A Characterization of Graphs with Vertex Cover up to Five

Kevin Cattell[1] and Michael J. Dinneen[1,2]

[1] Department of Computer Science, University of Victoria,
P.O. Box 3055, Victoria, B.C. Canada V8W 3P6
[2] Computer Research and Applications, Los Alamos National Laboratory,
M.S. B265, Los Alamos, New Mexico 87545 U.S.A.

Abstract. For the family of graphs with fixed-size vertex cover k, we present all of the forbidden minors (obstructions), for k up to five. We derive some results, including a practical finite-state recognition algorithm, needed to compute these obstructions.

1 Introduction

The proof of Wagner's conjecture by Robertson and Seymour (see [RS85,RS]), now known as the Graph Minor Theorem (GMT), has led to an explosion of interest in obstruction sets. Though the GMT is primarily of theoretical interest, our research group has been exploring applications of the theory. We have developed a system called VACS to help determine obstruction sets for certain graph families. One of these families, vertex cover, is the subject of this paper.

We make two main contributions in this paper. First, we present a linear-time algorithm that determines the vertex cover for the class of graphs with *bounded-pathwidth* (partial t-paths). There are related results as discussed below, but the algorithm we present has an important property of being *minimal* (defined in Section 3). It is this property which allows us to compute the second contribution of this paper: the obstruction sets for the first five fixed-parameter instances of the vertex cover problem.

The general problem of determining if a graph has a vertex cover of size k, with k part of the input, is well known to be $\mathcal{NP}$-complete [GJ79]. However, several $\mathcal{NP}$-complete problems have polynomial-time *fixed-parameter* versions for some fixed, problem-specific integer k. Vertex cover is an example of such a problem; the brute force approach of checking all k subsets of the vertices gives a crude $O(n^{k+2})$ algorithm. Alternatively, if a tree or path decomposition is available, determining the minimal vertex cover of a graph can be done linear time [ALS91]. In addition to our obstruction set characterizations, we present a practical, finite-state algorithm for path-decomposed graphs. Furthermore, by the facts (1) that path decompositions for fixed k can be found in linear time (see [Bod93,Klo93]) and (2) a pathwidth bound exists (see Theorem 10), we have an $O(n)$ algorithm. Interestingly, a direct fixed-parameter algorithm is presented in [DF] with the same complexity.

The rest of this paper is organized as follows. Section 2 formally defines the vertex cover problem, and introduces results and notation used in the paper. Section 3 presents our general vertex-cover algorithm and its minimal, finite-state variation for graphs of bounded pathwidth. Next, Section 4 contains results that reduce the amount of work needed to compute the obstructions sets. Finally, Section 5 presents the obtained obstructions sets.

2 Background

The most famous example of an obstruction set is found in Kuratowski's Theorem for planar graphs. It states that a graph G is planar if and only if G does not homeomorphically contain the complete bipartite graph $K_{3,3}$ or the complete graph K_5. This indicates the form of all obstruction set characterization of graph families; for some fixed graph family $\mathcal{F}$, $G \in \mathcal{F}$ if and only if G does not contain (under some partial order) any member of some set of graphs $\mathcal{O}(\mathcal{F}) = \{O_1, O_2, \ldots\}$.

For two graphs G and H, the graph H is a *minor* of the graph G if a graph isomorphic to H can be obtained from G by taking a subgraph and then contracting (possibly zero) edges. The GMT states that any set of finite graphs is a well-partial order under the minor order. A family $\mathcal{F}$ of graphs is a *lower ideal* (under the minor order) if $G \in \mathcal{F}$ implies that $H \in \mathcal{F}$ for any minor H of G. An *obstruction* O for a lower ideal $\mathcal{F}$ is a minor-order minimal graph not in $\mathcal{F}$. Hence, by using the GMT, a complete set of obstructions provides a *finite characterization* for any minor-order lower ideal.

For the remainder of this section, we formally define the vertex-cover lower ideals that are characterized in this paper along with other preliminary material. We first define the general vertex-cover decision problem (see [GJ79]) as follows:
Problem: Vertex Cover
Input: Graph $G = (V, E)$ and a positive integer $k \leq |V|$.
Question: Is there a subset $V' \subseteq V$ with $|V'| \leq k$ such that V' contains at least one vertex from every edge in E?

A set V' in the above problem is called a *vertex cover* for the graph G. The family of graphs that have a vertex cover of size at most k will be denoted by VC–k. For a given graph G, let $VC(G)$ denote the least k such that G has a vertex cover of cardinality k.

