# Bounded Persistence Pathwidth

**Rodney G. Downey**[1] **Catherine McCartin**[2]

[1] Victoria University, Wellington, New Zealand
Email: Rod.Downey@mcs.vuw.ac.nz
[2] Massey University, Palmerston North, New Zealand
Email: C.M.McCartin@massey.ac.nz

**Abstract**

The role of graph width metrics, such as treewidth, pathwidth, and cliquewidth, is now seen as central in both algorithm design and the delineation of what is algorithmically possible. In this article we introduce a new, related, parameter for graphs, *persistence*.

A path decomposition of width $k$, in which every vertex of the underlying graph belongs to at most $l$ nodes of the path, has pathwidth $k$ and *persistence* $l$, and a graph that admits such a decomposition has *bounded persistence pathwidth*.

We believe that this natural notion truly captures the intuition behind the notion of pathwidth. We present some basic results regarding the general recognition of graphs having bounded persistence path decompositions.

*Keywords:* Treewidth, Pathwidth, Bounded Persistence Pathwidth, Parameterized Complexity.

## 1 Introduction

The last 20 years has seen a revolution in the development of graph algorithms. This revolution has been driven by the systematic use of ideas from topological graph theory, with the use of graph width metrics emerging as a fundamental paradigm in such investigations. The role of graph width metrics, such as treewidth, pathwidth, and cliquewidth, is now seen as central in both algorithm design and the delineation of what is algorithmically possible.

It is now commonplace to find that by restricting some width parameter for the input graphs, a particular graph problem can be solved efficiently. A number of different graph width metrics naturally arise in this context which restrict the inherent complexity of a graph in various senses. The central idea is that a useful width metric should admit efficient algorithms for many (generally) intractable problems on the class of graphs for which the width is small. One of the most successful measures in this context is the notion of *treewidth* which arose from the seminal work of Robertson and Seymour on graph minors and immersions (Robertson & Seymour 1986). Treewidth measures, in a precisely defined way, how "tree-like" a graph is. The idea here is that we can lift many results from trees to graphs that are "tree-like". Related to treewidth is the notion of *pathwidth* which measures, in the same way, how "path-like" a graph is.



In this article we introduce a new, related, parameter for graphs, *persistence*, which, we believe, truly captures the intuition behind the notion of pathwidth. A path decomposition of width $k$ in which every vertex of the underlying graph belongs to at most $l$ nodes of the path has pathwidth $k$ and *persistence* $l$, and a graph that admits such a decomposition has *bounded persistence pathwidth*.

We present some basic results regarding the general recognition of graphs having bounded persistence pathwidth, using the framework of *parameterized complexity theory*, introduced by Downey and Fellows (Downey & Fellows 1999). Our results show that deciding whether or not a given graph has bounded persistence pathwidth is a parametrically hard problem, whereas the corresponding problem for traditional treewidth or pathwidth is not. Nevertheless, the input for many problems is *naturally* presented in a reasonable way and hence, in spite of the lack of efficient recognition algorithms (as exhibited by the hardness results that we present), we believe that persistence is an important parameter to be exploited. This idea is pursued in (Downey & McCartin 2004), and is part of an overall program of investigation into parameterized *promise* problems, where we are *promised* that the problem input obeys certain properties, or is presented in a certain fashion. In (Downey & McCartin 2004) we consider the algorithmic ramifications of such problems in the online setting.

In the following section we review the basic notions of parameterized complexity, and present definitions and background information on treewidth and pathwidth for graphs. In Section 3 we discuss our notion of *persistence*. In Section 4 we present our results regarding the complexity of recognizing graphs having bounded persistence path decompositions.

## 2 Preliminaries

### 2.1 Parameterized Complexity

To investigate the complexity of basic problems associated with persistence, we use the framework of *parameterized complexity theory*, introduced by Downey and Fellows (Downey & Fellows 1999).

We remind the reader that a parameterized language $L$ is a subset of $\Sigma^* \times \Sigma^*$, where $\Sigma$ is a finite alphabet. If $L$ is a parameterized language and $\langle \sigma, k \rangle \in L$ then we refer to $\sigma$ as the *main part* and $k$ as the *parameter*. The basic notion of tractability is *fixed parameter tractability* (FPT). Intuitively, we say that a parameterized problem is fixed-parameter tractable (FPT) if we can somehow confine any "bad" complexity behaviour to some limited aspect of the problem, the parameter. Formally, we say that a parameterized language, $L$, is fixed-parameter tractable if there is a computable function $f$, an algorithm $A$, and a constant $c$ such that for all $k$, $\langle x, k \rangle \in L$ iff

