, b)$ between two elements $a, b \in A$ of a structure $\mathcal{A}$ is the length of the shortest path in $\mathcal{G}_{\mathcal{A}}$ connecting a and b. For $r \geq 1$ and $a \in A$ we define the *r-neighborhood* of a in $\mathcal{A}$ to be $N_r^{\mathcal{A}}(a) := \{b \in A \mid d^{\mathcal{A}}(a, b) \leq r\}$. For a subset $B \subseteq A$ we let $N_r^{\mathcal{A}}(B) := \bigcup_{b \in B} N_r^{\mathcal{A}}(b)$.

For every $r \geq 0$ there is a first-order formula $\delta_r(x, y)$ such that for all τ-structures $\mathcal{A}$ and $a, b \in A$ we have $\mathcal{A} \models \delta_r(a, b) \iff d^{\mathcal{A}}(a, b) \leq r$. For example, if $\tau = \{E, T\}$ consists of a binary and a ternary relation symbol, we let

$$\begin{aligned}
\delta_0(x, y) &:= (x = y) \\
\delta_1(x, y) &:= \delta_0(x, y) \vee Exy \vee Eyx \vee \exists z \big(Txyz \vee Tyxz \vee Txzy \vee Tyzx \vee Tzxy \vee Tzyx \big) \\
\delta_2(x, y) &:= \delta_0(x, y) \vee \delta_1(x, y) \vee \exists z \big(\delta_1(x, z) \wedge \delta_1(z, y) \big)
\end{aligned}$$

In the following, we write $d(x, y) \leq r$ instead of $\delta_r(x, y)$ and $d(x, y) > r$ instead of $\neg \delta_r(x, y)$.

If $\varphi(x)$ is a first-order formula, then $\varphi^{N_r(x)}(x)$ is the formula obtained from $\varphi(x)$ by relativizing all quantifiers to $N_r(x)$, that is, by replacing every subformula of the form $\exists y \psi(x, y, \bar{z})$ by $\exists y (d(x, y) \leq r \wedge \psi(x, y, \bar{z}))$ and every subformula of the form $\forall y \psi(x, y, \bar{z})$ by $\forall y (d(x, y) \leq r \rightarrow \psi(x, y, \bar{z}))$. A formula $\psi(x)$ of the form $\varphi^{N_r(x)}(x)$, for some $\varphi(x)$, is called *r-local*. The basic property of r-local formulas $\psi(x)$ is that it only depends on the r-neighborhood of x whether they hold at x or not, that is, for all structures $\mathcal{A}$ and $a \in A$ we have $\mathcal{A} \models \psi(a) \iff \langle N_r^{\mathcal{A}}(a) \rangle \models \psi(a)$.

Theorem 3.1 (Gaifman [20]). *Every first-order sentence is equivalent to a Boolean combination of sentences of the form*

$$\exists x_1 \dots \exists x_k \big(\bigwedge_{1 \leq i < j \leq k} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \leq i \leq k} \psi(x_i) \big), \tag{3.1}$$

for suitable $r, k \geq 1$ and an r-local $\psi(x)$.

Remark 3.2. The best known upper bound for the quantifier-rank of a Boolean combination of sentences of the form (3.1) equivalent to a sentence φ of quantifier-rank q is 4^q [25]. There is no elementary upper bound for the size of a Boolean combination of sentences of the form (3.1) equivalent to a sentence φ in terms of the size of φ.

4. Tree-width

A *tree* is a connected acyclic graph $\mathcal{T}$. We always fix an (arbitrary) *root* $r^{\mathcal{T}} \in T$ in a tree $\mathcal{T}$. Then we have a natural partial order $\leq^{\mathcal{T}}$ on T, which is defined by

$$t \leq^{\mathcal{T}} u \iff t \text{ appears on the (unique) path from } r^{\mathcal{T}} \text{ to } u.$$

We say that u is a *child* of t or t is the *parent* of u if $(t, u) \in E^{\mathcal{T}}$ and $t \leq^{\mathcal{T}} u$.

A *tree-decomposition* of a τ-structure $\mathcal{A}$ is a pair $(\mathcal{T}, (B_t)_{t \in T})$, where $\mathcal{T}$ is a tree and $(B_t)_{t \in T}$ a family of subsets of A (called the *blocks* of the decomposition) such that

(1) For every $a \in A$, the set $\{t \in T \mid a \in B_t\}$ is non-empty and connected in $\mathcal{T}$ (that is, induces a subtree).

(2) For every $R \in \tau$ and all $\bar{a} \in R^{\mathcal{A}}$ there is a $t \in T$ such that $\bar{a} \in B_t$.

The *width* of a tree-decomposition $(\mathcal{T}, (B_t)_{t \in T})$ is $\max\{|B_t| \mid t \in T\} - 1$. The *tree-width* $\text{tw}(\mathcal{A})$ of $\mathcal{A}$ is the minimum taken over the widths of all tree-decompositions of $\mathcal{A}$.

It is easy to see that a structure $\mathcal{A}$ has the same tree-width as its Gaifman graph $\mathcal{G}_{\mathcal{A}}$. The reason for this is that if $(\mathcal{T}, (B_t)_{t \in T})$ is a tree-decomposition of a graph $\mathcal{G}$, then for every clique C of $\mathcal{G}$ there is a node t of $\mathcal{T}$ such that $C \subseteq B_t$ (see, e.g., [12]).

We occasionally use the following simple fact (cf. [30, 7]).

Lemma 4.1. *Let $w \geq 1$ and τ a vocabulary. Then for every τ-structure $\mathcal{A}$ of tree-width at most w and every r-ary relation symbol $R \in \tau$ we have $|R^{\mathcal{A}}| \leq (w+1)^r |A|$.*

Thus there is a constant c such that for every τ-structure $\mathcal{A}$ of tree-width at most w we have $||\mathcal{A}|| \leq c|A|$.

Proof: Let $\mathcal{A}$ be a τ-structure of tree-width w, and let $(\mathcal{T}, (B_t)_{t \in T})$ be a tree-decomposition of $\mathcal{A}$ of width at most w. For every tuple $\bar{a} = (a_1, \ldots, a_r) \in A^r$ we let $t(\bar{a})$ be the minimum node of $\mathcal{T}$ with respect to the tree-order $\leq^{\mathcal{T}}$ such that $a_1, \ldots, a_r \in B_t$, if such a node exists, otherwise we let $t(\bar{a})$ undefined. By the definition of tree-decompositions, $t(\bar{a})$ is defined for all $\bar{a} \in R^{\mathcal{A}}$. Letting $m_t := |\bar{a} \in A^r \mid t(\bar{a}) = t\}|$, we thus obtain $|R^{\mathcal{A}}| \leq \sum_{t \in T} m_t$.

For every vertex $t \in T$ we let

$$N_t := \begin{cases} B_t & \text{if } t \text{ is the root of } \mathcal{T} \\ B_t \setminus B_u & \text{if } t \text{ has a parent } u, \end{cases}$$

and $n_t := |N_t|$. Since every tuple $\bar{a} \in A^r$ with $t(\bar{a}) = t$ must contain at least one element of N_t, we obtain $m_t \leq (w+1)^r - (w+1-n_t)^r$. Since there are at most $|A|$ nodes t with $n_t = 0$, this yields

$$|R^{\mathcal{A}}| \leq \sum_{t \in T} m_t \leq |A|(w+1)^r.$$

□

Bodlaender [7] proved that for each $w \geq 1$ there is a linear time algorithm that, given a graph $\mathcal{G}$, either computes a tree-decomposition of $\mathcal{G}$ of width at most w, or rejects $\mathcal{G}$ if $\text{tw}(\mathcal{G}) > w$. This result is underlying most of the linear time algorithms on graphs of bounded tree-width. Using the well-known fact that a structure $\mathcal{A}$ has the same tree-width as its Gaifman graph $\mathcal{G}_{\mathcal{A}}$, Bodlaender's result can easily be extended to arbitrary relational structures.