Lemma 1. *The graph family* VC–k *is a lower ideal in the minor order.*

Proof. Assume a graph $G(V, E) \in$ VC–k has a minimal vertex cover $V' \subseteq V$. If $H = G \setminus (u, v)$ for some $(u, v) \in E$ (edge deletion), then V' is also a vertex cover for H. Likewise, if $u \in V$ is an isolated vertex of G, V' also covers $H = G \setminus \{u\}$ (vertex deletion). For any edge $(u, v) \in E$, observe that $|\{u, v\} \cap V'| \geq 1$. Let w be the new vertex created from u and v in $H = G/(u, v)$ (edge contraction). Clearly, $V'' = (V' \cup \{w\}) \setminus \{u, v\}$ is a vertex cover of H with cardinality at most k. Since any minor of G can be created by repeating the above operations, VC–k is a lower ideal.

Our computational system works with graphs of *bounded pathwidth*, which are defined below (see [CD] for detailed information). These graphs are somewhat related to the graphs of *bounded treewidth* characterized in [Ros73].

Definition 2. A *path-decomposition* of a graph $G = (V, E)$ is a sequence $X_1, X_2, \ldots, X_r$ of subsets of V that satisfy the following three conditions:

1. $\bigcup_{1 \le i \le r} X_i = V$,
2. for every edge $(u, v) \in E$, there exists an X_i, $1 \le i \le r$, such that $u \in X_i$ and $v \in X_i$, and
3. for $1 \le i < j < k \le r$, $X_i \cap X_k \subseteq X_j$.

The *pathwidth of a path-decomposition* $X_1, X_2, \ldots, X_r$ is $\max_{1 \le i \le r} |X_i| - 1$. The *pathwidth of a graph* G is the minimum pathwidth over all path-decompositions of G. Finding pathwidth is equivalent to many problems such as *gate matrix layout* and *vertex separation* [Möh90,EST87,KT92].

The family of graphs of pathwidth t or less, denoted by PW–t, can be represented by strings of operators from some *operator set*. There are many operators sets that can be used (e.g., for treewidth see [Wim87,ACPS91]), and we have chosen one of ours that eases both theory and implementation.

Our operator set Σ_t for bounded pathwidth graphs is defined by

$$\begin{aligned} \Sigma_t &= V_t \cup E_t \quad \text{where} \\ V_t &= \{ \textcircled{0}, \ldots, \textcircled{t} \} \quad \text{and} \\ E_t &= \{ \boxed{i\ j} \ : \ i, j \in V_t, i \neq j \}. \end{aligned}$$

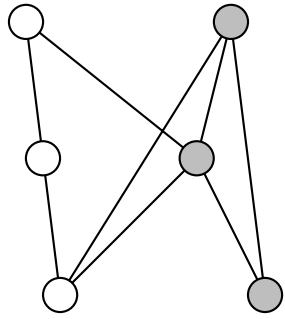
The semantics of these operators on $(t+1)$-boundaried graphs are as follows:

- $\textcircled{i}$ Add an isolated vertex to the graph, and label it as the new boundary vertex i.
- $\boxed{i\ j}$ Add an edge between boundary vertices i and j (ignore if operation causes a self loop).

A graph described by a string of these operators is called a *t-parse*, and has an implicit labeled boundary ∂ of $t+1$ vertices. By convention, a t-parse always begins with the string $[\textcircled{0}, \textcircled{1}, \ldots, \textcircled{t}]$ which represent the edgeless graph of order $t+1$. When G is any t-parse and $Z \in \Sigma_t^*$ is any sequence of operators from the operator set Σ_t the *concatenation* of G and Z forms a new t-parsedenoted by $G \cdot Z$. The labeled boundary of the graph described by $G \cdot Z$ is different from G if the *extension* Z contains any 'new vertex' operators from V_t.

In [CD] it is shown that a graph G has a t-parse representation if and only if $G \in$ PW–t.

Example 3. A t-parse with $t = 2$ and the graph it represents. (The shaded vertices denote the final boundary.)



$$[\,⓪,\ ①,\ ②,\boxed{0\ 1},\boxed{1\ 2},\ ①,\boxed{0\ 1},\boxed{1\ 2},\ ⓪,\boxed{0\ 1},\boxed{0\ 2},\ ②,\boxed{0\ 2},\boxed{1\ 2}\,]$$

3 Finite State Algorithm

In this section we give a practical, finite-state algorithm for the vertex cover problem on graphs of bounded pathwidth in t-parse form. This linear-time algorithm is a dynamic program that makes a single left to right scan of a t-parse $G_n = [g_1, g_2, \ldots, g_n]$. The computational process resembles a finite-state automaton in that it accepts words over the operator alphabet Σ_t. Let m be the current scan position of the algorithm on input G_n. The *state table* at operator g_m is indexed by each subset S of the boundary ∂. These 2^{t+1} different entries are defined as follows:

$$V_m(S) = \min\{\ |V'|\ :\ V' \text{ is a vertex cover of } G_m \text{ and } V' \supseteq S\}$$

Two important observations about the state table are:

1. For each boundary subset $S \in \partial$, $V_m(S)$ is a non-decreasing sequence of non-negative integers as m increases.
2. For any boundary subset $S \in \partial$ and any boundary vertex $i \notin S$, either $V_m(S) = V_m(S \cup \{i\})$ or $V_m(S) = V_m(S \cup \{i\}) - 1$.

