



Constructing tree decompositions of graphs with bounded gonality

Hans L. Bodlaender¹ · Josse van Dobben de Bruyn² · Dion Gijswijt²  · Harry Smit³

Accepted: 26 May 2021 / Published online: 2 August 2021
© The Author(s) 2021

Abstract

In this paper, we give a constructive proof of the fact that the treewidth of a graph is at most its divisorial gonality. The proof gives a polynomial time algorithm to construct a tree decomposition of width at most k , when an effective divisor of degree k that reaches all vertices is given. We also give a similar result for two related notions: stable divisorial gonality and stable gonality.

Keywords Tree decomposition · Gonality · Chip firing

This research was initiated at the Sandpiles and Chip Firing Workshop, held November 25–26, 2019 at the Centre for Complex Systems Studies, Utrecht University. Josse van Dobben de Bruyn was supported by NWO Grant 613.009.127. A preliminary version of this paper appeared in Proceedings of the 26rd International Computing and Combinatorics Conference 2020 (Bodlaender et al. 2020).

✉ Dion Gijswijt
d.c.gijswijt@tudelft.nl

Hans L. Bodlaender
h.l.bodlaender@uu.nl

Josse van Dobben de Bruyn
j.vandobbendebruyn@tudelft.nl

Harry Smit
smit@mpim-bonn.mpg.de

- ¹ Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
- ² Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
- ³ Max Planck Institute for Mathematics, Vivatsgasse 7, 53111 Bonn, Germany

1 Introduction

In this paper, we investigate the relation between well-studied graph parameters: treewidth and divisorial gonality. In particular, we give a constructive proof that the treewidth of a graph is at most its divisorial gonality.

Treewidth is a graph parameter with a long history. Its first appearance was under the name of *dimension*, in 1972, by Bertele and Brioschi (1972). It was rediscovered several times since, under different names (see e.g. Bodlaender 1998). Robertson and Seymour introduced the notions of *treewidth* and *tree decompositions* in their fundamental work on graph minors; these notions became the dominant terminology.

The notion of divisorial gonality finds its origin in algebraic geometry. Baker and Norine (2007) developed a divisor theory on graphs in analogy with divisor theory on curves, proving a Riemann–Roch theorem for graphs. The graph analog of gonality for curves was introduced by Baker (2008). To distinguish it from other notions of gonality (which we discuss briefly in Sect. 5), we denote the version we study by *divisorial gonality*. Divisorial gonality can be described in terms of a chip firing game. A placement of k chips on the vertices of a graph (where vertices can have 0 or more chips) is called an *effective divisor* of *degree* k . Under certain rules (see Sect. 2), sets of vertices can *fire*, causing some of the chips to move to different vertices. The divisorial gonality of a graph is the minimum degree of an effective divisor such that for each vertex v , there is a firing sequence ending with a configuration with at least one chip at v .

The treewidth of a graph is never larger than its divisorial gonality.¹ A non-constructive proof of this fact was given by van Dobben de Bruyn and Gijswijt (2020). Their proof is based on the characterization of treewidth in terms of brambles, due to Seymour and Thomas (1993). In this paper, we give a constructive proof of the same fact. We formulate our proof in terms of a search game characterization of treewidth, but with small modifications, we can also obtain a corresponding tree decomposition. The proof also yields a polynomial time algorithm that, when given an effective divisor of degree k , constructs a search strategy with at most $k + 1$ searchers and a tree decomposition of width at most k of the input graph.

This paper is organized as follows. Some preliminaries are given in Sect. 2. In Sect. 3, we prove the main result with help of a characterization of treewidth in terms of a search game and discuss that we also can obtain a tree decomposition of width equal to the degree of a given effective divisor that reaches all vertices. An example is given in Sect. 4. In Sect. 5, we give constructive proofs that bound the treewidth of a graph in terms of two related other notions of gonality. Some conclusions are given in Sect. 6.

¹ Conversely, graphs of treewidth 2 can have arbitrarily high divisorial gonality, which can be seen by considering ‘chains of circuits’. See for instance Cools et al. (2012), Hendrey (2018).

