A SAT Approach to Clique-Width

Marijn J. H. Heule[1⋆] and Stefan Szeider[2⋆⋆]

[1] Department of Computer Sciences, The University of Texas at Austin, USA
[2] Institute of Information Systems, Vienna University of Technology, Vienna, Austria

Abstract. Clique-width is a graph invariant that has been widely studied in combinatorics and computer science. However, computing the clique-width of a graph is an intricate problem, the exact clique-width is not known even for very small graphs. We present a new method for computing the clique-width of graphs based on an encoding to propositional satisfiability (SAT) which is then evaluated by a SAT solver. Our encoding is based on a reformulation of clique-width in terms of partitions that utilizes an efficient encoding of cardinality constraints. Our SAT-based method is the first to discover the exact clique-width of various small graphs, including famous graphs from the literature as well as random graphs of various density. With our method we determined the smallest graphs that require a small pre-described clique-width.

1 Introduction

Clique-width is a fundamental graph invariant that has been widely studied in combinatorics and computer science. Clique-width measures in a certain sense the "complexity" of a graph. It is defined via a graph construction process involving four operations where only a limited number of vertex labels are available; vertices that share the same label at a certain point of the construction process must be treated uniformly in subsequent steps. This graph composition mechanism was first considered by Courcelle, Engelfriet, and Rozenberg [10, 11] and has since then been an important topic in combinatorics and computer science.

Graphs of small clique-width have advantageous algorithmic properties. Algorithmic meta-theorems show that large classes of NP-hard optimization problems and #P-hard counting problems can be solved in *linear time* on classes of graphs of bounded clique-width [7, 8]. Similar results hold for the graph invariant *treewidth*, however, clique-width is more general in the sense that graphs of small treewidth also have small clique-width, but there are graphs of small clique-width but arbitrarily high treewidth [9, 6]. Unlike treewidth, dense graphs (e.g., cliques) can also have small clique-width.

All these algorithms for graphs of small clique-width require that a certificate for the graph having small clique-width is provided. However, it seems that computing the certificate, or just deciding whether the clique-width of a graph

⋆ Research supported in part by the National Science Foundation under grant CNS-0910913 and DARPA contract number N66001-10-2-4087.
⋆⋆ Research supported by the ERC, grant reference 239962 (COMPLEX REASON).

is bounded by a given number, is a very intricate combinatorial problem. More precisely, given a graph G and an integer k, deciding whether the clique-width of G is at most k is NP-complete [16]. Even worse, the clique-width of a graph with n vertices of degree greater than 2 cannot be approximated by a polynomial-time algorithm with an absolute error guarantee of n^ϵ unless P = NP, where $0 \leq \epsilon < 1$ [16]. In fact, it is even unknown whether graphs of clique-width at most 4 can be recognized in polynomial time [5]. There are approximation algorithms with an exponential error that, for fixed k, compute $f(k)$-expressions for graphs of clique-width at most k in polynomial time (where $f(k) = (2^{3k+2}-1)$ by [30], and $f(k) = 8^k - 1$ by [29]).

Because of this intricacy of this graph invariant, the exact clique-width is not known even for very small graphs.

Clique-width via SAT. We present a new method for determining the clique-width based on a sophisticated SAT encoding which entails the following ideas:

1. *Reformulation.* The conventional construction method for determining the clique-width of a graph consists of many steps. In the worst case, the number of steps is quadratic in the number of vertices. Translating this construction method into SAT would result in large instances, even for small graphs. We reformulated the problem in such a way that the number of steps is less than the number of vertices. The alternative construction method allows us to compute the clique-width of much larger graphs.
2. *Representative encoding.* Applying the frequently-used direct encoding [35] on the reformulation results in instances that have no arc consistency [18], i.e., unit propagation may find conflicts much later than required. We developed the representative encoding that is compact and realizes arc consistency.

Experimental Results. The implementation of our method allows us for the first time to determine the exact clique-width of various graphs, including famous graphs known from the literature, as well as random graphs of various density.

1. *Clique-width of small Random Graphs.* We determined experimentally how the clique-width of random graphs depends on the density. The clique-width is small for dense and sparse graphs and reaches its maximum for edge-probability 0.5. The larger n, the steeper the increase towards 0.5. These results complement the asymptotic results of Lee et al. [27].
2. *Smallest Graphs of Certain Clique-width.* In general it is not known how many vertices are required to form a graph of a certain clique-width. We provide these numbers for clique-width $k \in \{1, \ldots, 7\}$. In fact, we could compute the total number of connected graphs (modulo isomorphism) with a certain clique-width with up to 10 vertices. For instance, there are only 7 connected graphs with 8 vertices and clique-width 5 (modulo isomorphism), and no graphs with 9 vertices and clique-width 6. There are 68 graphs with 10 vertices and clique-width 6. The smallest one has 18 edges.

3. *Clique-width of Famous Named Graphs.* Over the last 50 years, researchers in graph theory have considered a large number of special graphs. These special graphs have been used as counterexamples for conjectures or for showing the tightness of combinatorial results. We considered several prominent graphs from the literature and computed their exact clique-width. These results may be of interest for people working in combinatorics and graph theory.

Related Work. We are not aware of any implemented algorithms that compute the clique-width exactly or heuristically. However, algorithms have been implemented that compute upper bounds on other width-based graph invariants, including *treewidth* [14, 19, 26], *branchwidth* [33], *Boolean-width* [24], and *rank-width* [2]. Samer and Veith [31] proposed a SAT encoding for the exact computation of treewidth. Boolean-width and rank-width can be used to approximate clique-width, however, the error can be exponential in the clique-width; in contrast, treewidth and branchwidth can be arbitrarily far from the clique-width, hence the approximation error is unbounded [4].

Our SAT encoding is based on a new characterization of clique-width that is based on partitions instead of labels. A similar partition-based characterization of clique-width, has been proposed by Heggernes et al. [23]. There are two main differences to our reformulation. Firstly, our characterization of clique-width uses three individual properties that can be easily expressed by clauses. Secondly, our characterization admits the "parallel" processing of several parts of the graph that are later joined together.

Full Version. Because of space constraints some proofs have been omitted or shortened. Detailed proofs can be found in the full version, available at arxiv.org/abs/1304.5498.

2 Preliminaries

2.1 Formulas and Satisfiability

We consider propositional formulas in Conjunctive Normal Form (*CNF formulas*, for short), which are conjunctions of clauses, where a clause is a disjunction of literals, and a literal is a propositional variable or a negated propositional variables. A CNF formula is *satisfiable* if its variables can be assigned true or false, such that each clause contains either a variable set to true or a negated variable set to false. The satisfiability problem (SAT) asks whether a given formula is satisfiable.

2.2 Graphs and Clique-width

All graphs considered are finite, undirected, and without self-loops. We denote a graph G by an ordered pair $(V(G), E(G))$ of its set of vertices and its set of edges, respectively. An edge between vertices u and v is denoted uv or equivalently vu. For basic terminology on graphs we refer to a standard text book [13].