$A(x,k) = 1$, and $A(x,k)$ runs in time $f(k)|x|^c$ ($c$ is independent of $k$). For instance, $k$-VERTEX COVER is solvable in time $\mathcal{O}(|x|)$. On the other hand, for $k$-TURING MACHINE ACCEPTANCE, the problem of deciding if a nondeterministic Turing machine with arbitrarily large fanout has a $k$-step accepting path, the only known algorithm is to try all possibilities, and this takes time $\Omega(|x|^k)$. This situation, akin to $NP$-completeness, is described by hardness classes, and reductions. A parameterized reduction, $L$ to $L'$, is a transformation which takes $\langle x,k\rangle$ to $\langle x',k'\rangle$, running in time $g(k)|x|^c$, with $k \mapsto k'$ a function purely of $k$.

Downey and Fellows (Downey & Fellows 1999) observed that these reductions gave rise to a hierarchy called the $W$-hierarchy.

$$FPT \subset W[1] \subseteq W[2] \subseteq \ldots \subseteq W[t] \subseteq \ldots.$$

The core problem for $W[1]$ is $k$-TURING MACHINE ACCEPTANCE, which is equivalent to the problem WEIGHTED 3SAT. The input for WEIGHTED 3SAT is a 3CNF formula, $\varphi$ and the problem is to determine whether or not $\varphi$ has a satisfying assignment of Hamming weight $k$. $W[2]$ has the same core problem except that $\varphi$ is in CNF form, with no bound on the clause size. In general, $W[t]$ has as its core problem the weighted satisfiability problem for $\varphi$ of the form "products of sums of products of ..." of depth $t$. It is conjectured that the $W$-hierarchy is proper, and from $W[1]$ onwards, all parametrically intractable.

### 2.2 Treewidth and Pathwidth

Many generally intractable problems become tractable for the class of graphs that have bounded treewidth or bounded pathwidth. Furthermore, treewidth and pathwidth subsume many graph properties that have been previously mooted, in the sense that tractability for bounded treewidth or bounded pathwidth implies tractability for many other well-studied classes of graphs. For example, planar graphs with radius $k$ have treewidth at most $3k$, series parallel multigraphs have treewidth 2, chordal graphs (graphs having no induced cycles of length 4 or more) with maximum clique size $k$ have treewidth at most $k-1$, graphs with bandwidth at most $k$ have pathwidth at most $k$.

A graph $G$ has treewidth at most $k$ if we can associate a tree $T$ with $G$ in which each node represents a subgraph of $G$ having at most $k+1$ vertices, such that all vertices and edges of $G$ are represented in at least one of the nodes of $T$, and for each vertex $v$ in $G$, the nodes of $T$ where $v$ is represented form a subtree of $T$. Such a tree is called a *tree decomposition* of $G$, of *width* $k$. We give a formal definition here:

**Definition 1** [Tree decomposition and Treewidth]

*Let $G = (V,E)$ be a graph. A tree decomposition of $G$ is a pair $(T,\mathcal{X})$ where $T = (I,F)$ is a tree, and $\mathcal{X} = \{X_i \mid i \in I\}$ is a family of subsets of $V$, one for each node of $T$, such that*

1. *$\bigcup_{i\in I} X_i = V$,*
2. *for every edge $\{v,w\} \in E$, there is an $i \in I$ with $v \in X_i$ and $w \in X_i$,*
3. *for all $i,j,k \in I$, if $j$ is on the path from $i$ to $k$ in $T$, then $X_i \cap X_k \subseteq X_j$.*

*The treewidth or width of a tree decomposition $((I,F),\{X_i \mid i \in I\})$ is $max_{i\in I}\ |X_i| - 1$. The treewidth of a graph $G$ is the minimum width over all possible tree decompositions of $G$.*

**Definition 2** [Path decomposition and Pathwidth]

*A path decomposition of a graph $G$ is a tree decomposition $(P,\mathcal{X})$ of $G$ where $P$ is simply a path (i.e. the nodes of $P$ have degree at most two). The pathwidth of $G$ is the minimum width over all possible path decompositions of $G$.*

Any path decomposition of $G$ is also a tree decomposition of $G$, so the pathwidth of $G$ is at least equal to the treewidth of $G$. For many graphs, the pathwidth will be somewhat larger than the treewidth. For example, let $B_k$ denote the complete binary tree of height $k$ and order $2^k - 1$, then $tw(B_k) = 1$, but $pw(B_k) = k$.