2 Preliminaries

2.1 Graphs

In this paper, all graphs are assumed to be finite. We allow multiple edges, but no loops. Let $G = (V, E)$ be a graph. For disjoint $U, W \subseteq V$ we denote by $E(U, W)$ the set of edges with one end in U and one end in W , and use the shorthand $\delta(U) = E(U, V \setminus U)$. The *degree* of a vertex $v \in V$ is $\deg(v) = |\delta(\{v\})|$, and given $v \in U \subseteq V$ we denote by $\text{outdeg}_U(v) = |E(\{v\}, V \setminus U)|$ the number of edges from v to $V \setminus U$. By $N(U)$ we denote the set of vertices in $V \setminus U$ that have a neighbor in U . The *Laplacian* of G is the matrix $Q(G) \in \mathbb{R}^{V \times V}$ given by

$$Q_{uv} = \begin{cases} \deg(u) & \text{if } u = v, \\ -|E(\{u\}, \{v\})| & \text{otherwise.} \end{cases}$$

2.2 Divisors and gonality

Let $G = (V, E)$ be a connected graph with Laplacian matrix $Q = Q(G)$. A *divisor* on G is an integer vector $D \in \mathbb{Z}^V$. The *degree* of D is $\deg(D) = \sum_{v \in V} D(v)$. We say that a divisor D is *effective* if $D \geq 0$, i.e., $D(v) \geq 0$ for all $v \in V$. We will denote by $\text{supp}(D) = \{v \in V \mid D(v) \neq 0\}$ the *support* of D .

The divisorial gonality can be defined in a number of equivalent ways. Most intuitive is the definition in terms of a chip firing game. An effective divisor D can be viewed as a chip configuration with $D(v)$ chips on vertex v . If $U \subset V$ is such that $\text{outdeg}_U(v) \leq D(v)$ for every $v \in U$ (i.e., each vertex has at least as many chips as it has edges to vertices outside U), then we say that U *can be fired*. If this is the case, then *firing* U means that every vertex in U gives chips to each of its neighbors outside U , one chip for every edge connecting to that neighbor. The resulting chip configuration is the divisor $D' = D - Q\mathbf{1}_U$. The assumption $\text{outdeg}_U(v) \leq D(v)$ guarantees that the number of chips on each vertex remains nonnegative, i.e. that D' is effective. Now, the divisorial gonality of a graph is the minimum number k such that there is a starting configuration (divisor) with k chips, such that for each vertex $x \in V$ there is a sequence of sets we can fire such that x receives a chip.

We now give the more formal definition, that is needed in our proofs. Two divisors D and D' are *equivalent* (notation: $D \sim D'$) if $D' = D - Qx$ for some $x \in \mathbb{Z}^V$. Note that equivalent divisors have the same degree since $Q^T \mathbf{1} = 0$. If D and D' are equivalent, then, since the null space of Q consists of all scalar multiples of $\mathbf{1}$, $D' = D - Qx$ has a unique solution $x \in \mathbb{Z}^V$ that is nonnegative and has $x_v = 0$ for at least one vertex v . We denote this x by $\text{script}(D, D')$ and write $\text{dist}(D, D') = \max\{x_v \mid v \in V\}$. Note that if $t = \text{dist}(D, D')$, then $\text{script}(D', D) = t\mathbf{1} - x$ and thus $\text{dist}(D', D) = \text{dist}(D, D')$. If D, D', D'' are pairwise equivalent, then we have the triangle inequality $\text{dist}(D, D'') \leq \text{dist}(D, D') + \text{dist}(D', D'')$ as $\text{script}(D, D'') = \text{script}(D, D') + \text{script}(D', D'') - c\mathbf{1}$ for some nonnegative integer c .

Let D be a divisor. If D is equivalent to an effective divisor, then we define

$$\text{rank}(D) = \max\{k \in \mathbb{Z}_{\geq 0} \mid D - E \text{ is equivalent to an effective divisor}\}$$

for every effective divisor E of degree at most k .

If D is not equivalent to an effective divisor, we set $\text{rank}(D) = -1$. The *divisorial gonality* of a graph G is defined as

$$\text{dgon}(G) = \min\{\text{deg}(D) \mid \text{rank}(D) \geq 1\}.$$

In the remainder of the paper, we will only consider effective divisors. If we can go from D to D' by sequentially firing a number of subsets, then clearly $D \sim D'$. The converse is also true (part (i) of the next lemma) as was shown in van Dobben de Bruyn and Gijswijt (2020, Lemma 2.3).

Lemma 1 *Let D and D' be equivalent effective divisors.*

- (i) *There is a unique increasing chain $\emptyset \subsetneq U_1 \subseteq U_2 \subseteq \dots \subseteq U_t \subsetneq V$ of subsets on which we can fire in sequence to obtain D' from D . That is, setting $D_0 = D$ and $D_i = D_{i-1} - Q\mathbf{1}_{U_i}$ for $i = 1, \dots, t$ we have $D_t = D'$ and D_i is effective for all $i = 0, \dots, t$.*
- (ii) *We have $t = \text{dist}(D, D') \leq \text{deg}(D) \cdot |V|$.*