Recall Courcelle's theorem that we mentioned in the introduction:

Theorem 4.2 (Courcelle [9]). *Let $w \geq 1$. Then for every sentence φ of monadic second-order logic there is a linear time algorithm that decides whether a given structure $\mathcal{A}$ of tree-width at most w satisfies φ.*

Monadic second-order logic is an extension of first-order logic that also allows quantification over sets of elements of the universe of a structure. If the structures we consider are graphs, we may also allow quantification over sets of edges. To see this, we can simply use the hypergraph eencoding of graphs.

Remark 4.3. There is a more general property of classes of graphs called *bounded clique-width* [10]. Theorem 4.2 would extend to classes of graphs of bounded clique-width if bounded clique-width decompositions of graphs of bounded clique-width could be computed in linear time [11], but it is not known whether this is possible. As a matter of fact, it is not even known whether there is a polynomial time algorithm deciding if a given graph has clique-width 4.

Using known techniques for algorithms on graphs of bounded tree-width, it is not hard to prove the following lemma (see [18]). We are only going to use the first-order version of the lemma later.

Lemma 4.4. *Let $w \geq 1$. Then for every formula $\varphi(x)$ of monadic second-order logic there is a linear time algorithm that, given a structure $\mathcal{A}$ of tree-width at most w, computes the set $\varphi(\mathcal{A}) := \{a \in A \mid \mathcal{A} \models \varphi(a)\}$.*

5. Local Tree-Width

Definition 5.1. (1) The *local tree-width* of a structure $\mathcal{A}$ is the function $\mathrm{ltw}^{\mathcal{A}} : \mathbb{N} \to \mathbb{N}$ defined by

$$\mathrm{ltw}^{\mathcal{A}}(r) := \max\big\{\mathrm{tw}(\langle N_r^{\mathcal{A}}(a)\rangle) \bigm| a \in A\big\}.$$

(Recall that $\langle N_r^{\mathcal{A}}(a)\rangle$ denotes the substructure induced by $\mathcal{A}$ on $N_r^{\mathcal{A}}(a)$.)

(2) A class C of structures has *bounded local tree-width* if there is a function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{ltw}^{\mathcal{A}}(r) \leq f(r)$ for all $\mathcal{A} \in$ C, $r \in \mathbb{N}$.

Example 5.2. *Structures of bounded tree-width.* Let $\mathcal{A}$ be a structure of tree-width at most k. Then $\mathrm{ltw}^{\mathcal{A}}(r) \leq k$ for all $r \in \mathbb{N}$.

The *valence* of a structure $\mathcal{A}$ is the maximal number of neighbors of a vertex $a \in A$ in the Gaifman graph $\mathcal{G}_{\mathcal{A}}$, i.e. $\max_{a\in A}|\{b \mid (a,b) \in E^{\mathcal{G}_{\mathcal{A}}}\}|$.

Example 5.3. *Structures of bounded valence.* Let $\mathcal{A}$ be a structure of valence at most ℓ, for an $\ell \geq 1$. Then $\mathrm{ltw}^{\mathcal{A}}(r) \leq \ell(\ell-1)^{r-1}$ for all $r \in \mathbb{N}$.

Example 5.4 (Robertson and Seymour [28]). *Planar Graphs.* The class of planar graphs has bounded local tree-width. More precisely, for every planar graph $\mathcal{G}$ and $r \geq 1$ we have $\mathrm{ltw}^{\mathcal{G}}(r) \leq 3r$.

Example 5.5 (Eppstein [16]). *Graphs of bounded genus.* Let S be a surface. Then the class of all graphs embeddable in S has bounded local tree-width. More precisely, there is a constant c such that for all graphs $\mathcal{G}$ embeddable in S and for all $r \geq 0$ we have $\mathrm{ltw}^{\mathcal{G}}(r) \leq c \cdot g(S) \cdot r$, where $g(S)$ denotes the genus of the surface S.

Recall that a *minor* of a graph $\mathcal{G}$ is a graph $\mathcal{H}$ that is obtained from a subgraph of $\mathcal{G}$ by contracting edges. The class of planar graphs, and, more generally, the classes of graphs of bounded genus are examples of classes of graphs that are closed under taking minors. Eppstein gave the following nice characterization of all classes of graphs of bounded local tree-width that are closed under taking minors. An *apex graph* is a graph $\mathcal{G}$ that has a vertex $v \in G$ such that $\langle G \setminus \{v\}\rangle^{\mathcal{G}}$ is planar.

Theorem 5.6 (Eppstein [15, 16]). *Let* C *be a class of graphs that is closed under taking minors. Then* C *has bounded local tree-width if, and only if,* C *does not contain all apex graphs.*

This yields further examples of classes of graphs of bounded local tree-width. For example, for every $n \geq 1$, the class of all graphs that do not contain the graph $K_{3,n}$ as a minor has bounded local tree-width. ($K_{m,n}$ denotes the complete bipartite graph with parts of size m and n, respectively.)

On the other hand, the class of all graphs of valence at most ℓ is *not* minor-closed for any $\ell \geq 3$. So the classes of Example 5.3 have bounded local tree-width, but are not minor closed.

Note that a structure has the same local tree-width as its Gaifman graph, so Examples 5.4 and 5.5 and Theorem 5.6 also give rise to examples of classes of structures of arbitrary vocabularies that have bounded local tree-width.

One of the nice things about bounded local tree-width is that the notion is quite flexible. Think of a structure modeling a subway map. The Gaifman graph of this structure will probably be close to planar, but there may be some edges crossing. Therefore, it may be the case that planar graph algorithms do not apply, although the graph is almost planar. On the other hand, the local tree-width of the graph is probably very close to that of a planar graph, and we can still use our algorithms for graphs of bounded local tree-width.

6. Neighborhood and tree covers

To explore the local tree-likeness of structures of bounded local tree-width we need to cover them by structures of small tree-width in a suitable way. The most general approach is to use *sparse neighborhood covers*, as they have been studied, for instance, in [5, 4, 27].

Definition 6.1. Let $r, s \geq 0$. An (r, s)-*neighborhood cover* of a structure $\mathcal{A}$ is a family $\mathcal{N}$ of subsets of A with the following properties:

(1) For every $a \in A$ there exists an $N \in \mathcal{N}$ such that $N_r^{\mathcal{A}}(a) \subseteq N$.

(2) For every $N \in \mathcal{N}$ there exists an $a \in A$ such that $N \subseteq N_s^{\mathcal{A}}(a)$.

We define the *size* of a family $\mathcal{N}$ of sets to be $||\mathcal{N}|| := \sum_{N \in \mathcal{N}} |N|$. Recall that the size of a τ-structure $\mathcal{A}$ is $||\mathcal{A}|| = |A| + \sum_{R \in \tau\ r-\text{ary}} r|R^{\mathcal{A}}|$. The algorithm of the following lemma is an adaptation of an algorithm due to Peleg [27] to our situation. We think it is worthwhile to present our version of the algorithm in some detail.

Lemma 6.2 (Peleg [27]). *Let $k \geq 1$. Then there is an algorithm that, given a graph $\mathcal{G}$ and an $r \geq 1$, computes an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{G}$ of size $||\mathcal{N}|| = O(|G|^{1+(1/k)})$ in time $O\big(\sum_{N \in \mathcal{N}} ||\langle N \rangle^{\mathcal{G}}||\big)$.*

Proof: The algorithm is described in Figure 1. It iteratively computes a neighborhood cover $\mathcal{N}$, maintaining a set H of vertices whose r-neighborhood has not yet been covered by a set in $\mathcal{N}$. In each iteration of the main loop in Lines 3–13, the algorithm picks an arbitrary vertex $a \in H$ and starts to compute increasing neighborhoods of a (in Lines 6–10) until a certain threshold is reached (cf. Line 10). Then it adds the computed set N to the cover $\mathcal{N}$ and removes all points whose neighborhood has now been covered from H, before it goes to the next iteration of the main loop. This process is repeated until H is empty.