Graphs of treewidth and pathwidth at most $k$ are also called *partial $k$-trees* and *partial $k$-paths*, respectively, as they are exactly the subgraphs of $k$-trees and $k$-paths. There are a number of other important variations equivalent to the notions of treewidth and pathwidth (see, e.g., Bodlaender 1996a). For algorithmic purposes, the characterizations provided by the definitions given above tend to be the most useful.

### 2.3 Finding tree and path decompositions

We mentioned above that many intractable problems become tractable for the class of graphs that have bounded treewidth or bounded pathwidth. A more accurate statement would be to say that many intractable problems become *theoretically* tractable for this class of graphs, in the general case.

The typical method employed to produce efficient algorithms for problems restricted to graphs of bounded treewidth (pathwidth) proceeds in two stages (see Bodlaender 1997).

1. Find a bounded-width tree (path) decomposition of the input graph that exhibits the underlying tree (path) structure.
2. Perform dynamic programming on this decomposition to solve the problem.

In order for this approach to produce practically efficient algorithms, as opposed to proving that problems are theoretically tractable, it is important to be able to produce the necessary decomposition reasonably efficiently.

Many people have worked on the problem of finding progressively better algorithms for recognition of bounded treewidth (pathwidth) graphs, and construction of associated decompositions.

As a first step, Arnborg et. al. (Arnborg, Corneil & Proskurowski 1987) showed that if a bound on the treewidth (pathwidth) of the graph is known, then a decomposition that acheives this bound can be found in time $O(n^{k+2})$, where $n$ is the size of the input graph and $k$ is the bound on the treewidth (pathwidth). They also showed that determining the treewidth or pathwidth of a graph in the first place is $NP$-hard.

Robertson and Seymour (Robertson & Seymour 1986) gave the first FPT algorithm, $O(n^2)$, for $k$-TREEWIDTH. Their algorithm, based upon the minor well-quasi-ordering theorem (Robertson & Seymour 1985), is highly non-constructive, non-elementary, and has huge constants.

The early work of (Arnborg, Corneil & Proskurowski 1987), and (Robertson & Seymour 1986) has been improved upon in the work of (Lagergen 1996), (Reed 1992), (Fellows & Langston 1989), (Matousek & Thomas 1991), (Bodlaender 1996), and (Bodlaender & Kloks 1996), among others.

Bodlaender (Bodlaender 1996) gave the first linear-time FPT algorithms for the constructive versions of both $k$-TREEWIDTH and $k$-PATHWIDTH, although the $f(k)$'s involved mean that treewidth and pathwidth still remain parameters of theoretical interest only, at least in the general case.

Bodlaender's algorithms recursively invoke a linear-time FPT algorithm due to Bodlaender and Kloks (Bodlaender & Kloks 1996) which "squeezes" a given width $p$ tree decomposition of a graph $G$ down to a width $k$ tree (path) decomposition of $G$, if $G$ has treewidth (pathwidth) at most $k$. A small improvement to the Bodlaender/Kloks algorithm would substantially improve the performance of Bodlaender's algorithms.

Perkovic and Reed (Perkovic & Reed 2000) have recently improved upon Bodlaender's work, giving a streamlined algorithm for $k$-TREEWIDTH that recursively invokes the Bodlaender/Kloks algorithm no more than $O(k^2)$ times, while Bodlaender's algorithms may require $O(k^8)$ recursive iterations.

For some graph classes, the optimal treewidth and pathwidth, or good approximations of these, can be found using practically efficient polynomial time algorithms. Examples are chordal bipartite graphs, interval graphs, permutation graphs, circle graphs, and co-graphs.

## 3 Bounded Persistence Pathwidth

We say that a path decomposition of width $k$ in which every vertex of the underlying graph belongs to at most $l$ nodes of the path has pathwidth $k$ and *persistence* $l$, and say that a graph that admits such a decomposition has *bounded persistence pathwidth*. We believe that this natural notion truly captures the intuition behind the notion of pathwidth.

A graph that can be presented in the form of a path decomposition with both low width and low persistence is properly pathlike, whereas graphs that have *high* persistence are, in some sense, "unnatural" or pathological. Consider the graph $G$ presented in Figure 1. $G$ is not really path-like, but still has a path decomposition of width only two. The reason for this is reflected in the presence of vertex $a$ in *every* node of the path decomposition. Our underlying idea is that a pathwidth 2 graph should look more like a "long 2-path" than a "fuzzy ball".