Proof For part (i), let $x = \text{script}(D, D')$ and let $t = \text{dist}(D, D') = \max\{x(v) \mid v \in V\}$. We let $U_1 \subseteq \dots \subseteq U_t$ be the level set decomposition of x , i.e. let

$$U_i = \{v \in V \mid x(v) \geq t + 1 - i\} \quad (i = 1, \dots, t).$$

Note $x = \sum_{i=1}^t \mathbf{1}_{U_i}$. To conclude the proof of the existence part of (i), it suffices to show that the divisors D_0, D_1, \dots, D_t are indeed effective. By assumption, this is true for $D_0 = D$ and $D_t = D'$. Consider any $v \in V$. If $v \notin U_t$, then $0 \leq D_0(v) \leq D_1(v) \leq \dots \leq D_t(v)$ since chips can only be added to v when firing a subset not containing v . Otherwise, let i be the smallest index for which $v \in U_i$. Then

$$0 \leq D_0(v) \leq D_1(v) \leq \dots \leq D_{i-1}(v)$$

and

$$D_{i-1}(v) \geq D_i(v) \geq D_{i+1}(v) \geq \dots \geq D_t(v) \geq 0.$$

Hence, $D_i(v) \geq 0$ for all $i = 0, \dots, t$.

Uniqueness in part (i) follows directly from the fact that $x = \text{script}(D, D')$ is the unique solution to $D' = D - Qx$ with the additional property that x is nonnegative with at least one entry equal to zero, together with the uniqueness of the level set decomposition of x .

For part (ii), we note that a set U can occur at most $\text{deg}(D)$ times in the chain $U_1 \subseteq \dots \subseteq U_t$, since each time we fire the set U at least one chip leaves U . It follows that $t \leq \text{deg}(D) \cdot |V|$. □

We see that the two definitions of divisorial gonality are equivalent. Lemma 1 shows that we can even require the sets of vertices that are fired to be increasing.

For a given vertex q , a divisor $D \geq 0$ is called q -reduced if there is no nonempty set $U \subseteq V \setminus \{q\}$ such that $D - Q\mathbf{1}_U \geq 0$.

Lemma 2 (Baker and Norine 2007, Proposition 3.1) *Let D be an effective divisor and let q be a vertex. There is a unique q -reduced divisor equivalent to D .*

A divisor D has positive rank if and only if for every $q \in V$ the q -reduced divisor has at least one chip on q . The ‘if’ part is clear. The ‘only if’ part follows from the following lemma.

Lemma 3 *Let D be an effective divisor, let $q \in V$ and let D_q be the q -reduced divisor equivalent to D . Then $D_q(q) \geq D(q)$. In particular, if D has positive rank, then $D_q(q) \geq 1$.*

Proof We may assume that $D \neq D_q$. By Lemma 1 we obtain D_q from D by firing on a chain of sets $U_1 \subseteq \dots \subseteq U_t$ and, conversely, we obtain D from D_q by firing on the complements of U_t, \dots, U_1 . Since D_q is q -reduced, it follows that q is in the complement of U_t , and hence $q \notin U_i$ for $i = 1, \dots, t$. It follows that $x = \text{script}(D, D_q)$ satisfies $x_q = 0$ and $D_q(q) \geq D(q)$. \square

Given an effective divisor D and a vertex q , Dhar’s algorithm (Dhar 1990) (see Algorithm 1) finds in polynomial time a nonempty subset $U \subseteq V \setminus \{q\}$ on which we can fire, or concludes that D is q -reduced.

Algorithm 1: Dhar’s burning algorithm

Input : Divisor $D \geq 0$ on G and vertex q .
Output: Nonempty subset $U \subseteq V(G) \setminus \{q\}$ s.t. $D - Q\mathbf{1}_U \geq 0$ or $U = \emptyset$ if none exists.
 $U := V \setminus \{q\}$;
while $\text{outdeg}_U(v) > D(v)$ for some $v \in U$ **do**
 $U := U \setminus \{v\}$
end
return U

Lemma 4 *Dhar’s algorithm is correct, and the output is the unique inclusionwise maximal subset $U \subseteq V \setminus \{q\}$ that can be fired.*

Proof The set returned by Algorithm 1 can be fired, as it satisfies the requirement $\text{outdeg}_U(v) \leq D(v)$ for every $v \in U$. To complete the proof it therefore suffices to show that U contains every subset $W \subseteq V \setminus \{q\}$ that can be fired.

Let $W \subseteq V \setminus \{q\}$ be any such subset. At the start of the algorithm $U = V \setminus \{q\}$ contains W . While $U \supseteq W$, we have $\text{outdeg}_U(v) \leq \text{outdeg}_W(v) \leq D(v)$ for any $v \in W$, so the algorithm never removes a vertex $v \in W$ from U . \square

Note: in particular, Lemma 4 shows that the output of Algorithm 1 does not depend on the order in which vertices are selected for removal.