To prove the correctness of the algorithm, let $\mathcal{G}$ be a graph, $n := |G|$, and $r \geq 1$. Let $\mathcal{N}$ be the cover computed by the algorithm.

Claim 1. For every $a \in G$ there exists an $N \in \mathcal{N}$ such that $N_r(a) \subseteq N$.

Proof: An element a is removed from the set H of uncovered elements in Line 12 if it belongs to a set L such that $N = N_r^{\mathcal{G}}(L)$ has been added to $\mathcal{N}$. Of course this N contains $N_r^{\mathcal{G}}(a)$. This proves Claim 1.

Claim 2. For every $N \in \mathcal{N}$ there exists an $a \in G$ such that $N \subseteq N_{2kr}^{\mathcal{G}}(a)$.

Proof: We consider the iteration of the main loop that leads to the definition of N. Let a be the element chosen in Line 4, and let $N_0 := \{a\}$. Let $l \geq 1$ be the number of times the loop in Lines 6–10 is repeated. For $1 \leq i \leq l$, let N_i be the value of N after the ith iteration. Then for $1 \leq i \leq l-1$ we have $|N_i| > n^{1/k}|N_{i-1}|$, and therefore $|N_i| > n^{i/k}$. Thus $l \leq k$.

Furthermore, it is easy to see that for $1 \leq i \leq l$ we have $N_i \subseteq N_{2ir}^{\mathcal{G}}(a)$. This implies Claim 2.

Claims 1 and 2 show that $\mathcal{N}$ is indeed an $(r, 2kr)$-neighborhood cover of $\mathcal{G}$. The following Claim 3 shows that the cover is not too large.

Claim 3. $||\mathcal{N}|| \leq n^{1+(1/k)}$.

Input: Graph $\mathcal{G}$, $r \geq 1$

1. $H := G$
2. $\mathcal{N} := \emptyset$
3. **while** $H \neq \emptyset$ **do**
4. choose arbitrary $a \in H$
5. $N := \{a\}$
6. **do**
7. $M := N$
8. $L := N_r^{\mathcal{G}}(M) \cap H$
9. $N := N_r^{\mathcal{G}}(L)$
10. **while** $|N| > n^{1/k}|M|$ **od**
11. $\mathcal{N} := \mathcal{N} \cup \{N\}$
12. $H := H \setminus L$
13. **od**

Output: $\mathcal{N}$

Figure 1.

Proof: For $N \in \mathcal{N}$, and let $M(N)$ be the corresponding set that is computed In Line 7 of the last iteration of the loop in Lines 6–10 that let to N (i.e. $M(N)$ is the value of N after the second but last iteration of the loop).

We first show that for distinct $N', N'' \in \mathcal{N}$ we have $M(N') \cap M(N'') = \emptyset$. To see this, suppose that N' is computed first. Let H' be the value of H after the iteration of the main loop in which N' has been computed. Note that by Lines 8 and 12 we have $H' \cap N_r^{\mathcal{G}}(M(N')) = \emptyset$. Thus

$$N_r^{\mathcal{G}}(H') \cap M(N') = \emptyset. \tag{6.1}$$

We claim that $M(N'') \subseteq N_r^{\mathcal{G}}(H')$. To prove this, we consider the iteration of the main loop where N'' is computed. Let H'' be the value of H in Line 3 of that iteration, that is, *before* N'' is computed. Then $H'' \subseteq H'$. Let $a'' \in H''$ be the element chosen in Line 4 and $N''_0 := \{a''\}$. Suppose the inner loop in Lines 6–10 is iterated $m \geq 1$ times, and, for $1 \leq i \leq m$, let N''_i be the value of N after the ith iteration. Then $N'' = N''_m$ and $M(N'') = N''_{m-1}$. An easy induction, which mainly uses Line 8, shows that for $0 \leq i \leq m$ we have $N''_i \subseteq N_r^{\mathcal{G}}(H'')$. Hence

$$M(N'') = N''_{m-1} \subseteq N_r^{\mathcal{G}}(H'') \subseteq N_r^{\mathcal{G}}(H'). \tag{6.2}$$

Together, (6.1) and (6.2) show that $M(N') \cap M(N'') = \emptyset$.

Noting that, by the condition of Line 10, for all $N \in \mathcal{N}$ we have $|N| \leq n^{1/k}|M(N)|$, we obtain

$$||\mathcal{N}|| = \sum_{N \in \mathcal{N}} |N| \leq n^{1/k} \sum_{N \in \mathcal{N}} |M(N)| \leq n^{1/k} \cdot n.$$

The last inequality holds because the sets $M(N)$, for $N \in \mathcal{N}$ are disjoint subsets of G (as we have seen). This proves Claim 3.

It remains to estimate the running time of the algorithm. We claim that each iteration of the main loop requires time $O(\langle N \rangle^{\mathcal{G}})$, for the N added to $\mathcal{N}$ in this iteration. To see this, note that essentially we have to do a breadth-first search on N starting in a. To compute L in Line 8, we may have to explore some edges not contained in $\langle L\rangle$. However, all these edges belong to $\langle N_r^{\mathcal{G}}(L)\rangle = \langle N\rangle$.

We store H as an array of size n; this enables us to compute the intersection in Line 8 using only a constant amount of time for each element of $N_r^{\mathcal{G}}(M)$. To be able to do the test $H \neq \emptyset$ in Line 3, we have to maintain the size of H', which can easily be done.

It may seem that to check the condition of Line 10 we need multiplication, which is not available as basic operation of a standard RAM. However, before we start the main computation we can produce tables that store the values m^k and $m^k \cdot n$ for $1 \leq m \leq n$ in linear time on a standard RAM. We proceed as follows: We first inductively compute m^l for $0 \leq l \leq k, 1 \leq m \leq n$, using the fact that

$$(m+1)^l = \sum_{(\epsilon_1,\dots,\epsilon_l)\in\{0,1\}^l} m^{\sum_{i=1}^{l}\epsilon_i}.$$

To go from m to $(m+1)$ we only need constant time, because we treat k as a constant. Then we compute $m^l \cdot n$ for $0 \leq l \leq k, 1 \leq m \leq n$ using the same induction with a different base case.

Once we have these tables, each time we have to check if $|N| > n^{1/k} \cdot |M|$ in Line 10 of the main algorithm, we look up the two values $|N|^k$ and $|M|^k \cdot n$ in our tables and test if $|N|^k - |M|^k \cdot n$ is greater than 0. This only requires constant time. □

Corollary 6.3. *Let $k, r \geq 1$, τ a vocabulary, and* C *a class of τ-structures of bounded local tree-width. Then there is an algorithm that, given a structure $\mathcal{A} \in$ C, computes an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{A}$ of size $||\mathcal{N}|| = O(|A|^{1+(1/k)})$ in time $O(|A|^{1+(1/k)})$.*

Proof: Since neighborhoods of radius $2kr$ in structures in C have bounded tree-width, by Lemma 4.1 there is a constant c such that for every structure $\mathcal{A} \in$ C, every $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{A}$, and every $N \in \mathcal{N}$ we have

$$||\langle N\rangle^{\mathcal{A}}|| \leq c|N|. \tag{6.3}$$

This implies $||\mathcal{A}|| \leq c||\mathcal{N}||$.

Our algorithm first computes the Gaifman graph $\mathcal{G}_{\mathcal{A}}$ of the input structure $\mathcal{A}$, which is possible in time $O(||\mathcal{A}||)$. Then it computes an an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{A}$ of size $||\mathcal{N}|| = O(|A|^{1+(1/k)})$. By Lemma 6.2 and (6.3), this is possible in time $O(||\mathcal{N}||) = O(|A|^{1+(1/k)})$. □

The following consequence of the proof of the previous corollary is worth being noted:

Corollary 6.4. *Let τ be a vocabulary and* C *be a class of τ-structures of bounded local tree-width. Then for every $k \geq 1$ there is a constant c such that for all structures $\mathcal{A} \in$ C we have $||\mathcal{A}|| \leq c|A|^{1+(1/k)}$.*

As a matter of fact, a neighborhood cover is more than we need. Often, the following weaker notion of a *tree cover* leads to better results.