The algorithm, given in Fig. 1, starts by setting the sizes for the minimal vertex covers on the empty graph $G_{t+1} = [\,⓪, ①, \ldots, ⓣ\,]$, for all subsets S of the initial boundary ∂.

The type of the operator g_{m+1} (a vertex operator or an edge operator) determines how the state table is updated during the scan. The update of an entry for a specific subset of the boundary S is further broken up according to the relationship between S and the operator. These transitions are described in cases 1–4 of Fig. 1.

When the algorithm reaches the end of the t-parse, it has computed the minimum number of vertices needed for a vertex cover of G_n. This is because

I For $m = t + 1$, set for all $S \in 2^{\partial}$

$$V_{t+1}(S) = |S|$$

II For $t + 1 < m < n$, do the following cases:

Case 1: vertex operator $\textcircled{i}$ and $i \notin S$

$$V_{m+1}(S) = V_m(S)$$

Case 2: vertex operator $\textcircled{i}$ and $i \in S$

$$V_{m+1}(S) = V_m(S \setminus \{i\}) + 1$$

Case 3: edge operator $\boxed{i\ j}$, where $i \in S$ or $j \in S$

$$V_{m+1}(S) = V_m(S)$$

Case 4: edge operator $\boxed{i\ j}$, where $i \notin S$ and $j \notin S$

$$V_{m+1}(S) = \min\{V_m(S \cup \{i\}), V_m(S \cup \{j\})\}$$

III The vertex cover of G is

$$V_n(\emptyset)$$

Fig. 1. General vertex cover algorithm for t-parses.

the entry $V_n(\emptyset)$ contains the size of the smallest vertex cover that contains the subset $\emptyset$ of the boundary. As this is an empty condition, $V_n(\emptyset)$ is the size of the smallest vertex cover in G_n.

Theorem 4. *For any t-parse $G_n = [g_1, g_2, \ldots, g_n]$, the algorithm in Fig. 1 correctly computes $VC(G_n)$.*

Proof. If G is the empty graph then only steps I and III are executed and the correct result of $VC(G) = 0$ is returned. Assume that the algorithm is correct for all (prefix-) graphs of length m and less. We show that cases 1–4 of step II correctly update the state table, $V_{m+1}(S)$ for $S \in 2^{\partial}$.

Case 1: $g_{m+1} = \textcircled{i}$ and $i \notin S$

Let V' be a witness vertex cover for $V_m(S)$. Since the new vertex created by g_{m+1} does not add any edges, V' is a vertex cover for G_{m+1}. Since $V' \supseteq S$ for G_m and $i \notin S$, $V' \supseteq S$ for G_{m+1}. Therefore, $V_{m+1}(S) \leq V_m(S)$.

Let V' be a witness vertex cover for $V_{m+1}(S)$. Since V' is minimal, the isolated vertex created by g_{m+1} is not in V'. Thus V' is a vertex cover for G_m. Since the property $V' \supseteq S$ is preserved, $V_m(S) \leq V_{m+1}(S)$.

Case 2: $g_{m+1} = \textcircled{i}$ and $i \in S$

Let $S' = S \setminus \{i\}$ and V' be a witness vertex cover for $V_m(S')$. Now $W = V' \cup \{i\}$ is a vertex cover for G_{m+1} such that $W \supseteq S$. So, $V_{m+1}(S) \leq V_m(S') + 1$.

For the other direction, let V' be a witness vertex cover for $V_{m+1}(S)$. Since the new boundary vertex i does not help in any vertex cover of G_m, $V'' = V' \setminus \{i\}$ is a vertex cover for G_m such that $V'' \supseteq S$. Hence $V_{m+1}(S) \geq V_m(S') + 1$.

Case 3: $g_{m+1} = \boxed{i\ j}$ where $i \in S$ or $j \in S$

Let V' be a witness vertex cover for $V_m(S)$. Since the boundary is not changed by the edge operator g_{m+1} and $i \in S$ or $j \in S$, V' also covers the edges of G_{m+1}. Thus, $V_{m+1}(S) \leq V_m(S)$. If V'' is a vertex cover of G_{m+1} with $i \in S$ or $j \in S$, then V'' also covers the edges of G_m. So $V_{m+1}(S) \geq V_m(S)$.

Case 4: $g_{m+1} = \boxed{i\ j}$ where $i \notin S$ and $j \notin S$

Let $S' = S \cup \{i\}$ and V' be a witness vertex cover for $V_m(S')$. Since $V' \supseteq S$ is a vertex cover for G_{m+1}, as vertex i is in V', $V_{m+1}(S) \leq V_m(S')$. Likewise, if $S'' = S \cup \{j\}$, then $V_{m+1}(S) \leq V_m(S'')$. So $V_{m+1}(S) \leq \min\{V_m(S \cup \{i\}), V_m(S \cup \{j\})\}$.