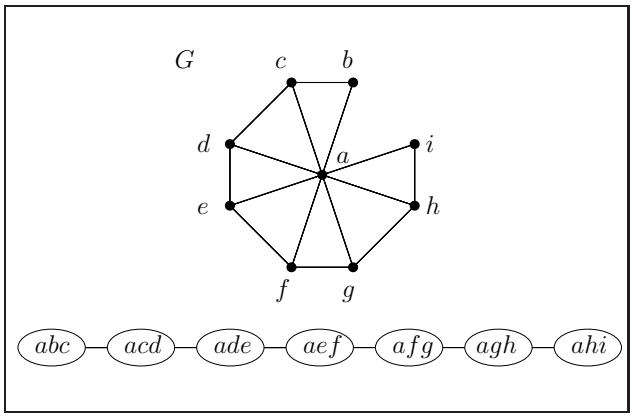


Figure 1: A graph $G$ having low pathwidth but high persistence.

The notion of persistence arose from some preliminary investigations into online presentations for graphs having bounded pathwidth (Downey & McCartin 2004), (McCartin 2003), but it appears to be a natural and interesting parameter in both the online setting *and* the offline setting. For many problems where the input is generated as an ordered sequence of small units, it seems natural to expect that the sphere of influence of each unit of input should be localized.

A related notion is *domino treewidth* introduced by Bodlaender and Engelfriet (Bodlaender & Engelfreit 1997). A *domino tree decomposition* is a tree decomposition in which every vertex of the underlying graph belongs to at most two nodes of the tree. *Domino pathwidth* is a special case of bounded persistence pathwidth, where $l = 2$.

Note that bounded persistence pathwidth gives us a natural characterization of graphs having both bounded pathwidth and bounded degree. If a graph $G$ admits a path decomposition of width $k$ and persistence $l$ then it must be the case that all vertices in $G$ have degree at most $k \cdot l$. On the other hand, if $G$ has pathwidth $k$ and maximum degree $d$ then the persistence that can be achieved in any path decomposition of $G$ must be "traded off" against the resulting width.

## 4 Complexity of Bounded Persistence Pathwidth

We now present some basic results regarding the complexity of recognizing graphs having bounded persistence path decompositions.

Persistence appears to be an interesting parameter in relation to graph width metrics and associated graph decompositions or layouts. However, in contrast to the case for *pathwidth*, which admits an FPT algorithm, deciding whether or not a given graph has *bounded persistence pathwidth* appears to be a hard problem.

We give some strong evidence for the likely parameterized intractability of both the BOUNDED PERSISTENCE PATHWIDTH problem and the DOMINO PATHWIDTH problem. We show that the BOUNDED PERSISTENCE PATHWIDTH problem is $W[t]$-hard, for all $t \in \mathbb{N}$, and we show that the DOMINO PATHWIDTH is $W[2]$-hard. These results mean that is likely to be impossible to find FPT algorithms for either of these problems, at least in the general case, unless an unlikely collapse occurs in the $W$-hierarchy.

Note that, even though we will show that the recognition problems are *hard*, in many real life instances we may reasonably expect to "know" that the persistence is relatively low, and, indeed be given such a decomposition.

A related result from (Bodlaender & Engelfreit 1997) is that finding the domino treewidth of a general graph is $W[t]$-hard, for all $t \in \mathbb{N}$. Our first result relies on the following theorem from (Bodlaender, Fellows & Hallett 1994).

**Theorem 1** *$k$-BANDWIDTH is $W[t]$-hard, for all $t \in \mathbb{N}$.*

$k$-BANDWIDTH is defined as follows:

| | |
|---|---|
| *Instance:* | A graph $G = (V, E)$. |
| *Parameter:* | A positive integer $k$. |
| *Question:* | Is there a bijective *linear layout* of $V$, $f : V \to \{1, 2, \ldots, \|V\|\}$, such that, for all $(u, v) \in E$, $\|f(u) - f(v)\| \leq k$? |

BOUNDED PERSISTENCE PATHWIDTH is defined as follows:

| | |
|---|---|
| *Instance:* | A graph $G = (V, E)$. |
| *Parameter:* | A pair of positive integers $(k, l)$. |
| *Question:* | Is there a path decomposition of $G$ of width at most $k$, and persistence at most $l$? |

DOMINO PATHWIDTH is a special case of this problem, where $l = 2$.

**Theorem 2** BOUNDED PERSISTENCE PATHWIDTH *is $W[t]$-hard, for all $t \in \mathcal{N}$*

**Proof:** We transform from $k$-BANDWIDTH.

Let $G = (V, E)$ be a graph and let $k$ be the parameter. We produce $G' = (V', E')$ such that $G'$ has a width $2(k+1)^2 - 1$, persistence $k+1$, path decomposition iff $G$ has bandwidth at most $k$.