Definition 6.5. Let $r, w \geq 0$. An (r, w)-*tree cover* of a structure $\mathcal{A}$ is a family $\mathcal{T}$ of subsets of A with the following properties:

(1) For every $a \in A$ there exists a $T \in \mathcal{T}$ such that $N_r^{\mathcal{A}}(a) \subseteq T$.
(2) For every $T \in \mathcal{T}$ we have $\mathrm{tw}(\langle T\rangle^{\mathcal{A}}) \leq w$.

Condition (2) in the definition of a neighborhood cover required the cover sets to have bounded diameter; for tree-coveres we replace this by bounded tree-width. Note that an (r, s)-neighborhood cover of a structure $\mathcal{A}$ is an $(r, \mathrm{ltw}^{\mathcal{A}}(s))$-tree cover of $\mathcal{A}$. The following lemma is implicit in [16]. In particular, this Lemma applies to the class of planar graphs and actually all classes of graphs of bounded genus.

Lemma 6.6 (Eppstein [16]). *Let $r \geq 0$ and* C *be a class of graphs that is closed under taking minors and has bounded local tree-width. Let $f : \mathbb{N} \to \mathbb{N}$ be a function bounding the local tree-width of the graphs in* C.

Then there is an algorithm that, given a graph $\mathcal{G} \in$ C, computes an $(r, f(2r+1))$-tree cover $\mathcal{T}$ of $\mathcal{G}$ of size $||\mathcal{T}|| = O(|G|)$ in time $O(|G|)$.

Proof: Let $\mathcal{G} \in \mathrm{C}$ and choose an arbitrary vertex $a_0 \in G$. For $0 \le i \le j$, let $G[i,j] := \{a \in G \mid i \le d^G(a_0, a) \le j\}$.

We claim that $\mathrm{tw}(\langle G[i,j] \rangle) \le f(j - i + 1)$. This is immediate if $i = 0$ or $i = 1$, because then $G[i,j] \subseteq N_j^{\mathcal{G}}(a_0)$. If $i > 1$, we simply contract the connected subgraph $\langle G[0, i-1] \rangle^{\mathcal{G}}$ to a single vertex b_0. We obtain a minor $\mathcal{G}'$ of $\mathcal{G}$, which is also an element of C by our assumption that C is closed under taking minors. $\mathcal{G}'$ still contains the set $G[i,j]$ as it is, but this set is contained in $N_{j-i+1}^{\mathcal{G}'}(b_0)$. This proves the claim.

The claim implies that for all $r \ge 0$, the family $\mathcal{T} := \{G[i, i+2r] \mid i \ge 0\}$ is an $(r, f(2r+1))$-tree cover of $\mathcal{G}$ of size at most $(2r+1)|G|$. On input $\mathcal{G}$, we can choose an arbitrary a_0 and then compute this tree cover in linear time by breadth-first search. $\Box$

The existence of a tree-cover of size linear in the size of the structure and a linear time algorithm computing such a cover is exactly what we need in our algorithms of the next section. This justifies the following definition:

Definition 6.7. A class C of structures is *locally tree-decomposable* if there are functions $f, g : \mathbb{N} \to \mathbb{N}$ and an algorithm that, given a structure $\mathcal{A} \in \mathrm{C}$ and an $r \in \mathbb{N}$, computes an $(r, g(r))$-tree cover of $\mathcal{A}$ in time at most $f(r) \cdot |A|$.

Observe that if a class C is locally tree-decomposable, then:

There exist functions $f, g : \mathbb{N} \to \mathbb{N}$ such that for every structure $\mathcal{A} \in \mathrm{C}$ and every $r \in \mathbb{N}$, there exists an $(r, g(r))$-tree cover $\mathcal{T}$ of $\mathcal{A}$ of size $||\mathcal{T}|| \le f(r) \cdot |A|$. (6.4)

This follows from the fact that no algorithm can compute a larger tree-cover in time $f(r) \cdot |A|$. Thus actually the definition of locally tree-decomposable combines two properties that also could be considered separately: The existence of linear-size tree-covers, as stated in (6.4), and the existence of an algorithm computing such covers. We decided to combine these two properties in just one definition, because we have already introduced an abundance of properties, and we do not know of any class of graphs that satisfies property (6.4), but is not locally tree-decomposable.[1]

Recall that a class C has bounded local tree-width if, and only, if:

There exists a function $g : \mathbb{N} \to \mathbb{N}$ such that for every structure $\mathcal{A} \in \mathrm{C}$ and every $r \in \mathbb{N}$, there exists an $(r, g(r))$-tree cover of $\mathcal{A}$. (6.5)

Thus any class that is locally tree-decomposable (or just satisfies (6.4)) has bounded local tree-width. Most natural classes of bounded local tree-width are also locally tree-decomposable:

Examples 6.8. *All examples of classes of structures of bounded local tree-width that we gave in Section 5 are actually locally tree-decomposable.*

For Example 5.3, classes of structures of bounded valence, this is trivial: If $\mathcal{A}$ is a structure of valence l and $r \ge 0$, then the family $\{N_r^{\mathcal{A}}(a) \mid a \in A\}$ is an $(r, l(l-1)^{r-1})$-tree cover of $\mathcal{A}$.

For all other examples, it follows from Lemma 6.6.

It is worth noting that we really need Lemma 6.6 (or some other argument) to show that certain classes of bounded local tree-width and unbounded degree are locally tree-decomposable, instead of just using the same trivial neighborhood covers $\{N_r^{\mathcal{A}}(a) \mid a \in A\}$, for $r \ge 1$, that we used for classes of bounded degree: Even for trees of unbounded degree, the size of these neighborhood covers cannot be bounded linearly in the structure size.

The following proposition is an immediate consequence of the definition of locally tree-decomposable classes of structures:

Proposition 6.9. *Let τ be a vocabulary and* C *be a locally tree-decomposable class of τ-structures. Then there is a constant c such that for all structures $\mathcal{A} \in \mathrm{C}$ we have $||\mathcal{A}|| \le c|A|$.*

[1] We conjecture that such a class exists, but doubt that it would be natural or give us any significant new insights.

Proof: Choose functions f, g according to (6.4). For $\mathcal{A} \in$ C, let $\mathcal{T}$ be an $(1, g(1))$-tree-cover of $\mathcal{A}$ of size $||\mathcal{T}|| \leq f(1) \cdot |A|$. Let $R \in \tau$ be r-ary and $\bar{a} = (a_1, \ldots, a_r) \in R^{\mathcal{A}}$. Since $a_1, \ldots, a_r \in N_1^{\mathcal{A}}(a_1)$, there exists a $T \in \mathcal{T}$ such that $a_1, \ldots, a_r \in T$. This implies that $|R^{\mathcal{A}}| \leq \sum_{T \in \mathcal{T}} |R^{\langle T \rangle^{\mathcal{A}}}|$ and thus

$$||\mathcal{A}|| \leq \sum_{T \in \mathcal{T}} ||\langle T \rangle^{\mathcal{A}}||.$$

By Lemma 4.1, we can choose $d > 0$ such that for every τ-structure $\mathcal{B}$ of tree-width at most $g(1)$ we have $||\mathcal{B}|| \leq d \cdot |B|$. We obtain

$$||\mathcal{A}|| \leq \sum_{T \in \mathcal{T}} ||\langle T \rangle^{\mathcal{A}}|| \leq \sum_{T \in \mathcal{T}} d \cdot |T| = d \cdot ||\mathcal{T}|| \leq d \cdot f(1) \cdot |A|.$$

□

We close this section with an example showing that the analogue of Proposition 6.9 for classes of bounded local tree-width is wrong. Remember Corollary 6.4, though.