Let k be a positive integer. A *k-graph* is a graph whose vertices are labeled by integers from $\{1, \ldots, k\}$. We consider an arbitrary graph as a k-graph with all vertices labeled by 1. We call the k-graph consisting of exactly one vertex v (say, labeled by i) an *initial* k-graph and denote it by $i(v)$. The *clique-width* of a graph G is the smallest integer k such that G can be constructed from initial k-graphs by means of repeated application of the following three operations.

1. Disjoint union (denoted by $\oplus$);
2. Relabeling: changing all labels i to j (denoted by $\rho_{i\to j}$);
3. Edge insertion: connecting all vertices labeled by i with all vertices labeled by $j, i \neq j$ (denoted by $\eta_{i,j}$ or $\eta_{j,i}$); already existing edges are not doubled.

A construction of a k-graph using the above operations can be represented by an algebraic term composed of $\oplus$, $\rho_{i\to j}$, and $\eta_{i,j}$, $(i, j \in \{1, \ldots, k\}$, and $i \neq j)$. Such a term is called a *k-expression* defining G. Thus, the clique-width of a graph G is the smallest integer k such that G can be defined by a k-expression.

Example 1. The graph $P_4 = (\{a, b, c, d\}, \{ab, bc, cd\})$ is defined by the 3-expression $\eta_{2,3}(\rho_{2\to 1}(\eta_{2,3}(\eta_{1,2}(1(a) \oplus 2(b)) \oplus 3(c))) \oplus 2(d))$. Hence $\mathrm{cwd}(P_4) \leq 3$. $\dashv$

2.3 Partitions

As partitions play an important role in our reformulation of clique-width, we recall some basic terminology. A *partition* of a set S is a set P of nonempty subsets of S such that any two sets in P are disjoint and S is the union of all sets in P. The elements of P are called *equivalence classes*. Let P, P' be partitions of S. Then P' is a *refinement* of P if for any two elements $x, y \in S$ that are in the same equivalence class of P' are also in the same equivalence class of P (this entails the case $P = P'$).

3 A Reformulation of Clique-width without Labels

Initially, we developed a SAT encoding of clique-width based on k-expressions. Even after several optimization steps, this encoding was only able to determine the clique-width of graphs consisting of at most 8 vertices. We therefore developed a new encoding based on a reformulation of clique-width which does not use k-expressions. In this section we explain this reformulation, in the next section we will discuss how it can be encoded into SAT efficiently.

Consider a finite set V, the *universe*. A *template* T consists of two partitions $\mathrm{cmp}(T)$ and $\mathrm{grp}(T)$ of V. We call the equivalence classes in $\mathrm{cmp}(T)$ the *components* of T and the equivalence classes in $\mathrm{grp}(T)$ the *groups* of T. For some intuition about these concepts, imagine that components represent induced subgraphs and that groups represent sets of vertices in some component with the same label in a k-expression. A *derivation* of length t is a finite sequence $\mathcal{D} = (T_0, \ldots, T_t)$ satisfying the following conditions.

D1 $|\mathrm{cmp}(T_0)| = |V|$ and $|\mathrm{cmp}(T_t)| = 1$.
D2 $\mathrm{grp}(T_i)$ is a refinement of $\mathrm{cmp}(T_i)$, $0 \leq i \leq t$.
D3 $\mathrm{cmp}(T_{i-1})$ is a refinement of $\mathrm{cmp}(T_i)$, $1 \leq i \leq t$.
D4 $\mathrm{grp}(T_{i-1})$ is a refinement of $\mathrm{grp}(T_i)$, $1 \leq i \leq t$.

We would like to note that D1 and D2 together imply that $|\mathrm{grp}(T_0)| = |V|$. Thus, in the first template T_0 all equivalence classes (groups and components) are singletons, and when we progress through the derivation, some of these sets are merged, until all components are merged into a single component in the last template T_t.

The *width* of a component $C \in \mathrm{cmp}(T)$ is the number of groups $g \in \mathrm{grp}(T)$ such that $g \subseteq C$. The width of a template is the maximum width over its components, and the width of a derivation is the maximum width over its templates. A *k-derivation.* is a derivation of width at most k. A derivation $\mathcal{D} = (T_0, \ldots, T_t)$ is a derivation *of* a graph $G = (V, E)$ if V is the universe of the derivation and the following three conditions hold for all $1 \leq i \leq t$.

Edge Property: For any two vertices $u, v \in V$ such that $uv \in E$, if u, v are in the same group in T_i, then u, v are in the same component in T_{i-1}.
Neighborhood Property: For any three vertices $u, v, w \in V$ such that $uv \in E$ and $uw \notin E$, if v, w are in the same group in T_i, then u, v are in the same component in T_{i-1}.
Path Property: For any four vertices $u, v, w, x \in V$, such that $uv, uw, vx \in E$ and $wx \notin E$, if u, x are in the same group in T_i and v, w are in the same group in T_i, then u, v are in the same component in T_{i-1}.

The neighborhood property and the path property could be merged into a single property if we do not insist that all mentioned vertices are distinct. However, two separate properties provide a more compact SAT encoding.

The following example illustrates that a derivation can define more than one graph, in contrast to a k-expression, which defines exactly one graph.

Example 2. Consider the derivation $\mathcal{D} = (T_0, \ldots, T_3)$ with universe $V = \{a, b, c, d\}$ and

$$\begin{aligned}
\mathrm{cmp}(T_0) &= \{\{a\}, \{b\}, \{c\}, \{d\}\}, & \mathrm{grp}(T_0) &= \{\{a\}, \{b\}, \{c\}, \{d\}\},\\
\mathrm{cmp}(T_1) &= \{\{a, b\}, \{c\}, \{d\}\}, & \mathrm{grp}(T_1) &= \{\{a\}, \{b\}, \{c\}, \{d\}\},\\
\mathrm{cmp}(T_2) &= \{\{a, b, c\}, \{d\}\}, & \mathrm{grp}(T_2) &= \{\{a\}, \{b\}, \{c\}, \{d\}\},\\
\mathrm{cmp}(T_3) &= \{\{a, b, c, d\}\}, & \mathrm{grp}(T_3) &= \{\{a, b\}, \{c\}, \{d\}\}.
\end{aligned}$$

The width of $\mathcal{D}$ is 3. Consider the graph $G = (V, \{ab, ad, bc, bd\})$. To see that $\mathcal{D}$ is a 3-derivation of G, we need to check the edge, neighborhood, and path properties. We observe that a, b are the only two vertices such that $ab \in E(G)$ and both vertices appear in the same group of some T_i (here, we have $i = 3$). To check the edge property, we only need to verify that a, b are in the same component of T_2, which is true. For the neighborhood property, the only relevant choice of three vertices is a, b, c ($bc \in E(G)$, $ac \notin E(G)$, and a, b in a group of T_3). The neighborhood property requires that b, c are in the same component in T_2, which is the case. The path property is satisfied since there is no template in which two pairs of vertices belong to the same group, respectively.