To build $G'$ we begin with the original graph $G$, and alter it as follows:

1. for each vertex $v$ in $G$, we introduce new vertices and form a clique of size $(k+1)^2 + 1$ containing these vertices and $v$, call this $C_v$.
2. for each neighbour $u$ of $v$ (we can assume at most $2k$ of these, otherwise $G$ cannot have bandwidth at most $k$), choose a unique vertex, $c_{v_u}$, from $C_v$ (not $v$) and attach this vertex to all the vertices in $C_u$.

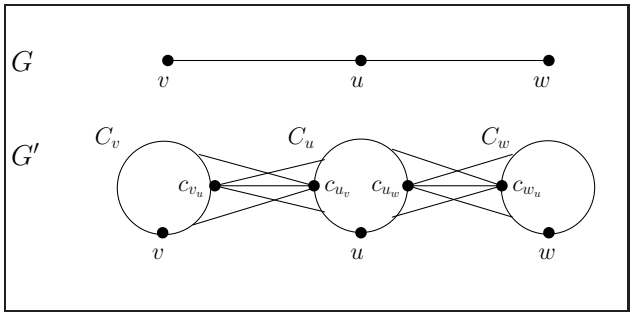


Figure 2: Bounded persistence pathwidth, transformation from $G$ to $G'$.

$\Leftarrow$ If $G$ has bandwidth $k$, then the required decomposition for $G'$ exists.

Let $\{v_0, \ldots, v_n\}$ be a layout of bandwidth $k$ for $G$. To build the decomposition $(P, \mathcal{X})$ for $G'$ we let the $i$th node of the decomposition, $X_i$, contain $\{v_i, \ldots, v_{i+k}\}$ plus $C_i$, plus each $c_{j_r}$ connected to $C_r$ with $j \leq i$ and $r \geq i$ or $r \leq i$ and $j \geq i$.

This fulfills the requirements for a path decomposition.

1. $\bigcup_{i \in I} X_i = V'$,
2. for every edge $\{v, w\} \in E'$, there is an $i \in I$ with $v \in X_i$ and $w \in X_i$, and
3. for all $i, j, k \in I$, if $j$ is on the path from $i$ to $k$ in $P$, then $X_i \cap X_k \subseteq X_j$.

Each node contains at most $(k+1) + k(k+1) + (k+1)^2 = 2(k+1)^2$ vertices. Thus, the decomposition has width $2(k+1)^2) - 1$.

Any $v_i$ from $G$ appears in at most $k+1$ nodes, being nodes $X_{i-k}$ up to $X_i$. Any $c_{j_r}$ appears in at most $k+1$ nodes, being nodes $X_j$ up to $X_r$, or $X_r$ up to $X_j$, where $|j - r| \leq k$. Thus, the decomposition has persistence $k+1$.

$\Rightarrow$ If the required decomposition of $G'$ exists, then $G$ has bandwidth $k$.

At some point in the decomposition, a node $X_{v'}$, containing $C_v$, will appear for the first time. No other $C_u$, $u \neq v$ can appear in this node, as there is not room.

Pick a neighbour $u$ of $v$ for which $C_u$ has already appeared. $C_u$ must have appeared for the first time in some node $X_{u'}$, where $v' - u' \leq k$, since some $c_{u_v}$ was present in this node which must appear with $C_v$ at some point, and $c_{u_v}$ cannot appear in more than $k+1$ nodes.

Pick a neighbour $w$ of $v$ for which $C_w$ has not yet appeared. $C_w$ must appear for the first time some node $X_{w'}$ where $w' - v' \leq k$, since there is some $c_{v_w}$ in $X_{v'}$ which must be present with $C_w$ at some point and $c_{v_w}$ cannot appear in more than $k+1$ nodes.

If we lay out the vertices of $G$ in the order in which the corresponding cliques first appear in the decomposition, then we have a layout of $G$ with bandwidth $k$. □

**Theorem 3** DOMINO PATHWIDTH *is $W[2]$-hard.*

**Proof:** We transform from $k$-DOMINATING SET, a fundamental $W[2]$-complete problem.

$k$-DOMINATING SET is defined as follows:

| | |
|---|---|
| *Instance:* | A graph $G = (V, E)$. |
| *Parameter:* | A positive integer $k$. |
| *Question:* | Does $G$ contain a set of vertices $V' \subseteq V$, of size $k$, such that, $\forall u \in V$, $\exists v \in V'$ with $uv \in E$? |

Let $G$ be a graph and $k$ the parameter. We produce $G'$ such that $G'$ has a width $K - 1$ domino path decomposition, where $K = k^2(k+4) + k(k+3)$, if and only if $G$ has a dominating set of size $k$.