Example 6.10. We construct a class C of graphs of bounded local tree-width such that for every constant c there is a graph $\mathcal{G} \in$ C with $||\mathcal{G}|| \geq c|G|$.

We use the following theorem due to Erdös [17]: *For all $g, k \geq 1$ there exists a graph of girth greater than g and chromatic number greater than k.* Remember that the *girth* $g(\mathcal{G})$ of a graph $\mathcal{G}$ is the length of the shortest cycle in $\mathcal{G}$ and the chromatic number $\chi(\mathcal{G})$ of $\mathcal{G}$ is the least number of colors needed to color the vertices of $\mathcal{G}$ in such a way that no two adjacent vertices have the same color. It is easy to see that every graph $\mathcal{G}$ with $\chi(\mathcal{G}) \geq k$ has a connected subgraph $\mathcal{H}$ with average degree

$$\frac{2|E^{\mathcal{H}}|}{|H|} \geq k - 1$$

(cf. [12], p. 98).

The *diameter* of a connected graph $\mathcal{G}$ is the number $\text{diam}(\mathcal{G}) := \max\{d^{\mathcal{G}}(a, b) \mid a, b \in G\}$.

We inductively construct a family $(\mathcal{G}_i)_{i \geq 1}$ of graphs as follows: $\mathcal{G}_1$ is the graph consisting of two vertices and an edge between them. Suppose now that $\mathcal{G}_i$ is already defined. Let $\mathcal{G}'_{i+1}$ be a graph with $g(\mathcal{G}'_{i+1}) \geq 2\text{diam}(\mathcal{G}_i) + 1$ and $\chi(\mathcal{G}'_{i+1}) \geq 2i + 3$. Let $\mathcal{G}_{i+1}$ be a connected subgraph of $\mathcal{G}'_{i+1}$ with

$$\frac{2|E^{\mathcal{G}_{i+1}}|}{|G_{i+1}|} \geq 2i + 2.$$

Clearly, $g(\mathcal{G}_{i+1}) \geq g(\mathcal{G}'_{i+1}) \geq 2\text{diam}(\mathcal{G}_i) + 1$. Since $\mathcal{G}_{i+1}$ cannot be a tree, this also implies $\text{diam}(\mathcal{G}_{i+1}) > \text{diam}(\mathcal{G}_i)$.

Observe that for every $r \geq 1$ and every graph $\mathcal{G}$, if $2r + 1 < g(\mathcal{G})$ then $\text{ltw}^{\mathcal{G}}(r) \leq 1$. Moreover, if $\mathcal{G}$ is connected then $\text{ltw}^{\mathcal{G}}(r) = \text{tw}(\mathcal{G})$ for all $r \geq \text{diam}(\mathcal{G})$. For every $i \geq 1$ and r with $\text{diam}(\mathcal{G}_i) \leq r < \text{diam}(\mathcal{G}_{i+1})$, we let

$$f(r) := \max\big(\{\text{ltw}^{\mathcal{G}_{i+1}}(r)\} \cup \{\text{tw}(\mathcal{G}_j) \mid 1 \leq j \leq i\}\big).$$

We claim that $\text{ltw}^{\mathcal{G}_i}(r) \leq f(r)$ for all $i, r \geq 1$. This is obvious for $i = 1$. For $i \geq 2$, we have to distinguish between three cases: If $r < \text{diam}(\mathcal{G}_{i-1}) \leq \frac{1}{2}(g(\mathcal{G}_i) - 1)$, then $\text{ltw}^{\mathcal{G}_i}(r) \leq 1 \leq f(r)$. If $\text{diam}(\mathcal{G}_{i-1}) \leq r < \text{diam}(\mathcal{G}_i)$, then $\text{ltw}^{\mathcal{G}_i}(r) \leq f(r)$ immediately by the definition of f. If $r \geq \text{diam}(\mathcal{G}_i)$, then $\text{ltw}^{\mathcal{G}_i}(r) = \text{tw}(\mathcal{G}_i) \geq f(r)$. Thus the class $\text{C} := \{\mathcal{G}_i \mid i \geq 1\}$ has bounded local tree-width.

Furthermore, for every $i \geq 2$ we have $||\mathcal{G}_i|| \geq |E^{\mathcal{G}_i}| \geq i|G_i|$.

7. An overview

In the previous two sections, we have described a confusing variety of properties of classes of graphs and structures. Figure 2 clarifies the relation between these properties. The *minor-closure* of a class C of graphs is the extension of C by all minors of graphs in C. *Non-trivial* simply means not the class of all graphs.

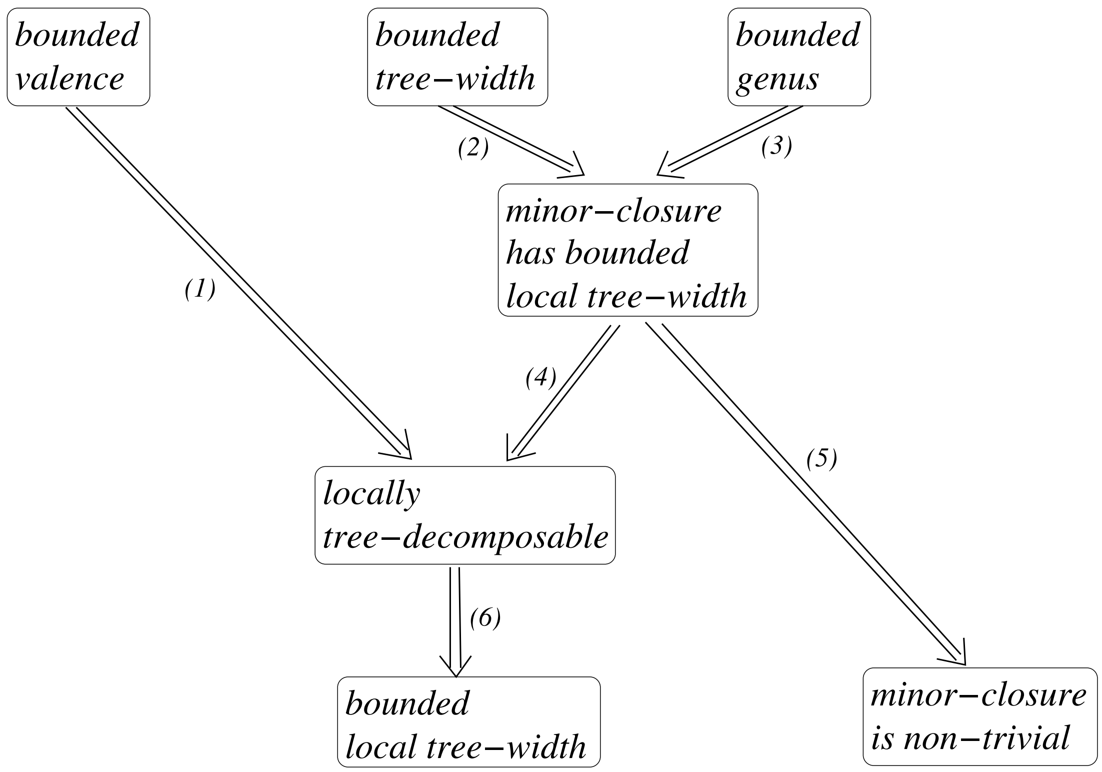

Figure 2. Properties of classes of graphs

Implication (1) is discussed in Examples 5.3 and 6.8. Implication (2) follows from Example 5.2 and the well-known fact that tree-width is minor-monotone (that is, the tree-width of a minor of a graph of tree-width k is at most k). Similarly, implication (3) follows from Example 5.5 and the fact that genus is minor-monotone. Implication (4) follows from Lemma 6.6. Implication (5) is trivial, and implication (6) follows from the discussion after Definition 6.7.

There hold no other relations between the different properties than those that can be derived from Figure 2 by transitivity; the following series of examples shows this:

Examples 7.1. (1) The class of all trees is minor closed and not trivial, but does not have bounded valence.