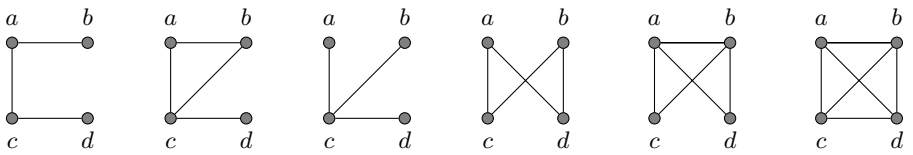

Fig. 1. All connected graphs with four vertices (up to isomorphism). The 3-derivation of Example 2 defines all six graphs. The clique-width for all, but the first graph is 2.

Similarly we can verify that $\mathcal{D}$ is a derivation of the graph $G' = (V, \{ab, bc, cd\})$. In fact, for all connected graphs with four vertices, there exists an isomorphic graph that is defined by $\mathcal{D}$ (see Figure 1). However, $\mathcal{D}$ is not a derivation of the graph $G'' = (V, \{ab, ac, bd, cd\})$ since the neighborhood property is violated: $bd \in E(G'')$ and $ad \notin E(G'')$, a, b belong to the same group in T_3, while a, d do not belong to the same component in T_2. ⊣

We call a derivation $(T_0, \ldots, T_t)$ to be *strict* if $|\mathrm{cmp}(T_{i-1})| > |\mathrm{cmp}(T_i)|$ holds for all $1 \leq i \leq t$. It is easy to see that if two consecutive templates in a derivation of a graph G have the same components, then one of the two templates can be omitted, and we still have a derivation of G. This yields the next lemma, whose detailed proof can be found in the full version of this paper.

Lemma 1. *If G has a k-derivation, it has a strict k-derivation.*

Lemma 2. *Every strict k-derivation of a graph with n vertices has length at most $n - 1$.*

Proof. Let $(T_0, \ldots, T_t)$ be a strict k-derivation of a graph with n vertices. Since $|\mathrm{cmp}(T_0)| = n$ and $|\mathrm{cmp}(T_0)| = 1$, it follows that $t \leq n - 1$. □

In the proofs of the next two lemmas we need the following concept of a *k-expression tree*, which is the parse tree of a k-expression equipped with some additional information. Let ϕ be a k-expression for a graph $G = (V, E)$. Let Q be the parse tree of ϕ with root r. Consider a node x of Q and let ϕ_x be the subexpression of ϕ whose parse tree is the subtree of Q rooted at x. Then x is labeled with the k-graph G_x constructed by the k-expression ϕ_x. Thus the leaves of Q are labeled with initial k-graphs and the root r is labeled with a labeled version of G. One $\oplus$-node of the parse tree can represent several directly subsequent $\oplus$-operations (e.g., the operation $(x \oplus y) \oplus z$ can be represented by a single node with three children). Evidently, k-expressions and their k-expression trees can be effectively transformed into each other.

We introduce some additional terminology on k-expression trees. We call a non-leaf node of Q an $\oplus$-node, η-node, or ρ-node, according to the operation it represents. We define the *rank* $R(x)$ of node x of Q as the largest number of $\oplus$-nodes that appear on a path from a leaf of Q to x. Hence leaves have rank 0. We denote the set of nodes of Q of rank i by $V_i(Q)$.

Lemma 3. *From a k-expression of a graph G we can obtain a k-derivation of G in polynomial time.*

Proof. Let ϕ be a k-expression of $G = (V, E)$ and let Q be the corresponding k-expression tree. We let $t := R(r)$ and define a derivation $\mathcal{D} = (T_0, \ldots, T_t)$ by setting $\mathrm{cmp}(T_i) = \{\, V(G_x)) : x \in V_i(Q) \,\}$ and $\mathrm{grp}(T_i) = \bigcup_{x \in V_i(Q)} \mathrm{grp}(G_x)$ where $\mathrm{grp}(G_x)$ denotes the partition of $V(G_x)$ into sets of vertices that have the same label. By construction, $\mathcal{D}$ is a derivation with universe V. Furthermore, since ϕ is a k-expression, $|\mathrm{grp}(G_x)| \leq k$ for all nodes x of Q. Hence $\mathcal{D}$ is a k-derivation. It is not difficult to verify that $\mathcal{D}$ is a k-derivation of G by checking the edge, neighborhood, and path properties. Due to space limitations, this part of the proof is only in the full version of this paper. The above procedure for generating the k-derivation can clearly be carried out in polynomial time. □

Example 3. Consider the 3-expression ϕ for the graph P_4 of Example 1. Applying the procedure described in the proof of Lemma 3 we obtain the 3-derivation $\mathcal{D}$ of Example 2. ⊣

Lemma 4. *From a k-derivation of a graph G we can obtain a k-expression of G in polynomial time.*

Proof. Due to space restrictions we only sketch the proof; a detailed proof can be found in the full version of the paper. Given a k-derivation $\mathcal{D} = (T_0, \ldots, T_t)$ of a graph $G = (V, E)$, we first construct a k-expression $\phi_{\oplus}$ that only contains $\oplus$-operations and initial graphs with label 1. These $\oplus$-operations reflect how components are merged along the derivation $\mathcal{D}$. Next we insert ρ-operations directly before $\oplus$-operations and obtain a k-expression $\phi_{\oplus,\rho}$. These $\oplus$-operations reflect how groups are merged along the derivation $\mathcal{D}$. In a last step we insert η-operations. For each edge $uv \in E$ we take the smallest subexpression of $\phi_{\oplus,\rho}$ that defines a graph containing both u and v (the outermost operation of this subexpression is clearly $\oplus$). Let a, b be the labels of u, v in this graph, respectively. It follows from the edge property that $a \neq b$. Right after this subexpression we insert the operation $\eta_{a,b}$. It follows from the neighborhood and path properties that this η-operation does not insert any edges not contained in E. □

We note that we could have saved some ρ-operations in the proof of Lemma 4. In particular the k-expression produced may contain ρ-operations where the number of different labels before and after the application of the ρ-operation remains the same. It is easy to see that such a ρ-operations can be omitted if we change labels of some initial k-graphs accordingly.

Example 4. Consider the derivation $\mathcal{D}$ of graph G in Example 2. We construct a 3-expression of G using the procedure as described in the proof of Lemma 4. First we obtain $\phi_{\oplus} = ((1(a) \oplus 1(b)) \oplus 1(c)) \oplus 1(d)$. Next we insert ρ operations to represent how the groups evolve through the derivation: $\phi_{\oplus,\rho} = \rho_{1\to 2}((1(a) \oplus \rho_{1\to 2}(1(b)) \oplus \rho_{1\to 3}1(c))) \oplus 1(d)$. Finally we add η operations, and obtain $\phi_{\oplus,\rho,\eta} = \eta_{1,2}(\rho_{1\to 2}(\eta_{2,3}(\eta_{1,2}(1(a) \oplus \rho_{1\to 2}(1(b))) \oplus \rho_{1\to 3}1(c))) \oplus 1(d))$. ⊣

By Lemma 2 we do not need to search for k-derivations of length $> n - 1$ when the graph under consideration has n vertices. The next lemma improves this bound to $n - k + 1$ which provides a significant improvement for our SAT encoding, especially if the graph under consideration has large clique-width.