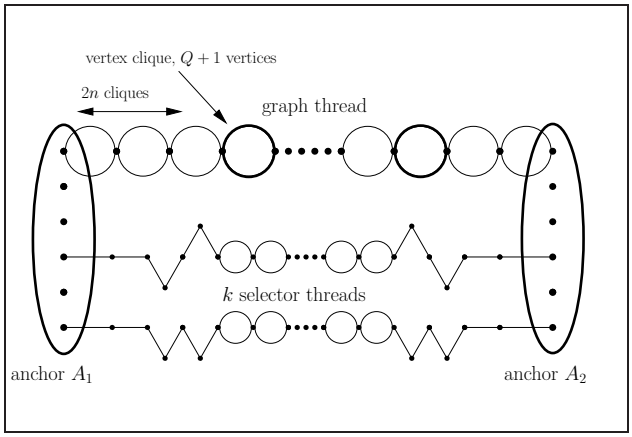


Figure 3: Gadget for domino pathwidth transformation.

$G' = (V, E)$ consists of the following components:

- **Two anchors.** Take two cliques, each with $K$ vertices, with vertex sets $A_1 = \{a_i^1 | 1 \leq i \leq K\}$ and $A_2 = \{a_i^2 | 1 \leq i \leq K\}$.
- **The graph thread.** Let $n = |V|$. Take $P = 2n + n^2 + (n+1)$ cliques, each with $Q = k^2(k+4)$ vertices, with vertex sets $C^i = \{c_r^i | 1 \leq r \leq Q\}$, for $1 \leq i \leq P$. Join them into a "thread" by choosing separate vertices in each clique, $c_{start}^i$ and $c_{end}^i$, and identifying $c_{end}^i$ with $c_{start}^{i+1}$. Identify $c_{start}^1$ with $a_1^1$, and identify $c_{end}^P$ with $a_1^2$. Now, for each $1 \leq i \leq (P-1)$ cliques $C^i$ and $C^{i+1}$ have a vertex in common, $C^1$ has a vertex in common with $A_1$, and $C^P$ has a vertex in common with $A_2$.

- **The vertex cliques.** For each $i$ of the form $i = 2n + j.n + 1$ for $0 \leq j \leq n-1$ take the clique $C^i$ from the graph thread and add another vertex to each such $C^i$, to make a clique with $Q+1$ vertices. Each of these $n$ cliques, $C^i_j$, represents a vertex, $v_j$, of $G$.
- **The selector threads.** Take $2n$ cliques of size 2; $n^2$ cliques of size $k+3$ with vertex sets $S^i = \{s^i_r | 1 \leq r \leq k+3\}$, for $1 \leq i \leq n^2$; and another $2n$ cliques of size 2. Join them into a thread as for the graph thread, so that $2n$ size 2 cliques form the first portion of the thread, and $2n$ size 2 cliques form the last portion of the thread. Now, the first $2n$ size 2 cliques form a simple path having $2n+1$ vertices and $2n$ edges, where the last vertex in this path is also an element of $S^1$. For each $1 \leq i \leq (n^2-1)$ cliques $S^i$ and $S^{i+1}$ have a vertex in common. The last $2n$ size 2 cliques form a simple path having $2n+1$ vertices and $2n$ edges, where the first vertex in this path is also an element of $S^{n^2}$.

  For $1 \leq i \leq n$, let $\mathcal{S}_i$ denote the $i$th consecutive set of $n$ cliques of size $k+3$, $\{S^{(i-1)n+1}, \ldots, S^{(i-1)n+n}\}$.

  Remove one (interior) vertex from the $i$th clique of $\mathcal{S}_i$, $S^{(i-1)n+i}$. If $v_i$ is connected to $v_j$ in $G$, remove one (interior) vertex from the $j$th clique of $\mathcal{S}_i$, $S^{(i-1)n+j}$.

  Make $k$ copies of the selector thread component described here, and identify the first vertex of the $i$th thread with $a^1_{i+1}$, the last vertex of the $i$th thread with $a^2_{i+1}$.

  Each of these threads is used to select a vertex in a dominating set of $G$, if one exists.

$\Leftarrow$ Suppose $G$ has a dominating set of size $k$. Then the required domino path decomposition of $G'$, $PD(G')$, exists.

There are $P+2$ nodes in the decomposition, $\{X_0, \ldots, X_{P+1}\}$. We let $A_1$ be contained in $X_0$. For $1 \leq i \leq P$, we let $X_i$ contain $C^i$ from the graph thread, and we let the $X_{P+1}$ contain $A_2$.

Note that $X_1$ must contain the first (size 2) clique of each of the selector threads, and $X_P$ must contain the last (size 2) clique of each of the selector threads, by domino-ness. Each of these cliques has a vertex in common with the anchors, and this vertex can appear in only two nodes. It must appear in some node with its clique, and this cannot be the same node as the one where it appears with the anchor.