Let V' be a witness vertex cover for $V_{m+1}(S)$. Since (i, j) is an edge, either $i \in V'$ or $j \in V'$. Thus $V' \supseteq S \cup \{i\}$ or $V' \supseteq S \cup \{j\}$. If $V' \supseteq S \cup \{i\}$, then V' is a vertex cover for G_m, and so $V_m(S \cup \{i\}) \leq V_{m+1}(S)$. Otherwise, $V' \supseteq S \cup \{j\}$, and $V_m(S \cup \{j\}) \leq V_{m+1}(S)$. Therefore, $\min\{V_m(S \cup \{i\}), V_m(S \cup \{j\})\} \leq V_{m+1}(S)$.

Example 5. The following table shows the application of the algorithm to the *t*-parse given in Example 3. As can been seen by examining the graph in Example 3, a minimum vertex cover has cardinality 3, which equals $V_{14}(\emptyset)$.

S \ m	3	4	5	6	7	8	9	10	11	12	13	14
g_m	–	0 1	1 2	①	0 1	1 2	⓪	0 1	0 2	②	0 2	1 2
$\emptyset$	0	1	1	1	2	2	2	2	3	3	3	**3**
$\{0\}$	1	1	2	2	2	2	3	3	3	3	3	3
$\{1\}$	1	1	1	2	2	2	2	2	3	3	3	3
$\{2\}$	1	2	2	2	2	2	2	3	3	4	4	4
$\{0,1\}$	2	2	2	3	3	3	3	3	3	3	3	3
$\{0,2\}$	2	2	2	2	2	2	3	3	3	4	4	4
$\{1,2\}$	2	2	2	3	3	3	3	3	3	4	4	4
$\{0,1,2\}$	3	3	3	3	3	3	4	4	4	4	4	4

The following lemma shows that we can limit the vertex-cover membership algorithm to a finite number of possible configurations when testing for membership in VC–k.

Lemma 6. *The algorithm in Fig. 1 is finite state for any fixed upper-bound k.*

Proof. We show that for fixed k, there are only a finite number of possible states. Consider the state table entry for a boundary subset S. If $V_i(S)$ becomes $k+1$ for some i, then the monotonicity of $V_m(S)$ guarantees that $V_j(S) \geq k+1$ for all $j > i$. As we are only interested in knowing whether or not there exists a vertex cover of size k containing S, we can restrict $V_m(S)$ to be in $\{0, 1, 2, \ldots, k, k+1\}$. As there are 2^{t+1} entries in the state table, the number of states is bounded by $(k+2)^{2^{t+1}}$.

To make a fixed-parameter algorithm for VC–k, change any update function $V_{m+1}(S) = f(V_m)$ in the four cases with $V_{m+1}(S) = \min(f(V_m), k+1)$. It is straightforward to verify that this modified algorithm correctly computes the same state table except that any entry greater than $k+1$ is replaced by $k+1$.

We define the *final state of a t-parse G*, denoted by V_G, to be the state of the finite-state algorithm when the algorithm terminates. For each boundary subset S, let $V_G(S)$ denote the S entry of V_G. If G and H are t-parses and $V_G = V_H$, it follows immediately that for any operator string $Z \in \Sigma_t^*$, we have $V_{G \cdot Z} = V_{H \cdot Z}$. This in turn implies that $G \cdot Z \in \mathcal{F} \Leftrightarrow H \cdot Z \in \mathcal{F}$ for all Z. That is, G and H *agree on all extensions*. The converse of this property is described by the following important definition.

Definition 7. A finite state algorithm for a family $\mathcal{F}$ is *minimal* if for any two t-parses G and H satisfying

$$G \cdot Z \in \mathcal{F} \Leftrightarrow H \cdot Z \in \mathcal{F} \text{ for all } Z \in \Sigma_t^*$$

then the final states of the algorithm are equal for the inputs G and H.

To show that our vertex-cover algorithm is minimal, we need to show that if G and H are t-parses, and G and H agree on all extensions, then $V_G = V_H$. We will show the contrapositive; that is, if $V_G \neq V_H$, then there exists an extension Z such that G and H do not agree on Z (that is, there exists Z such that either $G \cdot Z \in F$ and $H \cdot Z \notin F$, or $G \cdot Z \notin F$ and $H \cdot Z \in F$).

Before proving that our VC–k algorithm is minimal, we need the following lemma that provides us with an available boundary vertex for building such an extension Z.