Lemma 5. *Let $1 \leq k \leq n$. If a graph with n vertices has a k-derivation, then it has a k-derivation of length $n - k + 1$.*

Proof. Due to space restrictions we sketch the proof only, a detailed proof can be found in the full version of this paper. Let $\mathcal{D} = (T_0, \ldots, T_t)$ be a k-derivation of a graph $G = (V, E)$ with $|V| = n$. By Lemma 1 we may assume that $\mathcal{D}$ is strict. Let i be the largest index $1 \leq i \leq t$ where all components of T_i have size at most k and put $\ell = t - i$. It is not difficult to see that $\ell \leq n - k$. We define a new template T'_i with $\mathrm{cmp}(T'_i) = \mathrm{cmp}(T_i)$ and $\mathrm{grp}(T'_i) = \mathrm{grp}(T_0)$, and we set $\mathcal{D}' = (T_0, T'_i, T_{i+1}, \ldots, T_t)$. It can be verified that $\mathcal{D}'$ is a k-derivation of G. □

Example 5. Again, consider the derivation $\mathcal{D}$ of Example 2. $\mathcal{D}$ defines P_4 which has clique-width 3 [9]. According to Lemma 5, it should have a derivation of length $n - k + 1 = 4 - 3 + 1 = 2$. We can obtain such a derivation by removing T_1 from $\mathcal{D}$, which gives $\mathcal{D}' = (T_0, T_2, T_3)$. ⊣

By combining Lemmas 3, 4, and 5, we arrive at the main result of this section.

Proposition 1. *Let $1 \leq k \leq n$. A graph G with n nodes has clique-width at most k if and only if G has a k-derivation of length at most $n - k + 1$.*

4 Encoding a Derivation of a Graph

Let $G = (V, E)$ be graph, and $t > 0$ an integer. We are going to construct a CNF formula $F_{\mathrm{der}}(G, t)$ that is satisfiable if and only if G has a derivation of length t. We assume that the vertices of G are given in some arbitrary but fixed linear order $<$.

For any two distinct vertices u and v of G and any $0 \leq i \leq t$ we introduce a *component variable* $c_{u,v,i}$. Similarly, for any two distinct vertices u and v of G with $u < v$ and any $0 \leq i \leq t$ we introduce a *group variable* $g_{u,v,i}$. Intuitively, $c_{u,v,i}$ or $g_{u,v,i}$ are true if and only if u and v are in the same component or group, respectively, in the ith template of an implicitly represented derivation of G.

The formula $F_{\mathrm{der}}(G, t)$ is the conjunction of all the clauses described below. The following clauses represent the conditions D1–D4.

$$(\bar{c}_{u,v,0}) \wedge (c_{u,v,t}) \wedge (c_{u,v,i} \vee \bar{g}_{u,v,i}) \wedge (\bar{c}_{u,v,i-1} \vee c_{u,v,i}) \wedge (\bar{g}_{u,v,i-1} \vee g_{u,v,i}) \quad \text{for } u, v \in V,\ u < v,\ 0 \leq i \leq t.$$

We further add clauses that ensure that the relations of being in the same group and of being in the same component are transitive.

$$(\bar{c}_{u,v,i} \vee \bar{c}_{v,w,i} \vee c_{u,w,i}) \wedge (\bar{c}_{u,v,i} \vee \bar{c}_{u,w,i} \vee c_{v,w,i}) \wedge (\bar{c}_{u,w,i} \vee \bar{c}_{v,w,i} \vee c_{u,v,i}) \ \wedge$$
$$(\bar{g}_{u,v,i} \vee \bar{g}_{v,w,i} \vee g_{u,w,i}) \wedge (\bar{g}_{u,v,i} \vee \bar{g}_{u,w,i} \vee g_{v,w,i}) \wedge (\bar{g}_{u,w,i} \vee \bar{g}_{v,w,i} \vee g_{u,v,i}) \quad \text{for } u, v, w \in V,\ u < v < w,\ 0 \leq i \leq t.$$

In order to enforce the *edge property* we add the following clauses for any two vertices $u, v \in V$ with $u < v$, $uv \in E$ and $1 \leq i \leq t$:

$$(c_{u,v,i-1} \vee \bar{g}_{u,v,i}).$$

Further, to enforce the *neighborhood property*, we add for any three vertices $u, v, w \in V$ with $uv \in E$ and $uw \notin E$ and $1 \leq i \leq t$, the following clauses.

$(c_{\min(u,v),\max(u,v),i-1} \vee \bar{g}_{\min(v,w),\max(v,w),i})$

Finally, to enforce the *path property* we add for any four vertices u, v, w, x, such that $uv, uw, vx \in E$, and $wx \notin E$, $u < v$ and $1 \leq i \leq t$ the following clauses:

$(c_{u,v,i-1} \vee \bar{g}_{\min(u,x),\max(u,x),i} \vee \bar{g}_{\min(v,w),\max(v,w),i})$

The following statement is a direct consequence of the above definitions.

Lemma 6. *$F_{\mathrm{der}}(G,t)$ is satisfiable if and only if G has a derivation of length t.*

5 Encoding a k-Derivation of a Graph

In this section, we describe how the formula $F_{\mathrm{der}}(G,t)$ can be extended to encode a derivation of width at most k. Ideally, one wants to encode that unit propagation results in a conflict on any assignment of component and group variables representing a derivation containing a component with more than k groups. First we will describe the conventional direct encoding [35] followed by our representative encoding. Only the latter encoding realizes arc consistency [18].

5.1 Direct Encoding

We introduce new Boolean variables $l_{v,a,i}$ for $v \in V$, $1 \leq a \leq k$, and $0 \leq i \leq t$. The purpose is to assign each vertex for each template a group number between 1 and k. The intended meaning of a variable $l_{v,a,i}$ is that in T_i, vertex v has group number a. Let $F(G,k,t)$ denote the formula obtained from $F_{\mathrm{der}}(G,t)$ by adding the following three sets of clauses. The first ensures that every vertex has at least one group number, the second ensures that every vertex has at most one group number, and the third ensures that two vertices of the same group share the same group number.

$$(l_{v,1,i} \vee l_{v,2,i} \vee \cdots \vee l_{v,k,i}) \qquad \text{for } v \in V,\ 0 \leq i \leq t,$$
$$(\bar{l}_{v,a,i} \vee \bar{l}_{v,b,i}) \qquad \text{for } v \in V,\ 1 \leq a < b \leq k,\ 0 \leq i \leq t,$$
$$(\bar{l}_{u,a,i} \vee \bar{l}_{v,a,i} \vee \bar{c}_{u,v,i} \vee g_{u,v,i}) \wedge (\bar{l}_{u,a,i} \vee l_{v,a,i} \vee \bar{g}_{v,w,i}) \wedge (l_{v,a,i} \vee \bar{l}_{v,a,i} \vee \bar{g}_{u,v,i})$$
$$\text{for } u, v \in V,\ u < v,\ 1 \leq a \leq k,\ 0 \leq i \leq t.$$

Together with Lemma 6 this construction directly yields the following statement.