Suppose the dominating set of $G$ is $\{v_{d_1}, v_{d_2}, \ldots, v_{d_k}\}$. We will align the first selector thread so that the $v_{d_1}$th clique of $\mathcal{S}_1$ in this thread appears in $X_{2n+1}$, the same node in which $C^{2n+1}$, the first vertex clique in the graph thread, appears. We will align the second selector thread so that the $v_{d_2}$th clique of $\mathcal{S}_1$ in this thread appears in $X_{2n+1}$, and so on. For each selector thread, we place each of the $S^i$ cliques, $1 \leq i \leq n^2$, one per node consecutively, but we "fold" the size 2 cliques at the start of each selector thread, by placing 2 of these at at time into a node (i.e. 3 vertices per node) as many times as necessary, so as to ensure that the $v_{d_i}$th clique of $\mathcal{S}_1$ in $i$th thread occurs in the same node as $C^{2n+1}$ from the graph thread. The size 2 cliques at the end of each selector thread are also folded to fit into the nodes remaining before $A_2$ is reached in $X_{P+1}$.

Exactly $(n-1)$ folds will be required altogether, for each selector thread. The folding of the size 2 cliques will not breach the width bound, since each fold contributes 3 to the bag size, over at most $k$ threads, and at most $k(k+3)$ is permitted i.e $k+3$ per thread. Allowing $(n-1)$ folds will ensure that any of $\{v_1, \ldots, v_n\}$ can be positioned correctly.

If we align the selector threads this way, then each node containing a vertex clique from the graph thread will also contain a clique from at least one selector thread that is of size only $(k+2)$. Let $C^{2n+j.n+1}$ be a vertex clique representing vertex $v_j$ from $G$. If $v_j$ is in the dominating set then one of the selector threads will be aligned so that the $j$th clique of $\mathcal{S}_j$ from that thread appears in $X_{2n+j.n+1}$, and this clique has size only $(k+2)$. If $v_j$ is a neighbour of some vertex $v_i$ in the dominating set then one of the selector threads will be aligned so that the $i$th clique of $\mathcal{S}_j$ from that thread appears in $X_{2n+j.n+1}$, and this clique has size only $(k+2)$.

The decomposition described here preserves domino-ness. $X_0$ and $X_{P+1}$ each contain $K$ vertices. Each interior node in the decomposition contains either a non-vertex clique in the graph thread along with at most $k(k+3)$ other vertices, or a vertex clique in the graph thread along with $k$ cliques of size $(k+3)$ or $(k+2)$, where at least one of these cliques must have size $(k+2)$. Hence, each interior node contains at most $K$ vertices

Thus, we have a domino path decomposition of width at most $K-1$, as required.

$\Rightarrow$ Suppose a domino path decomposition of $G'$ with width $K$, $PD(G')$, exists, then $G$ must have a dominating set of size $k$.

- Each of $A_1$ and $A_2$ must be the contained in an end node of $PD(G')$, since no threads can pass over these large cliques, and all threads have vertices in common with both of them. Let us assume that $A_1$ is contained in the first node, $X_0$.
- Only one clique from the graph thread can be contained in any node of $PD(G')$, since there is not enough room to admit more than one. Each clique of the graph thread must be contained in some node of $PD(G')$. By domino-ness, and a simple induction argument, we must have the same situation described in the first part of this proof. The decomposition $PD(G')$ must consist of $P+2$ nodes, $A_1$ is contained in the first node, $X_0$, $A_2$ is contained in the last node, $X_{P+1}$, and for $1 \leq i \leq P$, node $X_i$ contains $C^i$ from the graph thread.
- The first vertex clique in the graph thread must appear in a node containing a clique from $\mathcal{S}_1$ for each of the selector threads. The last vertex clique in the graph thread must appear in a node containing a clique from $\mathcal{S}_n$ for each of the the selector threads.

  The first vertex clique in the graph thread appears in node $X_{2n+1}$. The last vertex clique in the graph thread appears in node $X_{2n+n(n-1)+1}$. There are $2n+1$ nodes that occur before $X_{2n+1}$ in the decomposition and $2n+1$ nodes that occur after $X_{2n+n(n-1)+1}$ in the decomposition. There are only $2n$ vertices occurring in a selector thread before the first clique of $\mathcal{S}_1$ is encountered. These are connected in a path, and by domino-ness, this path cannot stretch over more than $2n$ nodes. Similarly, the path at the end of the selector thread cannot stretch over more than $2n$ nodes.