Lemma 8. *If G and H are t-parses such that $V_G(\partial) \neq V_H(\partial)$, then there exists an $S \subset \partial$ such that $V_G(S) \neq V_H(S)$.*

Proof. Assume that $V_G(\partial) \neq V_H(\partial)$ is the only difference in the state table. The following three facts

1. $V_G(\partial \setminus \{i\}) = V_H(\partial \setminus \{i\})$ for all $i \in \partial$,
2. $V_G(\partial \setminus \{i\}) \leq V_G(\partial) \leq V_G(\partial \setminus \{i\}) + 1$ for all $i \in \partial$ and
3. $V_H(\partial \setminus \{i\}) \leq V_H(\partial) \leq V_H(\partial \setminus \{i\}) + 1$ for all $i \in \partial$

imply that

$$V_G(\partial) \leq V_G(\partial \setminus \{i\}) + 1 = V_H(\partial \setminus \{i\}) + 1 \leq V_H(\partial) + 1 \text{ for all } i \in \partial$$

and

$$V_G(\partial) \geq V_G(\partial \setminus \{i\}) = V_H(\partial \setminus \{i\}) \geq V_H(\partial) - 1 \text{ for all } i \in \partial \ .$$

After combining the above, $V_H(\partial) - 1 \leq V_G(\partial) \leq V_H(\partial) + 1$. So, without loss of generality, assume $V_G(\partial) = V_H(\partial) - 1 = d$. From this identity and facts 1 and 3 above (also see the partial state tables below), we must have $V_G(\partial) = V_G(\partial \setminus \{i\}) = d$ for all $i \in \partial$.

graph G	
$V_G(\partial)$	d
$V_G(\partial \setminus \{i\})$	$d-1$ or d

$<$

graph H	
$V_H(\partial)$	$d+1$
$V_H(\partial \setminus \{i\})$	d or $d+1$

This can happen if and only if each of the boundary vertices of G are attached to some non-boundary vertex. If not, then a vertex cover $V' \supseteq \partial$ of G would have a redundant vertex $i \in \partial$. The vertex cover created by eliminating vertex i from V' contradicts the value of $V_G(\partial \setminus \{i\})$. However, such a graph G can not exist since the last vertex operator can only have boundary vertex neighbors. Therefore, we can conclude that $V_G(\partial) = V_H(\partial)$ or there exists a $S \subset \partial$ such that $V_G(S) \neq V_H(S)$.

Theorem 9. *The finite-state algorithm in Fig. 1 is minimal for* VC–k.

Proof. Let G and H be t-parses. As discussed above, we show that if $V_G \neq V_H$, then there exists an extension Z such that G and H do not agree on Z. Note that the theorem holds trivially if either one of G or H is not in $\mathcal{F} = \text{VC–}k$ by the empty extension $Z = [\,]$. If both $G \notin \mathcal{F}$ and $H \notin \mathcal{F}$ then $V_G = V_H = [k+1, k+1, \ldots, k+1]$ since $V_G(\emptyset) = k+1$ implies $V_G(S) = k+1$ for all $S \subseteq \partial$.

So suppose that $V_G \neq V_H$. Then without loss of generality, there is a boundary subset S with minimum cardinality such that $V_G(S) < V_H(S) < k+1$. Lemma 8 guarantees that $S \neq \partial$.

Let $\{v_1, v_2, \ldots, v_{|S|}\}$ be the boundary vertices in S. Pick a boundary vertex $i \notin S$ and any other boundary vertex $j \neq i$. Construct an extension Z as follows.

$$Z = [\textcircled{i}, \boxed{i\ v_1}, \textcircled{i}, \boxed{i\ v_2}, \ldots \textcircled{i}, \boxed{i\ v_{|S|}}, \overbrace{\textcircled{i}, \textcircled{j}, \boxed{i\ j}, \ldots, \textcircled{i}, \textcircled{j}, \boxed{i\ j}}^{k-V_H(S)+1 \text{ times}}]$$

The extension Z essentially forces the boundary vertices S to be covered while adding $k - V_H(S) + 1$ isolated edges. Now, $\text{VC}(H \cdot Z)$ is given by $V_H(S) + (k - V_H(S) + 1)$, which equals $k+1$, and so $H \cdot Z \notin \mathcal{F}$. However, $\text{VC}(G \cdot Z)$ is $V_G(S) + (k - V_H(S) + 1) < V_H(S) + (k - V_H(S) + 1) = k+1$, and so $G \cdot Z \in \mathcal{F}$. Therefore, the t-parses G and H do not agree on all extensions.

4 VC–k Obstructions

Two ingredients suffice to compute obstruction sets for a lower ideal $\mathcal{F}$. First, we need to know a bound on the pathwidth (or treewidth) of the obstruction set. Such a bound always exists, as the obstruction set is finite. Given such a bound, we can compute all of the obstructions by restricting our search to a fixed pathwidth. Second, we require a minimal finite-state algorithm for $\mathcal{F}$ that operates on t-parses. An overview of how such an algorithm is used is given in Section 5. Our approach for computing obstruction sets is derived from the two theoretical approaches that appear in [FL89] and [LA91].

For vertex cover, we have both of these ingredients. A minimal finite-state algorithm was described in the previous section, and a pathwidth bound is shown later in this section.