Proposition 2. *Let $G = (V,E)$ be graph and $t = |V| - k + 1$. Then $F(G,k,t)$ is satisfiable if and only if* $\mathrm{cwd}(G) \leq k$.

Example 6. Let $G = (V,E)$ and $k = 2$. Vertices $u, v, w \in V$ in template T_i, are in one component, but in different groups. Hence the corresponding component variables are true, and the corresponding group variables are false. The clauses containing the variables $l_{u,a,i}, l_{v,a,i}, l_{w,a,i}$ with $a \in \{1,2\}$ after removing falsified literals are: $(l_{u,1,i} \vee l_{u,2,i})$, $(l_{v,1,i} \vee l_{v,2,i})$, $(l_{w,1,i} \vee l_{w,2,i})$, $(\bar{l}_{u,1,i} \vee \bar{l}_{v,1,i})$, $(\bar{l}_{u,1,i} \vee \bar{l}_{w,1,i})$, $(\bar{l}_{v,1,i} \vee \bar{l}_{w,1,i})$, $(\bar{l}_{u,2,i} \vee \bar{l}_{v,2,i})$, $(\bar{l}_{u,2,i} \vee \bar{l}_{w,2,i})$, $(\bar{l}_{v,2,i} \vee \bar{l}_{w,2,i})$. These clauses cannot be satisfied, yet unit propagation will not result in a conflict. Therefore, a SAT solver may not be able to cut off the current branch. ⊣

5.2 The Representative Encoding

To overcome the unit propagation problem of the direct encoding, as described in Example 6, we propose the *representative encoding* which uses two types of variables. First, for each $v \in V$ and $1 \leq i \leq t$ we introduce a representative variable $r_{v,i}$. This variable, if assigned to true, expresses that vertex v is the representative of a group in template T_i. In each group, only one vertex can be the representative and we choose to make the first vertex in the lexicographical ordering the representative. This results in the following clauses:

$$(r_{v,i} \vee \bigvee_{u \in V, u<v} g_{u,v,i}) \wedge \bigwedge_{u \in V, u<v} (\bar{r}_{v,i} \vee \bar{g}_{u,v,i}) \qquad \text{for } v \in V,\ 0 \leq i \leq t$$

Additionally we introduce auxiliary variables to efficiently encode that the number of representative vertices in a component is at most k. These auxiliary variables are based on the *order encoding* [34]. Consider a (non-Boolean) variable $L_{v,i}$ with domain $D = \{1, \ldots, k\}$, whose elements denote the group number of vertex v in template T_i. In the direct encoding, we used k variables $l_{v,a,i}$ with $a \in D$. Assigning $l_{v,a,i} = \mathtt{1}$ in that encoding means $L_{v,i} = a$. Alternatively, we can use *order variables* $o^{>}_{v,a,i}$ with $v \in V$, $a \in D \setminus \{k\}$, $0 \leq i \leq t$. Assigning $o^{>}_{v,a,i} = \mathtt{1}$ means $L_{v,i} > a$. Consequently, $o^{>}_{v,a,i} = \mathtt{0}$ means $L_{v,i} \leq a$.

Example 7. Given an assignment to the order variables $o^{>}_{v,a,i}$, one can easily construct the equivalent assignment to the variables in the direct encoding (and the other way around). Below is a visualization of the equivalence relation with $k = 5$. In the middle is a binary representation of each of the k labels by concatenating the Boolean values to the order variables.

$$\begin{array}{l}
L_v = 1 \leftrightarrow l_{v,1,i} = \mathtt{1} \leftrightarrow \mathtt{0000} \leftrightarrow o^{>}_{v,1,i} = o^{>}_{v,2,i} = o^{>}_{v,3,i} = o^{>}_{v,4,i} = \mathtt{0} \\
L_v = 2 \leftrightarrow l_{v,2,i} = \mathtt{1} \leftrightarrow \mathtt{1000} \leftrightarrow o^{>}_{v,1,i} = \mathtt{1}, o^{>}_{v,2,i} = o^{>}_{v,3,i} = o^{>}_{v,4,i} = \mathtt{0} \\
L_v = 3 \leftrightarrow l_{v,3,i} = \mathtt{1} \leftrightarrow \mathtt{1100} \leftrightarrow o^{>}_{v,1,i} = o^{>}_{v,2,i} = \mathtt{1}, o^{>}_{v,3,i} = o^{>}_{v,4,i} = \mathtt{0} \\
L_v = 4 \leftrightarrow l_{v,4,i} = \mathtt{1} \leftrightarrow \mathtt{1110} \leftrightarrow o^{>}_{v,1,i} = o^{>}_{v,2,i} = o^{>}_{v,3,i} = \mathtt{1}, o^{>}_{v,4,i} = \mathtt{0} \\
L_v = 5 \leftrightarrow l_{v,5,i} = \mathtt{1} \leftrightarrow \mathtt{1111} \leftrightarrow o^{>}_{v,1,i} = o^{>}_{v,2,i} = o^{>}_{v,3,i} = o^{>}_{v,4,i} = \mathtt{1}
\end{array} \qquad \dashv$$

Although our encoding is based on the variables from the order encoding, we use none of the associated clauses. We implemented the original order [34], which resulted in many long clauses and the performance was comparable to the direct encoding.

Instead, we combined the representative and order variables. Our use of the order variables can be seen as the encoding of a sequential counter [32]. We would like to point out that if u and v are both representative vertices in the same component of template T_i and $u < v$, then $o^{>}_{u,a,i} = \mathtt{0}$ and $o^{>}_{v,a,i} = \mathtt{1}$ must hold for some $1 \leq a < k$. Consequently, $o^{>}_{u,k-1,i} = \mathtt{0}$ (vertex u has not the highest group number in T_i), $o^{>}_{v,1,i} = \mathtt{1}$ (vertex v has not the lowest group number in T_i), and $o^{>}_{u,a,i} \rightarrow o^{>}_{v,a+1,i}$: These constraints can be expressed by the following clauses.

$$\begin{array}{l}
(\bar{c}_{u,v,i} \vee \bar{r}_{u,i} \vee \bar{r}_{v,i} \vee \bar{o}^{>}_{u,k-1,i}) \wedge (\bar{c}_{u,v,i} \vee \bar{r}_{u,i} \vee \bar{r}_{v,i} \vee o^{>}_{v,1,i}) \wedge \\
\bigwedge_{1 \leq a < k-1} (\bar{c}_{u,v,i} \vee \bar{r}_{u,i} \vee \bar{r}_{v,i} \vee \bar{o}^{>}_{u,a,i} \vee o^{>}_{v,a+1,i}) \quad \text{for } u, v \in V,\ u < v,\ 0 \leq i \leq t.
\end{array}$$