If throughout the algorithm we keep for every vertex v the number $\text{outdeg}_U(v)$ and a list of vertices for which $\text{outdeg}_U(v) > D(v)$, then we need only $O(|E|)$ updates, and we can implement the algorithm to run in time $O(|E|)$.

Lemma 5 *Let D be an effective divisor on the graph $G = (V, E)$, let $q \in V$, and let D_q be the q -reduced divisor equivalent to D . Let U be the set returned by Dhar's algorithm when applied to D and q , and suppose that $U \neq \emptyset$. Let $D' = D - Q\mathbf{1}_U$. Then $\text{dist}(D', D_q) = \text{dist}(D, D_q) - 1$.*

Proof Let $x = \text{script}(D, D_q)$. Since D_q is q -reduced, we have $x_q = 0$. On the other hand, since $D \neq D_q$ (as we can fire on U), the number $t = \max\{x_v \mid v \in V\}$ is positive. Let $W = \{v \in V \mid x_v = t\}$. By Lemma 1, we can fire on W , so by Lemma 4 we have $W \subseteq U$.

Let $x' = \text{script}(D', D_q)$ and let $t' = \max\{x'_v \mid v \in V\}$. As D_q is q -reduced, we have $x'_q = 0$. Since there is a unique nonnegative $y \in \mathbb{Z}^V$ with $y_q = 0$ and $D_q = D - Qy$, and we have $D - Qx = D_q = (D - Q\mathbf{1}_U) - Qx'$, it follows that $x = x' + \mathbf{1}_U$. Since $U \supseteq W$, it follows that $x - \mathbf{1}_W \geq x'$, and hence $t - 1 \geq t'$. We find that $\text{dist}(D', D_q) \leq \text{dist}(D, D_q) - 1$. Since $\text{dist}(D, D') = 1$, equality follows by the triangle inequality. \square

Since $\text{dist}(D, D_q) \leq \deg(D) \cdot |V|$, we can find a q -reduced divisor equivalent to D using no more than $\deg(D) \cdot |V|$ applications of Dhar's algorithm.

2.3 Treewidth and tree decompositions

The notions of treewidth and tree decomposition were introduced by Robertson and Seymour (1986) in their fundamental work on graph minors.

Let $G = (V, E)$ be a graph, let $T = (I, F)$ be a tree, and let $X_i \subseteq V$ be a set of vertices (called *bags*) associated to i for every node $i \in I$. The pair $(T, (X_i)_{i \in I})$ is a *tree decomposition* of G if it satisfies the following conditions:

1. $\bigcup_{i \in I} X_i = V$;
2. for all $e = vw \in E$, there is an $i \in I$ with $v, w \in X_i$;
3. for all $v \in V$, the set of nodes $I_v = \{i \in I \mid v \in X_i\}$ is connected (it induces a subtree of T).

The *width* of the tree decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* of G is the minimum width of a tree decomposition of G . Note that the treewidth of a multigraph is equal to the treewidth of the underlying simple graph.

There are several notions that are equivalent to treewidth. We will use a notion that is based on a Cops and Robbers game, introduced by Seymour and Thomas (1993). Here, a number of searchers need to catch a fugitive. Searchers can move from a vertex in the graph to a 'helicopter', or from a helicopter to any vertex in the graph. Between moves of searchers, the fugitive can move with infinite speed in the graph, but may not move over or to vertices with a searcher. The fugitive is captured when a searcher moves to the vertex with the fugitive, and there is no other vertex without a searcher

that the fugitive can move to. The location of the fugitive is known to the searchers at all times. We say that k searchers can capture a fugitive in a graph G , if there is a strategy for k searchers on G that guarantees that the fugitive is captured. In the initial configuration, the fugitive can choose a vertex, and all searchers are in a helicopter. A search strategy is *monotone* if it is never possible for the fugitive to move to a vertex that had been unreachable before. In particular, in a monotone search strategy, there is never a path without searchers from the location of the fugitive to a vertex previously occupied by a searcher.

Theorem 1 (Seymour and Thomas 1993) *Let G be a graph and k a positive integer. The following statements are equivalent.*

1. *The treewidth of G is at most k .*
2. *$k + 1$ searchers can capture a fugitive in G .*
3. *$k + 1$ searchers can capture a fugitive in G with a monotone search strategy.*

3 Construction of a search strategy

In this section, we present a polynomial time algorithm that, given a positive rank effective divisor D of degree k as input, constructs a monotone search strategy with $k + 1$ searchers to capture the fugitive.

We start by providing a way to encode monotone search strategies. Let G be a graph. For $X \subseteq V(G)$, the vertex set of a component of $G - X$ is called an X -flap. A *position* is a pair (X, R) , where $X \subseteq V(G)$ and R is a union² of X -flaps (we allow $R = \emptyset$).