In summary, computing the obstruction set for a vertex-cover family VC–k is as follows:

input: • pathwidth t
• minimal finite-state algorithm for VC–k, that operates on t-parses
output: • obstructions of pathwidth t

The next two subsections show what value of t is needed to get the complete set of obstructions $\mathcal{O}(\text{VC–}k)$ for VC–k and why we can restrict our search to connected graphs.

4.1 Pathwidth of VC–k Obstructions

As discussed, a bound is required on the pathwidth of the obstruction set. That is, we need a result of the form *if $G \in \mathcal{O}$(VC–k), then G is of pathwidth k' or less.* For vertex cover, such a bound is easily obtained. We first show that the family VC–k is contained in the family PW–k. It follows from this that $\mathcal{O}$(VC–k) is contained in PW–$(k+1)$.

Theorem 10. *The pathwidth of any member of* VC–k *is at most* k.

Proof. For a given graph G of VC–k, let V' be a subset of of the vertices of size k that covers all edges. Denote the order of G by n. Let the vertices V of G be indexed by $1, 2, \ldots, n$ with the vertices $V \setminus V'$ coming first. We claim that $\{X_i \mid 1 \leq i \leq n-k\}$ where $X_i = V' \cup \{i\}$ is a path decomposition of G.

Since every vertex is either in V' or is in $V \setminus V'$ we have $\bigcup_{1 \leq i \leq n-k} X_i = V$. Let (u, v) be an edge of G. Since V' is a vertex cover, without loss of generality assume $u \in V'$. If also $v \in V'$ then any subset X_i contains both u and v. Otherwise, v must be indexed between 1 and $n-k$ and the subset X_v contains both u and v. Finally, note that for any $1 \leq i < j \leq n-k$ we have $X_i \cap X_j = V'$ (interpolation property satisfied). Thus, we have a path decomposition of pathwidth k.

The above theorem can not be improved, as the complete graph K_{k+1} with pathwidth k is a member of VC–k.

Corollary 11. *If $G \in \mathcal{O}$(VC–k), then the pathwidth of G is at most $k+1$.*

Proof. For any edge $(u, v) \in E(G)$, let $G' = G \setminus (u, v)$. Since G is an obstruction for VC–k, $G' \in$ VC–k and hence VC$(G') \leq k$ by Theorem 10. Let V' be a witness vertex cover for G'. Now $V = V' \cup \{u\}$ is a vertex cover for G of order at most $k+1$. Therefore, by Theorem 10 again, the pathwidth of G is at most $k+1$.

4.2 Disconnected Obstructions

The number of obstructions we need to find can be reduced by some straightforward observations. The following are special cases of the more general results found in [CD].

Remark 12. Let C_1 and C_2 be graphs. Then VC$(C_1 \cup C_2)$ = VC(C_1) + VC(C_2).

Lemma 13. *If $O = C_1 \cup C_2$ is an obstruction for* VC–k, *then C_1 and C_2 are obstructions for* VC–k' *and* VC–k'', *respectively, for some $0 < k', k'' < k$, with $k' + k'' = k - 1$.*

Hence we can restrict our attention to connected obstructions; any disconnected obstruction O of VC–k is a union of graphs from $\bigcup_{i=0}^{k-1} \mathcal{O}$(VC–$i$) such that VC$(O) = k+1$.

Example 14. Since K_3 is an obstruction for VC–1, and K_4 is an obstruction for VC–2, the graph $K_3 \cup K_4$ is an obstruction for VC–4.

5 Results

The obstructions were computed using our VACS machinery (described in [CD]), which allows us to compute for any t all of the obstructions that have pathwidth at most t. However, tractability problems arise as t increases. As shown in the preceding section, we need to use pathwidth $k+1$ to obtain all of the obstructions for VC–k.

A brief description of the obstruction set computation is as follows. The set of all t-parses can be viewed as a tree, in which the parent of a length n t-parse G is the length $n-1$ prefix of G. The root of the tree is the empty graph $[\,⓪, ①, \ldots, ⓣ\,]$. The minimality of the finite-state algorithm allows us to compute a 'pruning rule' for the tree. When this is done, the tree becomes finite, and the t-parses represented by the leaves form a set closely related to the VC–k obstruction set. This set, in fact, contains the obstruction set for VC–k.

A summary of our obstruction set computations (using a SPARC-2) for various VC–k families is shown in Table 1. The total graphs column shows the size of the pruned tree described above. In the minimal graphs column, the number of internal graphs plus obstructions is shown; that is, the leaves that are not obstructions have not been counted. The growth rate of the tree can be seen to be extremely high as k (and hence the pathwidth t) increases.

Table 1. Summary of obstruction set computation for vertex cover.

k	Elapsed time	Minimal graphs	Total graphs	Connected obstructions	Total obstructions
1	5 seconds	8	31	1	2
2	25 seconds	42	301	2	4
3	3 minutes	320	3,871	3	8
4	4 hours	4,460	82,804	8	18
5	6 days	121,228	3,195,445	31	56

Besides the single obstruction K_2 for the trivial family VC–0, the connected obstructions for VC–k, $1 \leq k \leq 5$, are shown in Figs. 2–6. Some patterns become apparent in this set of obstructions. One such easily-proven observation is as follows.