(2) The disjoint union of n 5-cliques has genus n. This can easily be seen using the additivity of the genus (cf. [32]). So the class of all graphs that are disjoint unions of 5-cliques has unbounded genus, but its tree-width and valence are 4.

(3) The class of all grids has unbounded tree-width [29], but is planar and of valence 4.

(4) If we replace each vertex v of a k-clique by a binary tree with k leaves and connect each leaf of the tree with one neighbor of v (in an arbitrary way), be obtain a graph of valence 3 with a k-clique minor. Thus the class of all graphs obtained in this way from cliques of arbitrary size has valence 3 and is thus locally tree-decomposable, but its minor closure is trivial (i.e., the class of all graphs).

(5) Example 6.10 shows that there are classes of graphs of bounded local tree-width that are not locally tree-decomposable.

(6) Recall the definition of apex-graphs from Page 6. Since planar graphs do not have 5-cliques as minors, apex-graphs do not have 6-cliques as minors. Thus class of all apex-graphs has a non-trivial minor closure, but by Theorem 5.6 it does not have bounded local tree-width.

(7) The class of all cliques has none of the properties displayed in Figure 2, but deciding a whether a given clique has some first-order property is easy; certainly it is possible in linear time in the size of the clique.

Remark 7.2. Although it plays no role in this article, we have included the property of having a non-trivial minor-closure here to give a more comprehensive picture. Let us first observe that the minor closure of a class C of graphs is non-trivial if, and only if, C has an *excluded minor*, that is, there is some graph that is not a minor of any graph in C. It is proved in [19] that for every class C of graphs with a non-trivial minor-closure there is a constant c such that for every sentence φ of first-order logic there is an algorithm deciding in time $O(n^c)$ whether a given structure $\mathcal{A} \in$ C satisfies φ.

Finally, it is worth mentioning that every class of graphs with a non-trivial minor-closure admits tree-decompositions whose blocks have "almost bounded-local tree-width" [22]. Thus there is an interesting property unifying bounded local tree-width and non-trivial minor closure.

8. The main algorithm

For every structure $\mathcal{A}$, subset $T \subseteq A$, and $r \geq 0$ we let the *r-kernel of T in $\mathcal{A}$* be the set

$$K_r^{\mathcal{A}}(T) := \{a \in A \mid N_r^{\mathcal{A}}(a) \subseteq T\}.$$

Lemma 8.1. *There is an algorithm that solves the following problem in time $O(r \cdot ||\langle T\rangle^{\mathcal{A}}||)$:*

Input:	Graph $\mathcal{G}$, subset $T \subseteq A$, $r \geq 1$.
Problem:	Compute $K_r^{\mathcal{G}}$.

Here we assume that $\mathcal{G}$ is given in adjacency list representation and that T is given as a linked list.

Proof: Since for all graphs $\mathcal{G}$, subsets $T \subseteq G$, and $i \geq 0$ we have

$$K_{i+1}^{\mathcal{G}}(T) = K_1^{\mathcal{G}}(K_i^{\mathcal{G}}(T)),$$

it suffices to show that, given $\mathcal{G}$ and $T \subseteq G$, we can compute $K_1^{\mathcal{G}}(T)$ in time $O(||\langle T\rangle^{\mathcal{G}}||)$.

Essentially, this is very easy. Let $\mathcal{G}$ be a graph and $T := \{a_1, \ldots, a_m\} \subseteq G$. We simply step through the adjacency lists of $a_1, \ldots, a_m$ and test for every edge if it points to an element of T or not. As soon as we find an edge out of a_i that points to an element outside of T, we discard a_i and continue with a_{i+1}. Then $K_1^{\mathcal{G}}(T)$ is the set of all vertices that are not discarded in this procedure.

To analyse the runtime of this procedure, we first note that for each a_i, $1 \leq i \leq m$, we have to process at most one edge that points to an element outside of T, i.e., that is not an edge of $\langle T\rangle$. Thus we have to process at most $m + |E^{\langle T\rangle}| \leq ||\langle T\rangle||$ edges. For each of these edges, we have to check if it points to an element of T or not. We shall now explain how to set up a data structure that enables us to do this in constant time.

To be able to do this, we assume without loss of generality that the vertex set of our input graph $\mathcal{G}$ is $\{1, \ldots, n\}$. (It is explained in the appendix of [18] why we can make this assumption). We represent T by an array AT of size n. Since T may be much smaller than n, we cannot intialize the whole array, for example, in such a way that $AT[i] = 1$ if $i \in T$ and $AT[i] = 0$ otherwise. Instead, we maintain a second "control array" Γ of length m. The jth entry of Γ is initialized to a_j, for $j = 1$ to m. Moreover, for $1 \leq i \leq m$ we intialize AT$[a_i]$ to i. Both of these initializations only require time $O(m)$. Then an element $b \in G$ is contained in T if, and only if,

$$\text{AT}[b] \in \{1, \ldots, m\} \text{ and } \Gamma[\text{AT}[b]] = b.$$

Once the two arrays are set up, this condition can be tested in constant time.

Figure 3 shows the full algorithm. □

For the rest of this section, we fix a vocabulary τ.

Input: $\mathcal{G}, T = \{a_1, \dots , a_m\} \subseteq G$

1. **for** $j = 1$ **to** m **do** $\Gamma[j] := a_j$ **od**
2. **for** $j = 1$ **to** m **do** AT$[a_j] := j$ **od**
3. $K := \emptyset$
4. **for** $j = 1$ **to** m **do**
5.   **for all** neighbors b of a_j **do**
6.   **if** AT$[b] \notin \{1, \dots , m\}$ or $\Gamma[$AT$[b]] \neq b$ **then**
7.   break out of inner loop and continue in Line 4
8.   **fi**
9.   **od**
10. $K := K \cup \{a_i\}$
11. **od**

Output: K

Figure 3.

Corollary 8.2. *Let* C *be a class of* τ*-structures of bounded local tree-width and* $r, w \geq 1$. *Then there is an algorithm that solves the following problem in time* $O(||\mathcal{T}||)$*:*

Input:	Structure $\mathcal{A} \in$ C, (r, w)-tree cover $\mathcal{T}$ of $\mathcal{A}$.
Problem:	Compute $K_r^{\mathcal{A}}$ for all $T \in \mathcal{T}$.

Proof: Given $\mathcal{A}$ and $\mathcal{T}$, we first compute the Gaifman graph $\mathcal{G}$ of $\mathcal{A}$. Since the vocabulary τ is fixed, this is possible in time $O(||\mathcal{A}||) = O(||\mathcal{G}||)$. Then for every $T \in \mathcal{T}$, we compute $K_r^{\mathcal{G}}(T) = K_r^{\mathcal{A}}(T)$, by Lemma 8.1 is is possible in time $O(||\langle T\rangle^{\mathcal{G}}||)$.

Thus overall we need time

$$O(||\mathcal{A}||) + \sum_{T \in \mathcal{T}} O(||\langle T\rangle^{\mathcal{G}}||).$$

Noting that $||\mathcal{A}|| \in O(||\mathcal{G}||) \subseteq \sum_{T\in\mathcal{T}} O(||\langle T\rangle^{\mathcal{G}}||)$, this can be simplified to $\sum_{T\in\mathcal{T}} O(||\langle T\rangle^{\mathcal{G}}||)$ Since for each $T \in \mathcal{T}$, $\langle T\rangle^{\mathcal{G}}$ is a graph of tree-width at most w, so by Lemma 4.1 we have $||\langle T\rangle^{\mathcal{G}}|| \in O(|T|)$. Thus $\sum_{T\in\mathcal{T}} O(||\langle T\rangle^{\mathcal{G}}||) = O\big(\sum_{T\in\mathcal{T}} |T|\big) = O(||\mathcal{T}||)$. $\square$

Lemma 8.3. *Let* C *be a class of structures of bounded local tree-width and* $r, m \geq 1$. *Then the following problem can be solved in time* $O(|A|)$*:*

Input:	Structure $\mathcal{A} \in$ C, set $P \subseteq A$.
Problem:	Decide if there exist $a_1, \dots , a_m \in P$ such that $d^{\mathcal{A}}(a_i, a_j) > r$ for all i, j with $1 \leq i < j \leq m$.