The set X represents the vertices occupied by searchers, and the fugitive can move freely within some X -flap contained in R (if $R = \emptyset$, then the fugitive has been captured). In a monotone search strategy, the fugitive will remain confined to R , so placing searchers on vertices other than R is of no use. Therefore, it suffices to consider three types of moves for the searchers: (a) remove searchers that are not necessary to confine the fugitive to R ; (b) add searchers to R ; (c) if R consists of more than one X -flap, restrict attention to the X -flap $R_i \subset R$ containing the fugitive. This leads us to the following definition.

Definition 1 *Let G be a graph and let k be a positive integer. A *monotone search strategy* (MSS) with k searchers for G is a directed tree $T = (\mathcal{P}, F)$ where \mathcal{P} is a set of positions with $|X| \leq k$ for every $(X, R) \in \mathcal{P}$, and the following hold:*

- (i) *The root of T is (\emptyset, V) .*
- (ii) *If (X, R) is a leaf of T , then $R = \emptyset$.*
- (iii) *Let (X, R) be a non-leaf of T . Then $R \neq \emptyset$ and there is a set $X' \subseteq X \cup R$ such that exactly one of the following applies:*
 - (a) *$X' \subset X$ and position (X', R) is the unique out-neighbor of (X, R) .*

² Here we deviate from the definition of position as stated in Seymour and Thomas (1993) in that we allow R to consist of zero X -flaps or more than one X -flap.

- (b) $X' \supset X$ and position (X', R') is the unique out-neighbor of (X, R) , where $R' = R \setminus X'$.
- (c) $X' = X$ and the out-neighbors of (X, R) are the positions $(X, R_1), \dots, (X, R_t)$ where $t \geq 2$ and R_1, \dots, R_t are the X -flaps contained in R .

If condition (ii) does not necessarily hold, we say that T is a *partial* MSS. Note that we do not consider the root node to be a leaf even if it has degree 1.

It is clear that if T is an MSS for k searchers, then, as the name suggests, k searchers can capture the fugitive, the fugitive can never reach a vertex that it could not reach before, and a searcher is never placed on a vertex from which a searcher was previously removed.

We should point out that Definition 1 is slightly different from existing (formal or informal) definitions of a monotone search strategy in the literature. Compared to Seymour and Thomas (1993), we also allow a position to consist of zero X -flaps or more than one X -flap. At the end of this section, we will show that an MSS with k searchers yields a tree decomposition of width at most $k - 1$ in a relatively straightforward fashion.

First we focus on constructing an MSS in polynomial time. For this we use the following lemmas.

Lemma 6 *Let G be a graph on n vertices and let T be a (partial) MSS with k searchers for G . Then T has no more than $n^2 + 1$ nodes.*

Proof For any position (X, R) , define $f(X, R) = |R|(|X| + |R|)$. For any leaf node (X, R) we have $f(X, R) = 0$. For any non-leaf node (X, R) , the value $f(X, R)$ is at least the sum of the values of its children plus the number of children. Indeed, in case (a) and (b) we have $f(X, R) \geq f(X', R') + 1$, and in case (c) we have $f(X, R) \geq f(X, R_1) + \dots + f(X, R_t) + t$ since

$$\begin{aligned}
 & f(X, R) - (f(X, R_1) + \dots + f(X, R_t)) \\
 &= \sum_{1 \leq i < j \leq t} 2|R_i| \cdot |R_j| \geq 2|R_1| \cdot (|R_2| + \dots + |R_t|) \geq 2(t - 1) \geq t.
 \end{aligned}$$

It follows that $f(X, R)$ is an upper bound on the number of descendants of (X, R) in T . Since every non-root node is a descendant of the root, it follows that the total number of nodes is at most $1 + f(\emptyset, V) = 1 + n^2$. □

Lemma 7 *Let R be an X -flap. Let D be a positive rank effective divisor such that $X \subseteq \text{supp}(D)$ and $R \cap \text{supp}(D) = \emptyset$. Then we can find in polynomial time an effective divisor $D' \sim D$ such that $X \subseteq \text{supp}(D')$, $R \cap \text{supp}(D') = \emptyset$, and such that from D' we can fire a subset U with $U \cap R = \emptyset$ and $U \cap X \neq \emptyset$.*

Proof Let $q \in R$. Since $D(q) = 0$ and D has positive rank, it follows by Lemma 3 that D is not q -reduced. Let U be the set found by Dhar’s algorithm (with respect to q). Since R is connected and U does not contain R , it follows that $U \cap R = \emptyset$ (otherwise $\text{outdeg}_U(r) \geq 1 > D(r)$ for some $r \in U \cap R$). If $U \cap X$ is nonempty, we

set $D' := D$ and we are done. Otherwise, we set $D := D - Q\mathbf{1}_U$. Then $X \subseteq \text{supp}(D)$, $R \cap \text{supp}(D) = \emptyset$ and we iterate. We must finish in no more than $\text{deg}(D) \cdot |V(G)|$ iterations by Lemma 1 and Lemma 5. Hence, we can find the required D' and U in time $|E(G)| \cdot |V(G)| \text{deg}(D)$. \square