Remark 15. For the family VC–k, both the complete graph K_{k+2} and the cycle C_{2k+1} are obstructions.

6 Conclusions

In this paper, we have presented the minor-order obstruction sets for the graph families VC–k, $1 \leq k \leq 5$. To calculate these obstructions, a minimal finite-state algorithm was developed. This algorithm was described and proven to be correct.

We are hopeful that this paper is the first in a succession of successful computations of obstruction sets for the plethora of graph families that are lower ideals in the minor order (e.g., k feedback vertex set and within-k-vertices of planarity). The system used to compute the obstruction sets is constantly improving, allowing us to reach higher pathwidths and more complicated problems.

Acknowledgement

The authors wish to thank Mike Fellows for introducing us to this exciting field of study and Todd Wareham for comments on an earlier version of this paper.

References

[ACPS91] Stefan Arnborg, Derek G. Corneil, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. In *Proceedings of the Fourth Workshop on Graph Grammars and Their Applications to Computer Science*, volume 532, pp. 70–83. Lecture Notes in Computer Science, Springer-Verlag, 1991. To appear in Journal of the ACM.

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, **12** (1991), pp. 308–340.

[Bod93] Hans L. Bodlaender. A linear time algorithm for finding tree-decompostions of small treewidth. In *Proc. 25th Annual ACM Symposium on Theory of Computing*. ACM Press, 1993.

[CD] Kevin Cattell and Michael J. Dinneen. VLSI Automated Compilation System – obstruction set computations (technical notes). In preparation.

[DF] Rod Downey and Michael R. Fellows. Parameterized computational feasibility. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*. Birkhauser. To appear.

[EST87] J. Ellis, I. H. Sudborough, and J. Turner. Graph separation and search number. Report DCS-66-IR, Dept. of Computer Science, University of Victoria, August 1987. To appear in *Information and Computation*.

[FL89] Michael R. Fellows and Michael A. Langston. An analogue of the Myhill-Nerode Theorem and its use in computing finite-basis characterizations. In *Proc. Symposium on Foundations of Computer Science (FOCS)*, pp. 520–525, 1989.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[Klo93] Ton Kloks. *Treewidth*. Ph.D. dissertation, Dept. of Computer Science, Utrecht University, Utrecht, the Netherlands, 1993.

[KT92] András Kornai and Zsolt Tuza. Narrowness, pathwidth, and their application in natural language processing. *Discrete Applied Mathematics*, **36** (1992), pp. 87–92.

[LA91] Jens Lagergren and Stefan Arnberg. Finding minimal forbidden minors using a finite congruence. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, volume 510, pp. 533–543. Springer-Verlag, Lecture Notes in Computer Science, 1991.

[Möh90] Rolf H. Möhring. Graph problems releted to gate matrix layout and PLA folding. In G. Tinhofer, E. Mayr, H. Noltemeier, and M. Syslo, editors, *Computational Graph Theory*, pp. 17–51. Springer-Verlag, 1990.

[Ros73] Donald J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, **7** (1973), pp. 17–322.

[RS] Neil Robertson and Paul D. Seymour. Graph Minors. XVI. Wagner's conjecture. To appear *Journal of Combinatorial Theory, Series B.*

[RS85] Neil Robertson and Paul D. Seymour. Graph Minors – A Survey, In *Surveys in Combinatorics*, volume 103, pp. 153–171, Cambridge University Press, 1985.

[Wim87] T. V. Wimer. *Linear algorithms on k-terminal graphs.* Ph.D. dissertation, Dept. of Computer Science, Clemson University, August 1987.

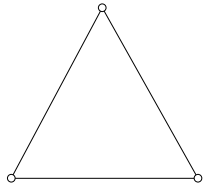


Fig. 2. Connected obstructions for Vertex Cover 1.

Fig. 3. Connected obstructions for Vertex Cover 2.

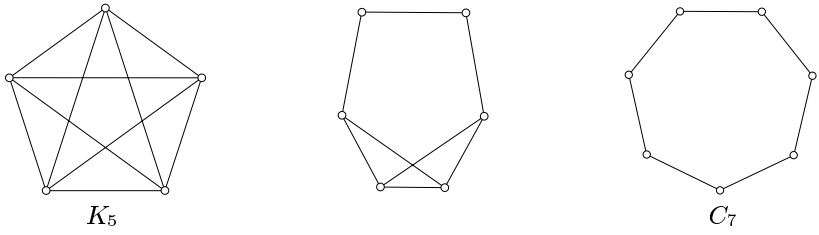


Fig. 4. Connected obstructions for Vertex Cover 3.