- Each node in the decomposition, apart from the first and the last, contains a clique from the graph thread and so can contain at most $k$ distinct $S^i$ cliques from the selector threads.

  Each clique from the graph thread contains at least $k^2(k+4)$ vertices and each $S^i$ clique contains at least $(k+2)$ vertices. In any node there is room for only $k(k+3)$ more vertices apart from the graph thread clique. $(k+1)$ distinct $S^i$ cliques will consist of at least $(k+1)(k+2) = k(k+3)+2$ vertices.

- The arguments given here, along with dominoness, force the following situation.

  Every node in the decomposition from $X_{2n+1}$, which contains the first vertex clique of the graph thread, to $X_{2n+n(n-1)+1}$, which contains the last vertex clique of the graph thread, must contain exactly $k$ $S^i$ cliques, one from each of the selector threads. These must appear in the order in which they occur in the threads.

- For the width bound to be maintained, each node containing a vertex clique $C^{2n+j.n+1}$ from the graph thread must contain at least one $S^i$ clique of size $(k+2)$. Thus, the $\mathcal{S}_1$ cliques that occur in node $X_{2n+1}$ must correspond to $k$ vertices that form a dominating set in $G$. □

## 5 Conclusions

The notion of persistence for a path decomposition is introduced. Whilst this natural parameter has a number of very interesting algorithmic implications, the present article establishes that recognition of graphs having bounded persistence pathwidth is parametrically hard.

## References


Arnborg, S., Corneil, D. G. & Proskurowski, A. (1987), Complexity of finding embeddings in a $k$-tree, *SIAM J. Alg. Disc. Meth.* 8, pp. 277–284.

Bodlaender, H.L. (1996), A linear time algorithm for finding tree decompositions of small treewidth, *SIAM J. Comput.* 25, pp. 1305–1317.

Bodlaender, H.L. (1996a), A partial $k$-arboretum of graphs with bounded treewidth, Technical Report UU-CS-1996-02, Department of Computer Science, Utrecht University, Utrecht.

Bodlaender, H.L. (1997), Treewdith: Algorithmic techniques and results, *in* Proceedings of 22nd MFCS, Springer-Verlag LNCS 1295, pp. 19–36.

Bodlaender, H. L. & Engelfreit, J. (1997), Domino Treewidth, *J. Algorithms* 24, pp. 94–127.

Bodlaender, H. L., Fellows, M. R. & Hallett, M. T. (1994), Beyond NP-completeness for problems of bounded width: Hardness for the W-hierarchy, *in* Proceedings of 26th Annual Symposium on Theory of Computing, ACM Press, New York, pp. 449–458.

Bodlaender, H. L. & Kloks, T. (1996), Efficient and constructive algorithms for the pathwidth and treewdith of graphs, *J. Algorithms* 21, pp. 358–402.

Downey, R. G. & Fellows, M. R. (1999), *Parameterized Complexity*, Springer-Verlag.

Downey, R. G. & McCartin, C. M. (2004), Online Problems, Pathwidth, and Persistence, *in* Proceedings of IWPEC 2004, Springer-Verlag LNCS 3162, pp. 13–24.

de Fluiter, B. (1997), Algorithms for Graphs of Small Treewidth, ISBN 90-393-1528-0.

Fellows, M. R. & Langston, M. A. (1989), An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterizations, *in* Proceedings of the 30th Annual Symposium on Foundations of Computer Science, IEEE Computer Science Press, Los Alamitos, California, pp. 520–525.

Lagergren, J. (1996), Efficient parallel algorithms for graphs of bounded treewidth, *J. Algorithms* 20, pp. 20–44.

Matousek, J. & Thomas, R. (1991), Algorithms for finding tree-decompositions of graphs, *J. Algorithms* 12, pp. 1–22.

McCartin, C. (2003), Contributions to Parameterized Complexity, Ph.D., Victoria University, Wellington.

Perkovic, L. & Reed, B. (2000), An Improved Algorithm for Finding Tree Decompositions of Small Width, *International Journal of Foundations of Computer Science* 11 (3), pp. 365–371.

Reed, B. (1992), Finding approximate separators and computing treewdith quickly, *in* Proceedings of the 24th Annual Symposium on Theory of Computing, ACM Press, New York, pp. 221–228.

Robertson, N. & Seymour, P. D. (1986), Graph minors II. Algorithmic aspects of tree-width, *J. Algorithms* 7, pp. 309–322.

Robertson, N. & Seymour, P. D. (1985), Graph minors - a survey, *in* I. Anderson ed, 'Surveys in Combinatorics', Cambridge Univ. Press, pp. 153–171.