Proof: Let $\mathcal{A} \in$ C and $P \subseteq A$. Our algorithm is displayed in Figure 4. It proceeds in two phases.

In the first phase (Lines 1–12) it iteratively computes elements $a_1, \dots , a_i \in P$, for some $i \leq m$, such that $d^{\mathcal{A}}(a_i, a_j) > r$ for $1 \leq i < j \leq l$ and either $l = m$ or for all $b \in P$ there is an $i \leq l$ such that

Input: $\mathcal{A} \in$ C, $P \subseteq A$

1. $Q := P$
2. $l := 0$
3. **while** $Q \neq \emptyset$ **and** $l < m$ **do**
4. $\quad$ $l := l + 1$
5. $\quad$ choose $a_l \in Q$ arbitrarily
6. $\quad$ $Q := Q \setminus N_r^{\mathcal{A}}(a_l)$
7. **od**
8. **if** $l = m$ **then**
9. $\quad$ ACCEPT
10. **else**
11. $\quad$ **if** $l = 0$ **then** REJECT **fi**
12. **fi**
13. compute $H := N_{2r}^{\mathcal{A}}(\{a_1, \dots, a_l\})$
14. **if** $\big(\langle H \rangle^{\mathcal{A}}, P\big) \models \exists x_1 \dots \exists x_m \Big(\bigwedge_{i=1}^{m} Px_i \wedge \bigwedge_{1 \leq i < j \leq m} d(x_i, x_j) > r \Big)$ **then**
15. $\quad$ ACCEPT
16. **else**
17. $\quad$ REJECT
18. **fi**

Figure 4.

$b \in N_r^{\mathcal{A}}(a_i)$. If $l = m$, the algorithm accepts. If $l = 0$, i.e. $P = \emptyset$, then it rejects. Otherwise, it goes into the second phase (Lines 13–18).

When the algorithm enters Line 13, we have $P \subseteq N_r^{\mathcal{A}}(\{a_1, \dots, a_l\})$. Let $\mathcal{H} := \langle N_{2r}^{\mathcal{A}}(\{a_1, \dots, a_l\})$. Then for all $b, b' \in P$ we have $d^{\mathcal{A}}(b, b') \leq r \iff d^{\mathcal{H}}(b, b') \leq r$, because $P \subseteq N_r^{\mathcal{A}}(\{a_1, \dots, a_l\})$ and thus every path of length at most r between two elements of P must be contained in H. Thus there exist $b_1, \dots, b_m \in P$ such that $d^{\mathcal{A}}(b_i, b_j) > r$ if, and only if, there exist $b_1, \dots, b_m \in P$ such that $d^{\mathcal{H}}(b_i, b_j) > r$, i.e. if the condition in Line 14 is satisfied. Thus the algorithm is correct

To estimate the running time, we note that $||\langle N_r^{\mathcal{A}}(a_i)\rangle^{\mathcal{A}}|| = O(|N_r^{\mathcal{A}}(a_i)|)$, because C is a class of bounded local tree-width. Since we treat r and m as constants, Lines 1–13 require time $O(|A|)$. It is easy to see that $\mathrm{tw}(\mathcal{H}) \leq \mathrm{ltw}^{\mathcal{A}}(2lr)$. Since C is a class of bounded local-tree width, this gives us a bound on $\mathrm{tw}(\mathcal{H})$ only depending on the constants l and r. Thus the condition in Line 14 can also be checked in time $O(|H|) \subseteq O(|A|)$ by Courcelle's Theorem 4.2. $\square$

We are now ready to prove our main results, Theorems 1.1 and 1.2. Recall the statements:

Let C *be a class of structures of bounded local tree-width and* φ *a sentence of first-order logic.*

(1) For every $k \geq 1$ *there is an algorithm that decides whether a given structure* $\mathcal{A} \in$ C *satisfies* φ *in time* $O(|A|^{1+(1/k)})$.

(2) If C *is locally tree-decomposable, then there is an algorithm that solves the problem in time* $O(|A|)$.

Proof: We describe the algorithm for (1) and then explain how it has to be modified to obtain (2).

Recall that by Gaifman's Theorem 3.1, every first-order sentence is equivalent to a Boolean combination of sentences of the form

$$\exists x_1 \dots \exists x_m \big(\bigwedge_{1 \le i < j \le m} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \le i \le m} \psi(x_i) \big), \tag{8.1}$$

for suitably chosen $r, m \ge 1$ and an r-local ψ. Thus without loss of generality we can assume that φ is of the form (8.1), because if we can check in linear time for each sentence in a finite collection $\varphi_1, \dots, \varphi_\ell$ if it holds in a structure $\mathcal{A}$ or not, and then we can also check if a particular Boolean combination of $\varphi_1, \dots, \varphi_\ell$ holds in $\mathcal{A}$ or not.

Let $k \ge 1$ and $f : \mathbb{N} \to \mathbb{N}$ be a function bounding the local tree-width of the structures in C. Let τ be the vocabulary of the sentence φ; without loss of generality we can assume that all structures in C are τ-structures.

Figure 5 shows our algorithm.

Input: Structure $\mathcal{A} \in$ C

1. compute an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of $\mathcal{A}$ of size $O(A^{1+(1/k)})$
2. **for all** $N \in \mathcal{N}$ **do**
3.   compute $K_r^{\mathcal{A}}(N)$
4. **od**
5. **for all** $N \in \mathcal{N}$ **do**
6.   compute $P_N := \big\{ a \in K_r^{\mathcal{A}} \bigm| \langle N \rangle^{\mathcal{A}} \models \psi(a) \big\}$.
7. **od**
8. compute $P := \bigcup_{N \in \mathcal{N}} P_N$
9. **if** there are $a_1, \dots, a_m \in P$ such that $d(a_i, a_j) > 2r$ for $1 \le i < j \le k$ **then**
10.   ACCEPT
11. **else**
12.   REJECT
13. **fi**

Figure 5.

To see that the algorithm is correct, note that since $\psi(x)$ is r-local we have $P = \{a \in A \mid \mathcal{A} \models \psi(a)\}$.

So we shall prove that the algorithm can be implemented as an $O(n^{1+(1/k)})$-algorithm, where $n := |A|$ is the cardinality of the input structure.

Line 1 requires time $O(n^{1+(1/k)})$ by Corollary 6.3. Lines 2–4 require time $O(||\mathcal{N}||)$ by Corollary 8.2, because a $(r, 2kr)$-neighborhood cover of a structure $\mathcal{A} \in$ C is also an $(r, f(2kr))$ tree-cover. For every $N \in \mathcal{N}$, Line 6 requires time $O(|N|)$ by Lemma 4.4. Thus the loop in Lines 5–7 also requires time $O(||\mathcal{N}||)$. Clearly, Line 8 can be performed in time $O(||\mathcal{N}||)$, and the condition in Line 9 can be checked in time $O(|A|)$ by Lemma 8.3. Thus the overall running time is $O(||\mathcal{N}||) = O(n^{1+(1/k)})$.

It remains to prove (2), but this is very easy now. Instead of a neighborhood cover, in Line 1 of the algorithm we compute a tree cover of linear size. This can be done in linear time by the definition of local tree-decomposability. Since the running time of the rest of the algorithm is linear in the size of the cover, we obtain a linear time algorithm. $\square$

9. Concluding remarks

Uniformity. A close look at our proofs shows that actually for each locally tree-decomposable class C of structures there is a recursive function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm that decides, given a first-order sentence φ and a structure $\mathcal{A} \in$ C, whether $\mathcal{A} \models \varphi$ in time $O(f(||\varphi||)|A|)$, where $||\varphi||$ denotes the length of the sentence φ. We can obtain an anlogous uniform version of Theorem 1.2.