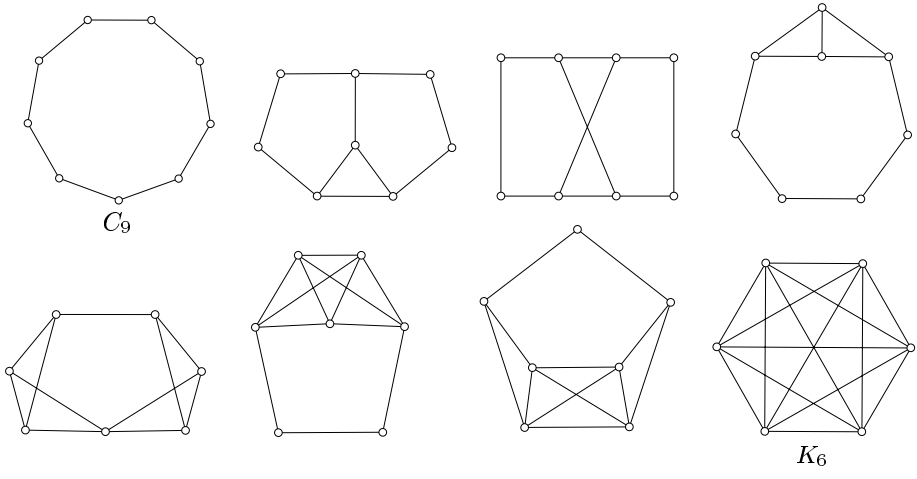


Fig. 5. Connected obstructions for Vertex Cover 4.

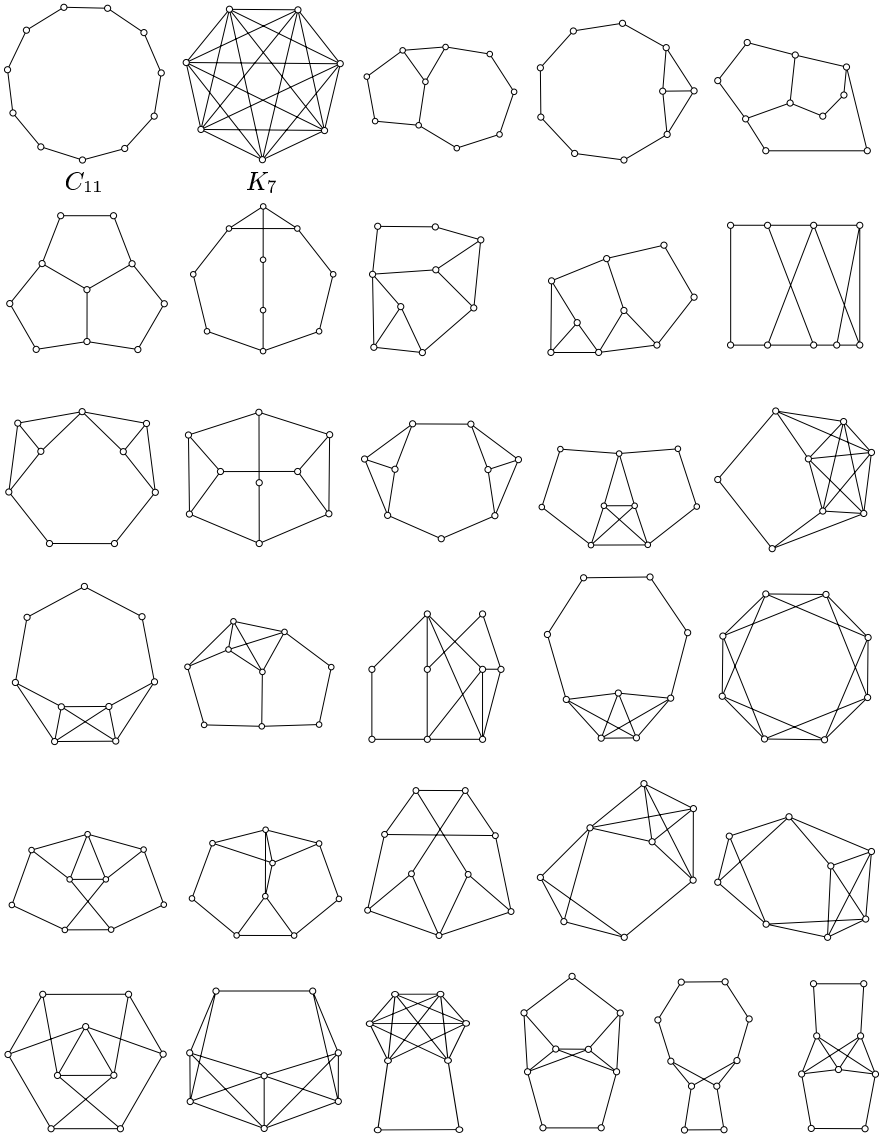


Fig. 6. Connected obstructions for Vertex Cover 5.