Construction of a monotone search strategy Let G be a connected graph and let D be an effective divisor on G of positive rank. Let $k = \text{deg}(D)$. We will construct an MSS for $k + 1$ searchers on G . We do this by keeping a partial MSS, starting with only the root node (\emptyset, V) and an edge to the node $(X, V \setminus X)$, where $X = \text{supp}(D)$. Then, we iteratively grow T at the leaves (X, R) with $R \neq \emptyset$ until T is an MSS. At each step, we also keep, for every leaf (X, R) of T , an effective divisor $D' \sim D$ such that $X \subseteq \text{supp}(D')$ and $R \cap \text{supp}(D') = \emptyset$. We now describe the iterative procedure.

While T has a leaf (X, R) with $R \neq \emptyset$, let D' be the divisor associated to (X, R) and perform one of the following steps.

- I. If R consists of multiple X -flaps R_1, \dots, R_t , then we add nodes $(X, R_1), \dots, (X, R_t)$ as children of (X, R) and associate D' to each. Iterate.
- II. If $X' := N(R)$ is a strict subset of X , then add the node (X', R) as a child of (X, R) , associate D' to this node and iterate.
- III. The remaining case is that $N(R) = X$ and R is a single X -flap. By Lemma 7 we can find an effective divisor $D'' \sim D'$ such that $X \subseteq \text{supp}(D'')$, $R \cap \text{supp}(D'') = \emptyset$ and from D'' we can fire on a set U such that $U \cap R = \emptyset$, and $U \cap X \neq \emptyset$. We set $U \cap X = \{s_1, s_2, \dots, s_t\}$. That we can fire on U implies that

$$D''(s_i) \geq |N(s_i) \cap R| \quad \text{for } i = 1, \dots, t. \tag{1}$$

For $i = 1, \dots, t$ we define positions (X_i, R_i) and (X'_i, R_i) as follows:

$$X_i = X'_{i-1} \cup (N(s_i) \cap R), \quad R_i = R \setminus X_i, \quad \text{and} \quad X'_i = X_i \setminus \{s_i\},$$

where we set $X'_0 = X$. Using (1) and the fact that $X \subseteq \text{supp}(D'')$, we see that

$$\begin{aligned} |X'_i| &= |X \setminus \{s_1, \dots, s_i\}| + |(N(s_1) \cap R) \cup \dots \cup (N(s_i) \cap R)| \\ &\leq |X \setminus \{s_1, \dots, s_i\}| + |N(s_1) \cap R| + \dots + |N(s_i) \cap R| \\ &\leq \sum_{v \in X \setminus \{s_1, \dots, s_i\}} D''(v) + \sum_{v \in \{s_1, \dots, s_i\}} D''(v) \\ &= \sum_{v \in X} D''(v) \end{aligned}$$

for $i = 0, \dots, t$. Hence, $|X'_i| \leq k$ and $|X_i| \leq k + 1$ for every i .

Since every edge in $\delta(R)$ has at least one endpoint in every X'_i , it follows that indeed R_i is a union of X'_i -flaps (and of X_i -flaps). We add the path $(X, R) \rightarrow (X_1, R_1) \rightarrow (X'_1, R_1) \rightarrow \dots \rightarrow (X_t, R_t) \rightarrow (X'_t, R_t)$ to T (it may happen that $(X_i, R_i) = (X'_{i-1}, R_{i-1})$ in which case we leave out one of the two). We associate $D'' - Q\mathbf{1}_U$ to the leaf (X'_t, R_t) .

By Lemma 6, we are done in at most $|V(G)|^2$ steps. This completes the construction. *Construction of a tree decomposition.* By combining the construction described above with that of Lemma 8, we obtain Theorem 2 below. Note that so far only a non-constructive proof was known of the fact that the divisorial gonality of a graph is an upper bound for the treewidth (van Dobben de Bruyn and Gijswijt 2020).

Lemma 8 *Let $T' = (\mathcal{P}, F)$ be a monotone search strategy for k searchers in the connected graph G and let T be the undirected tree obtained by ignoring the orientation of edges in T' . Then $(T, \{X\}_{(X,R) \in \mathcal{P}})$ is a tree decomposition of G of width at most $k - 1$.*

Proof Let $v \in V$. We first show that $v \in X$ for some $(X, R) \in \mathcal{P}$. Let $\mathcal{P}' := \{(X, R) \in \mathcal{P} : v \in R\}$. Note that \mathcal{P}' contains the root node (\emptyset, V) . Let $(X, R) \in \mathcal{P}'$ have maximum distance from the root. Since $v \in R$, it follows from the definition of MSS that (X, R) has a child (X', R') with $v \in X' \cup R'$. Hence, by the maximality assumption, we have $v \in X'$.