Example 8. Consider a graph $G = (V, E)$ with $u, v, w, x \in V$ and the representative encoding with $k = 3$. We will show that if u,v,w, and x are all in the same component and they are all representatives of their respective group numbers in template T_i, then unit propagation will result in a conflict (because there are four representatives and only three group numbers). Observe that all corresponding component and representative variables are true. This example, with falsified literals removed, contains the clauses $(\bar{o}^{>}_{u,2,i})$, $(\bar{o}^{>}_{u,1,i} \vee \boldsymbol{o^{>}_{v,2,i}})$, $(o^{>}_{v,1,i})$, $(\bar{o}^{>}_{u,2,i})$, $(\bar{o}^{>}_{u,1,i} \vee \boldsymbol{o^{>}_{w,2,i}})$, $(o^{>}_{w,1,i})$, $(\bar{o}^{>}_{u,2,i})$, $(\bar{o}^{>}_{u,1,i} \vee o^{>}_{x,2,i})$, $(o^{>}_{x,1,i})$, $(\bar{o}^{>}_{v,2,i})$, $(\boldsymbol{\bar{o}^{>}_{v,1,i}} \vee \boldsymbol{o^{>}_{w,2,i}})$, $(o^{>}_{w,1,i})$, $(\bar{o}^{>}_{v,2,i})$, $(\boldsymbol{\bar{o}^{>}_{v,1,i}} \vee o^{>}_{x,2,i})$, $(o^{>}_{x,1,i})$, $(\bar{o}^{>}_{w,2,i})$, $(\boldsymbol{\bar{o}^{>}_{w,1,i}} \vee o^{>}_{x,2,i})$, $(o^{>}_{x,1,i})$. Literals that are falsified by unit clauses are shown in bold. Notice that $(\bar{o}^{>}_{v,1,i} \vee o^{>}_{w,2,i})$ is falsified, i.e., a conflicting clause. ⊣

Both the direct and representative encoding require $n(n + k - 1)(n - k + 2)$ variables. The number of clauses depends on the set of edges. In worst case, the number of clauses can be $\mathcal{O}(n^5 - n^4k)$ due to the path condition.

6 Experimental Results

In this section we report the results we obtained by running our SAT encoding on various classes of graphs. Given a graph $G = (V, E)$, we compute that G has clique-width k by determining for which value of k it holds that $F(G, k, |V| - k+1)$ is satisfiable and $F(G, k-1, |V|-k+2)$ is unsatisfiable. We used the SAT solver `Glucose` version 2.2 [1] to solve the encoded problems. `Glucose` solved the hardest instances about twice as fast (or more) as other state-of-the-art solvers such as `Lingeling` [3], `Minisat` [15] and `Clasp` [17]. We used a 4-core Intel Xeon CPU E31280 3.50GHz, 32 Gb RAM machine running Ubuntu 10.04.

Although the direct and representative encodings result in CNF formulas of almost equal size, there is a huge difference in costs to solve these instances. To determine the clique-width of the famous named graphs (see below) using the direct encoding takes about two to three orders of magnitude longer as compared to the representative encoding. For example, we can establish that the Paley graph with 13 vertices has clique-width 9 within a few seconds using the representative encoding, while the solver requires over an hour using the direct encoding. Because of the huge difference in speed, we discarded the use of the direct encoding in the remainder of this section.

We noticed that upper bounds (satisfiable formulas) are obtained much faster than lower bounds (unsatisfiable formulas). The reason is twofold. First, the whole search space needs to be explored for lower bounds, while for upper bounds, one can be "lucky" and find a solution fast. Second, due to our encoding, upper bound formulas are smaller (due to a smaller t) which makes them easier. Table 1 shows this for a random graph with 20 vertices for the direct encoding and the representative encoding.

We examined whether adding symmetry-breaking predicates could improve performance. We used `Saucy` version 3 for this purpose [25]. After the addition of the clauses with representative variables, the number of symmetries is drastically

Table 1. Runtimes in seconds of the direct and representative encoding on a random graph with 20 vertices and 95 edges for different values of k. Up to $k = 9$ the formulas are unsatisfiable, afterwards they are satisfiable. Timeout (TO) is 20,000 seconds.

k	3	4	5	6	7	8	9	10	11	12	13	14	15	16
direct	1.39	14.25	101.1	638.5	18,337	TO	TO	TO	TO	30.57	0.67	0.50	0.10	0.10
repres	0.62	2.12	8.14	12.14	33.94	102.3	358.6	9.21	0.40	0.35	0.32	0.29	0.29	0.28

reduced. However, one can generate symmetry-breaking predicates for $F_{\mathrm{der}}(G, t)$ and add those instead. Although it is helpful in some cases, the average speed-up was between 5 to 10%.

Our experimental computations are ongoing. Below we report on some of the results we have obtained so far.

6.1 Random Graphs

The asymptotics of the clique-width of random graphs have been studied by Lee et al. [27]. Their results show that for random graphs on n vertices the following holds asymptotically almost surely: If the graphs are very sparse, with an edge probability below $1/n$, then clique-width is at most 5; if the edge probability is larger than $1/n$, then the clique-width grows at least linearly in n. Our first group of experiments complements these asymptotic results and provides a detailed picture on the clique-width of small random graphs. We have used the SAT encoding to compute the clique-width of graphs with 10, 15, and 20 vertices, with the edge probability ranging from 0 to 1. A plot of the distribution is displayed in Figure 2. It is interesting to observe the symmetry at edge probability 1/2, and the how the steepness of the curve increases with the number of vertices. Note the "shoulders" of the curve for very sparse and very dense graphs.

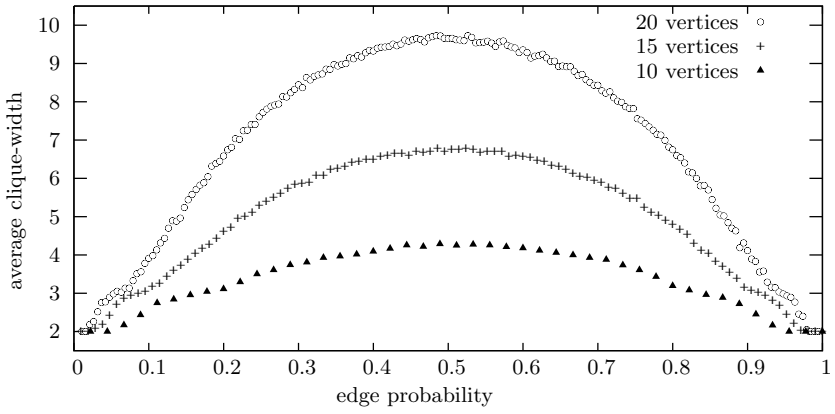

Fig. 2. Average clique-width of random graphs with edge probabilities between 0 and 1. Each dot in the graph represents the average clique-width of 100 graphs.