We stated and proved non-uniform versions of the theorems for the sake of a clearer presentation.

Dependence on the formula size. There is no elementary lower bound on the runtime of the algorithm in terms of the size of the formula φ. There are two main factors contributing to this high complexity: The first is the translation of a first-order formula into a formula in Gaifman's normal form. We have mentioned in Remark 3.2 that this translation causes an enourmous blow-up in the formula size. But even for formulas that already are in the normal form (8.1), the runtime of our algorithm is non-elementary in the formula size, because we still have to use Courcelle's algorithm for evaluating the formulas locally. Here the dependence of the runtime on the formula size is roughly q-fold exponential in the length of φ, where q is the number of quantifier-alternations in φ. The intuitive explanation for this is that we have translate the formula into an equivalent tree-automaton, and every quantifier alternation requires turning a non-deterministic into a deterministic automaton, which cause an exponential blow-up in the size.

Practical Considerations. The large hidden constants seem to make our algorithms useless for practical purposes. Nevertheless, let us briefly discuss a few more practical aspects.

If we think of a database application, we will usually only have to handle very simple formulas. As matter of fact, most database queries are so called *conjunctive queries*; they can be defined by first-order formulas of the form $\exists x_1 \dots \exists x_k(\alpha_1 \wedge \dots \wedge \alpha_m)$. Such formulas do not have any quantifier alternations. Moreover, for conjunctive queries, we can avoid Gaifman's theorem and instead use algorithmic techniques similar to those in [15]. With these techniques, the dependence on the formula size can be reduced to a singly exponential factor, which seems acceptable because usually in practice we have to evaluate small formulas (queries) in large structures (databases).

Another expensive subroutine of our algorithms is the computation of tree-decompositions, which are needed to evaluate the formulas locally. While currently there is no good algorithm to compute bounded width decompositions of arbitrary graphs of bounded tree-width, there are simple and efficient algorithms computing tree-decompositions of planar graphs of bounded radius. This means that at least we get a practical algorithm for evaluating conjunctive queries on planar graphs.

Nevertheless, our results are mostly theoretical. Similarly to Courcelle's Theorem [9], their main benefit is to provide a quick and simple way to recognize a property as being linear time computable on certain classes of graphs. Analyzing the combinatorics of the specific property then, one may also find a practical algorithm.

Further Research. Although Example 6.10 shows that for classes C of bounded local tree-width we cannot expect an algorithm deciding a first-order property of structures $\mathcal{A} \in$ C in time $O(|A|)$, it does not rule out an $O(||\mathcal{A}||)$-algorithm. To obtain such an algorithm, it would be sufficient to find, for every $r \geq 1$, a $w \geq 1$ and an $O(||\mathcal{A}||)$-algorithm that computes an (r, w)-tree cover of a structure $\mathcal{A} \in$ C.

As we mentioned, one of the main factors contributing to the heavy dependence of the running time of our algorithms on the size of the formula is the transformation into a "local formula" according to Gaifman's theorem. Though this transformation is clearly effective, as far as we know its complexity has not been studied. We do expect this complexity to be non-elementary, but this does not rule out the existence of more efficient algorithms for particular classes of formulas or the existence of good heuristics. It has been recently been shown [23] that existential first-order formulas can be translated into existential Gaifman normal form sentences, this implies that for existential sentences the dependence of the runtime of the algorithm on the formula size can be reduced to doubly exponential.

In general, we consider it as one of the main challenges for further research to reduce the dependence on the formula size (not only in our results, but also in Courcelle's theorem). For example, is there an algorithm that decides, given a first-order sentence φ and a planar graph $\mathcal{G}$ (or a tree, or just a word), whether $\mathcal{G} \models \varphi$ in time $O(2^{|\varphi|}n^c)$ for some fixed-constant c?

Another direction for further research is to find necessary conditions on classes of structures such that first-order properties or monadic second-order properties can be decided in linear time or are fixed-parameter tractable. It is an easy consequence of Courcelle's Theorem 4.2 and Robertson and Seymour's Excluded Grid Theorem [29] that, unless P $=$ NP, for every minor-closed class C of graphs we have: There is a $c \geq 1$ such that monadic-second order properties of graphs in C can be decided in time $O(n^c)$ if, and only if, C has bounded tree-width. But this is unsatisfying, because being minor-closed is a strong assumption. And clearly there are classes of graphs of unbounded tree-width on which monadic-second-order properties can be decided in linear time, the simplest example being the class of all cliques. Recalling Remark 4.3, we find that the classes of graphs of bounded clique-width may be further examples. It would be very interesting to get a clearer picture here.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] A.V. Aho and J.D. Ullman. The universality of data retrieval languages. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 110–120, 1979.

[3] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: Exponential speed-up for planar graph problems. In F. Orejas, P.G. Spirakis, and J. van Leeuwen, editors, *28th International Colloquium on Automata, Languages and Programming (ICALP 2001)*, volume 2076 of *Lecture Notes in Computer Science*, pages 261–272. Springer-Verlag, 2001.

[4] B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 638–647, 1993.

[5] B. Awerbuch and D. Peleg. Sparse partitions. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990.

[6] B.S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180, 1994.

[7] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[8] H.L. Bodlaender. Treewidth: Algorithmic techniques and results. In I. Privara and P. Ruzicka, editors, *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97*, volume 1295 of *Lecture Notes in Computer Science*, pages 29–36. Springer-Verlag, 1997.

[9] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, pages 194–242. Elsevier Science Publishers, 1990.

[10] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46:218–270, 1993.

[11] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[12] R. Diestel. *Graph Theory*. Springer-Verlag, second edition, 2000.

[13] R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[14] R.G. Downey, M.R. Fellows, and U. Taylor. The parameterized complexity of relational database queries and an improved characterization of $W[1]$. In Bridges, Calude, Gibbons, Reeves, and Witten, editors, *Combinatorics, Complexity, and Logic – Proceedings of DMTCS '96*, pages 194–213. Springer-Verlag, 1996.

[15] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3:1–27, 1999.

[16] D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291, 2000.

[17] P. Erdös. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959.

[18] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. Currently available at `http://www.dcs.ed.ac.uk/~grohe`. A preliminary version of the paper appeared in *Proceedings of the 8th International Conference on Database Theory*, LNCS 1973, Springer-Verlag, 2001.

[19] J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model checking. *SIAM Journal on Computing*, 31(1):113–145, 2001.

[20] H. Gaifman. On local and non-local properties. In *Proceedings of the Herbrand Symposium, Logic Colloquium '81*. North Holland, 1982.

[21] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[22] M. Grohe. Local tree-width, excluded minors, and approximation algorithms. To appear in *Combinatorica*. Currently available at `http://www.dcs.ed.ac.uk/~grohe`.

[23] M. Grohe and S. Wöhrle. An existential locality theorem. In *Computer Science Logic, 15th International Workshop CSL'01*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[24] N. Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.

[25] S. Lifsches and S. Shelah. Distorted sums of models. Unpublished Manuscript.

[26] C.H. Papadimitriou and M. Yannakakis. On the complexity of database queries. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, pages 12–19, 1997.

[27] D. Peleg. Distance-dependent distributed directories. *Information and Computation*, 103:270–298, 1993.

[28] N. Robertson and P.D. Seymour. Graph minors III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36:49–64, 1984.

[29] N. Robertson and P.D. Seymour. Graph minors V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.

[30] N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65–110, 1995.

[31] D. Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6:505–526, 1996.

[32] C. Thomassen. Embeddings and minors. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume 1, chapter 5, pages 301–349. Elsevier, 1995.

[33] P. van Emde Boas. Machine models and simulations. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1, pages 1–66. Elsevier Science Publishers, 1990.

[34] M.Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[35] M. Yannakakis. Perspectives on database theory. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 224–246, 1995.