Next, we show that the set of nodes $\{(X, R) \in \mathcal{P} \mid v \in X\}$ is a subtree of T . Equivalently, we must show that if node (X_2, R_2) lies on a path from (X_1, R_1) to (X_3, R_3) in T , then $X_1 \cap X_3 \subseteq X_2$. It suffices to check this in two cases: the case that (X_3, R_3) is a descendant of (X_1, R_1) in T' , and the case that (X_2, R_2) is the last common ancestor of (X_1, R_1) and (X_3, R_3) . In the first case, it is easy to see that $X_3 \subseteq X_2 \cup R_2$ and $R_2 \subseteq R_1$ hold. It follows that

$$X_1 \cap X_3 \subseteq X_1 \cap (X_2 \cup R_2) \subseteq X_1 \cap (X_2 \cup R_1) \subseteq X_2$$

since X_1 and R_1 are disjoint. In the second case, node (X_2, R_2) has more than one out-neighbor, so its out-neighbors are positions (X_2, R) , where R runs over the X_2 -flaps contained in R_2 . It follows that $X_1 \subseteq X_2 \cup R'$ and $X_3 \subseteq X_2 \cup R''$ for distinct X_2 -flaps R' and R'' . Hence, $X_1 \cap X_3 \subseteq X_2$.

To complete the proof, it suffices to show that for every edge $\{u, v\} \in E(G)$ there is some node (X, R) of T with $u, v \in X$. Suppose for contradiction that this is not the case for edge $\{u, v\}$.

We first show that there is a node (X, R) such that $u \in X$ and $v \in R$ (or vice versa). To this end, consider the nodes (X, R) of T with $u, v \in R$ (e.g. the root node), and take such a node that has maximum distance from the root. This node cannot be a leaf since $R \ni v$ is non-empty. Since u and v belong to the same X -flap, it follows by the maximality assumption that (X, R) has a child (X', R') with $u \in X'$ and $v \in R'$ (or vice versa).

Now consider all nodes (X, R) with $u \in X$ and $v \in R$ and take such a node for which the distance to the root is maximised. This node cannot be a leaf (since $R \ni v$ is non-empty). Consider a child (X', R') of (X, R) . If we are in case (iii)(a) then $v \in R'$ and we must have $u \in X'$ since otherwise R' is not a union of X' -flaps as $\{u, v\}$ is an edge. This contradicts the maximality assumption. If we are in case (iii)(b), then $u \in X'$ and $v \in R'$ contradicting the maximality assumption. If we are in case (iii)(c), we may assume that R' is the X -flap containing v and again this contradicts the maximality assumption. □

Fig. 1 An example graph G . The divisor $D = 3a$ has positive rank. (It is not optimal, as $\text{dgon}(G) = 2$, since the divisor $b + f$ also has positive rank)

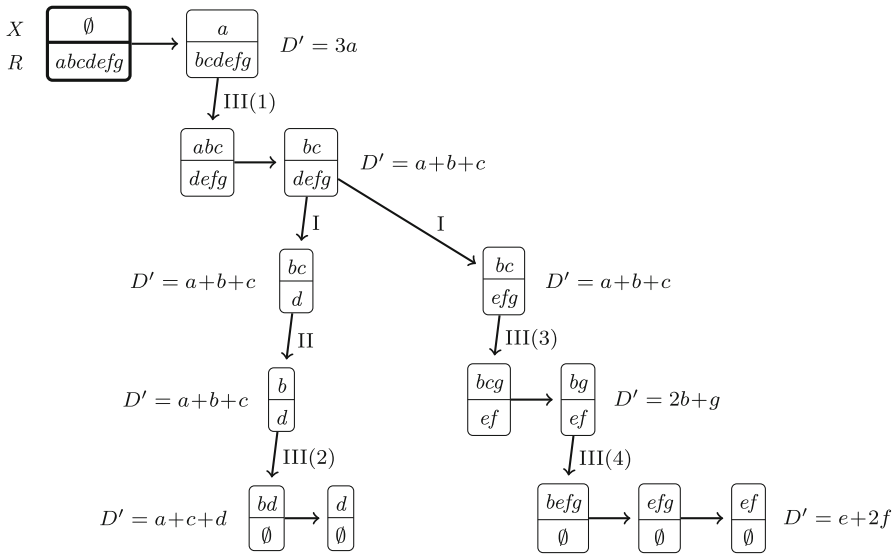
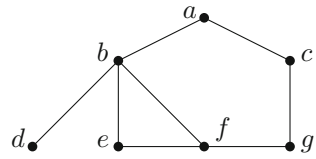