6.2 The Clique-Width Numbers

For every $k > 0$, let n_k denote the smallest number such that there exists a graph with n_k vertices that has clique-width k. We call n_k the kth *clique-width number*. From the characterizations known for graphs of clique-width 1, 2, and 3, respectively [5], it is easy to determine the first three clique-width numbers (1, 2, and 4). However, determining n_4 is not straightforward, as it requires nontrivial arguments to establish clique-width lower bounds. We would like to point out that a similar sequence for the graph invariant *treewidth* is easy to determine, as the complete graph on n vertices is the smallest graph of treewidth $n-1$. One of the very few known graph classes of unbounded clique-width for which the exact clique-width can be determined in polynomial time are grids [23]; the $k \times k$ grid with $k \geq 3$ has clique-width $k+1$ [20]. Hence grids provide the upper bounds $n_4 \leq 9$, $n_5 \leq 16$, $n_6 \leq 25$, and $n_7 \leq 36$. With our experiments we could determine $n_4 = 6$, $n_5 = 8$, $n_6 = 10$, $n_7 = 11$, $n_8 \leq 12$, and $n_9 \leq 13$. It is known that the path on four vertices (P_4) is the unique smallest graph in terms of the number of vertices with clique-width 3. We could determine that the triangular prism (3-Prism) is the unique smallest graph with clique-width 4, and that there are exactly 7 smallest graphs with clique-width 5. There are 68 smallest graphs with clique-width 6 and one of them has only 18 edges. See Figure 3 for an illustration. Additionally, we found several graphs of size 11 with clique-width 7 by extending a graph of size 10 with clique-width 6.

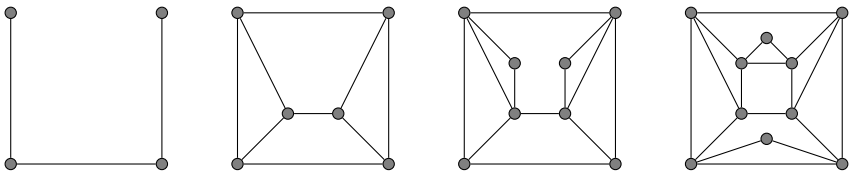

Fig. 3. Smallest graphs with clique-width 3, 4, 5, and 6 (from left to right).

Proposition 3. *The clique-width sequence starts with the numbers* 1*,* 2*,* 4*,* 6*,* 8*,* 10*,* 11*.*

We used Brendan McKay's software package Nauty [28] to avoid checking isomorphic copies of the same graph. There are several other preprocessing methods that can speed up the search for small graphs of prescribed clique-width $k \geq 2$. Obviously, we can limit the search to *connected* graphs, as the clique-width of a graph is clearly the maximum clique-width of its connected components. We can also ignore graphs that contain *twins*—two vertices that have exactly the same neighbors—as we can delete one of them without changing the clique-width. Similarly, we can ignore graphs with a *universal vertex*, a vertex that is adjacent to all other vertices, as it can be deleted without changing the clique-width. All these filtering steps are subsumed by the general concept of *prime graphs*. Consider a graph $G = (V, E)$. A vertex $u \in V$ *distinguishes* vertices $v, w \in V$ if $uv \in E$ and $uw \notin E$. A set $M \subseteq V$ is a *module* if no vertex from $V \setminus M$

distinguishes two vertices from M. A module M is *trivial* if $|M| \in \{0, 1, |V|\}$. A graph is *prime* if it contains only trivial modules. It is well-known that the clique-width of a graph is either 2 or the maximum clique-width of its induced prime subgraphs [9]. Hence, in our search, we can ignore all graphs that are not prime. We can efficiently check whether a graph is prime [21]. The larger the number of vertices, the larger the fraction of non-prime graphs (considering connected graphs modulo isomorphism). Table 2 gives detailed results.

Table 2. Number of connected and prime graphs with specified clique-width, modulo isomorphism.

			clique-width				
$\|V\|$	connected	prime	2	3	4	5	6
4	6	1	0	1	0	0	0
5	21	4	0	4	0	0	0
6	112	26	0	25	1	0	0
7	853	260	0	210	50	0	0
8	11,117	4,670	0	1,873	2,790	7	0
9	261,080	145,870	0	16,348	125,364	4,158	0
10	11,716,571	8,110,354	0	142,745	5,520,350	2,447,190	68

6.3 Famous Named Graphs

The graph theoretic literature contains several graphs that have names, sometimes inspired by the graph's topology, and sometimes after their discoverer. We have computed the clique-width of several named graphs, the results are given in Table 3 (definitions of all considered graphs can be found in MathWorld [37]). The *Paley graphs*, named after the English mathematician Raymond Paley (1907–1933), stick out as having large clique-width. Our results on the clique-width of Paley graphs imply some upper bounds on the 9th and 11th clique-width numbers: $n_9 \leq 13$ and $n_{11} \leq 17$.

7 Conclusion

We have presented a SAT approach to the exact computation of clique-width, based on a reformulation of clique-width and several techniques to speed up the search. This new approach allowed us to systematically compute the exact clique-width of various small graphs. We think that our results could be of relevance for theoretical investigations. For instance, knowing small vertex-minimal graphs of certain clique-width could be helpful for the design of discrete algorithms that recognize graphs of bounded clique-width. Such graphs can also be useful as gadgets for a reduction to show that the recognition of graphs of clique-width 4 is NP-hard, which is still a long-standing open problem [16]. Furthermore, as

Table 3. Clique-width of named graphs. Sizes are reported for the unsatisfiables.

graph	$\|V\|$	$\|E\|$	cwd	variables	clauses	UNSAT	SAT
Brinkmann	21	42	10	8,526	163,065	3,932.56	1.79
Chvátal	12	24	5	1,800	21,510	0.40	0.09
Clebsch	16	40	8	3,872	60,520	191.02	0.09
Desargues	20	30	8	7,800	141,410	3,163.70	0.26
Dodecahedron	20	30	8	7,800	141,410	5,310.07	0.33
Errera	17	45	8	4,692	79,311	82.17	0.16
Flower snark	20	30	7	8,000	148,620	276.24	3.9
Folkman	20	40	5	8,280	168,190	11.67	0.36
Franklin	12	18	4	1,848	21,798	0.07	0.04
Frucht	12	18	5	1,800	20,223	0.39	0.02
Hoffman	16	32	6	4,160	64,968	8.95	0.46
Kittell	23	63	8	12,006	281,310	179.62	18.65
McGee	24	36	8	13,680	303,660	8,700.94	59.89
Sousselier	16	27	6	4,160	63,564	3.67	11.75
Paley-13	13	39	9	1,820	22,776	12.73	0.05
Paley-17	17	68	11	3,978	72,896	194.38	0.12
Pappus	18	27	8	5,616	90,315	983.67	0.14
Petersen	10	15	5	1,040	9,550	0.10	0.02
Poussin	15	39	7	3,300	50,145	9.00	0.21
Robertson	19	38	9	6,422	112, 461	478.83	0.76
Shrikhande	16	48	9	3,680	59,688	129.75	0.11

discussed in Section 1, there are no heuristic algorithms to compute the clique-width directly, but heuristic algorithms for related parameters can be used to obtain upper bounds on the clique-width. Our SAT-based approach can be used to empirically evaluate how far heuristics are from the optimum, at least for small and medium-sized graphs.

So far we have focused in our experiments on the exact clique-width, but for various applications it is sufficient to have good upper bounds. Our results (see Table 1) suggest that our approach can be scaled to medium-sized graphs for the computation of upper bounds. We also observed that for many graphs the upper bound of Lemma 5 is not tight. Thus, we expect that if we search for shorter derivations, which is significantly faster, this will yield optimal or close to optimal solutions in many cases.

Finally, we would like to mention that our SAT-based approach is very flexible and open. It can easily be adapted to variants of clique-width, such as linear clique-width [22, 16], m-clique-width [12], or NLC-width [36]. Hence, our approach can be used for an empirical comparison of these parameters.

Acknowledgement

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing grid resources that have contributed to the research results reported within this paper.

References

1. Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 399–404, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.
2. Martin Beyß. Fast algorithm for rank-width. In *Mathematical and Engineering Methods in Computer Science, 8th International Doctoral Workshop, MEMICS 2012, Znojmo, Czech Republic, October 25-28, 2012, Revised Selected Papers*, volume 7721 of *Lecture Notes in Computer Science*, pages 82–93. Springer Verlag, 2013.
3. Armin Biere. Lingeling and friends entering the SAT Challenge 2012. In A. Balint, A. Belov, A. Diepold, S. Gerber, M. Järvisalo, and C. Sinz, editors, *Solver and Benchmark Descriptions*, volume B-2012-2 of *Department of Computer Science Series of Publications B.*, pages 33–34. University of Helsinki, 2012.
4. Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Boolean-width of graphs. *Theoretical Computer Science*, 412(39):5187–5204, 2011.
5. Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce Reed, and Udi Rotics. Polynomial-time recognition of clique-width $\leq$ 3 graphs. *Discr. Appl. Math.*, 160(6):834–865, 2012.
6. Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005.
7. B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.
8. B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discr. Appl. Math.*, 108(1-2):23–52, 2001.
9. B. Courcelle and S. Olariu. Upper bounds to the clique-width of graphs. *Discr. Appl. Math.*, 101(1-3):77–114, 2000.
10. Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Context-free handle-rewriting hypergraph grammars. In Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors, *Graph-Grammars and their Application to Computer Science, 4th International Workshop, Bremen, Germany, March 5–9, 1990, Proceedings*, volume 532 of *Lecture Notes in Computer Science*, pages 253–268, 1991.
11. Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *J. of Computer and System Sciences*, 46(2):218–270, 1993.
12. Bruno Courcelle and Andrew Twigg. Constrained-path labellings on graphs of bounded clique-width. *Theory Comput. Syst.*, 47(2):531–567, 2010.
13. Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 2nd edition, 2000.
14. P. Alex Dow and Richard E. Korf. Best-first search for treewidth. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1146–1151. AAAI Press, 2007.
15. Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer Berlin Heidelberg, 2004.
16. Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width is NP-complete. *SIAM J. Discrete Math.*, 23(2):909–939, 2009.

17. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, volume 4483 of *Lecture Notes in Artificial Intelligence*, pages 260–265. Springer-Verlag, 2007.
18. Ian P. Gent. Arc consistency in SAT. In F. van Harmelen, editor, *15th European Conference on Artificial Intelligence (ECAI 2002)*, pages 121–125. IOS Press, 2002.
19. Vibhav Gogate and Rina Dechter. A complete anytime algorithm for treewidth. In *Proceedings of the Proceedings of the Twentieth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 201–208, Arlington, Virginia, 2004. AUAI Press.
20. Martin Charles Golumbic and Udi Rotics. On the clique-width of some perfect graph classes. *Internat. J. Found. Comput. Sci.*, 11(3):423–443, 2000. Selected papers from the Workshop on Graph-Theoretical Aspects of Computer Science (WG 99), Part 1 (Ascona).
21. Michel Habib and Christophe Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.
22. Pinar Heggernes, Daniel Meister, and Charis Papadopoulos. Characterising the linear clique-width of a class of graphs by forbidden induced subgraphs. *Discr. Appl. Math.*, 160(6):888–901, 2012.
23. Pinar Heggernes, Daniel Meister, and Udi Rotics. Computing the clique-width of large path powers in linear time via a new characterisation of clique-width. In Alexander S. Kulikov and Nikolay K. Vereshchagin, editors, *Computer Science - Theory and Applications - 6th International Computer Science Symposium in Russia, CSR 2011, St. Petersburg, Russia, June 14-18, 2011. Proceedings*, volume 6651 of *Lecture Notes in Computer Science*, pages 233–246. Springer Verlag, 2011.
24. Eivind Magnus Hvidevold, Sadia Sharmin, Jan Arne Telle, and Martin Vatshelle. Finding good decompositions for dynamic programming on dense graphs. In Dániel Marx and Peter Rossmanith, editors, *Parameterized and Exact Computation - 6th International Symposium, IPEC 2011, Saarbrücken, Germany, September 6-8, 2011. Revised Selected Papers*, volume 7112 of *Lecture Notes in Computer Science*, pages 219–231. Springer Verlag, 2012.
25. Hadi Katebi, Karem A. Sakallah, and Igor L. Markov. Conflict anticipation in the search for graph automorphisms. In Nikolaj Bjørner and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings*, volume 7180 of *Lecture Notes in Computer Science*, pages 243–257. Springer Verlag, 2012.
26. Arie M. C. A. Koster, Hans L. Bodlaender, and Stan P. M. van Hoesel. Treewidth: Computational experiments. *Electronic Notes in Discrete Mathematics*, 8:54–57, 2001.
27. Choongbum Lee, Joonkyung Lee, and Sang-il Oum. Rank-width of random graphs. *J. Graph Theory*, 70(3):339–347, 2012.
28. Brendan D. McKay. Practical graph isomorphism. In *Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. I (Winnipeg, Man., 1980)*, volume 30, pages 45–87, 1981.
29. Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Transactions on Algorithms*, 5(1), 2008.
30. Sang-il Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

31. Marko Samer and Helmut Veith. Encoding treewidth into SAT. In *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 45–50. Springer Verlag, 2009.
32. Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer Verlag, 2005.
33. J. Cole Smith, Elif Ulusal, and Illya V. Hicks. A combinatorial optimization algorithm for solving the branchwidth problem. *Comput. Optim. Appl.*, 51(3):1211–1229, 2012.
34. Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272, 2009.
35. Toby Walsh. SAT v CSP. In R. Dechter, editor, *6th International Conferenc on Principles and Practice of Constraint Programming (CP 2000)*, volume 1894 of *Lecture Notes in Computer Science*, pages 441–456. Springer Verlag, 2000.
36. Egon Wanke. k-NLC graphs and polynomial algorithms. *Discr. Appl. Math.*, 54(2-3):251–266, 1994. Efficient algorithms and partial k-trees.
37. Eric Weisstein. MathWorld online mathematics resource. mathworld.wolfram.com