Fig. 2 The monotone search strategy obtained from G with divisor $D = 3a$. Each node shows the corresponding pair (X, R) with the root being $(\emptyset, \{a, b, c, d, e, f, g\})$. The labels I–III refer to the steps in the construction

Theorem 2 *There is a polynomial time algorithm that, when given a graph G and an effective divisor of degree k , finds a tree decomposition of G of width at most k .*

4 An example

We apply the constructions of the previous section to a relatively small example. Let $G = (V, E)$ be the graph depicted in Fig. 1 and let³ $D = 3a$ be the divisor on G that has value 3 on vertex a and value 0 elsewhere.

If we follow the construction of Sect. 3, we will end up with the monotone search strategy found in Fig. 2. Recall that every node of the search tree is a pair (X, R) of subsets of V such that R is a union of X -flaps in G . For every node, the sets X and R are indicated in the figure. The three ways of growing the tree (steps I, II, III of the construction) at a node with $R \neq \emptyset$ are indicated by downward arrows. Steps of type I are the only steps that involve branching of the tree. Steps of type III are the most involved: a path of nodes is added to the tree (depicted horizontally) and the divisor

³ For compactness of notation, we write the divisors as a formal sum. For instance, if divisor D' has 2 chips on b and 1 chip on g , we write $D' = 2b + g$.

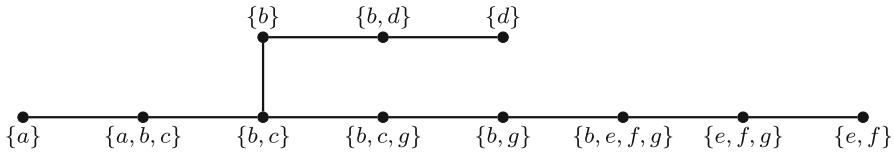


Fig. 3 Tree decomposition of G derived from the MSS

D' changes. For reference, the four steps of type III are labelled (1)–(4). Below, we will elaborate on the construction.

Root The initial partial MSS consists of a root $(X, R) = (\emptyset, V)$ connected to a leaf node $(\text{supp}(D), V \setminus \text{supp}(D)) = (\{a\}, \{b, c, d, e, f, g\})$. The divisor associated to the leaf node is $D' = 3a$.

Step III(1) For the leaf node $(X, R) = (\{a\}, \{b, c, d, e, f, g\})$, the set R is a single X -flap and $N(R) = X$, so we apply step III. Dhar’s algorithm applied to divisor $D' = 3a$ (and an arbitrary vertex in R) gives the set $U = \{a\}$ to fire on. We have $U \cap X = \{s_1\}$ with $s_1 = a$. We obtain

$$X_1 = X \cup (N(s_1) \cap R) = \{a, b, c\}, \quad R_1 = R \setminus X_1 = \{d, e, f, g\}, \quad X'_1 = \{b, c\}.$$

The path $(X_1, R_1) \rightarrow (X'_1, R_1)$ is attached to leaf node (X, R) . The divisor $D' - Q\mathbf{1}_U = a + b + c$ is associated to the new leaf node (X'_1, R_1) .

Step I For the leaf node $(X, R) = (\{b, c\}, \{defg\})$, the set R is the union of two X -flaps: $R = \{d\} \cup \{e, f, g\}$. Hence, we apply step I to obtain two new leaf nodes $(\{b, c\}, \{d\})$ and $(\{b, c\}, \{e, f, g\})$.

Step II In the left-hand leaf node $(X, R) = (\{b, c\}, \{d\})$ we have $N(R) = \{b\} \subset X$, so we apply step II and add a new leaf node $(\{b\}, \{d\})$.

Step III(2–4) The remaining steps in the construction of the MSS are of type III. We summarise the details below.

- (2) Divisor D' is equal to $a + b + c$. Applying Dhar’s algorithm to D' and the vertex d , we obtain the set $U = \{a, b, c, e, f, g\}$. Firing on U , we obtain the new divisor $a + c + d$.
- (3) Divisor D' is equal to $a + b + c$. Applying Dhar’s algorithm to D' and any of the vertices in $R = \{e, f, g\}$ we obtain the set $U = \{a, c\}$. Firing on U , we obtain the new divisor $2b + g$.
- (4) Divisor D' is equal to $2b + g$. Applying Dhar’s algorithm to D' and the vertex e (or equivalently f) we obtain the set $U = \{a, b, c, d, g\}$. Firing on U , we obtain the new divisor $e + 2f$.

From the search tree, we obtain a tree-decomposition of width $\text{deg}(D) = 3$ by labeling each node (X, R) by the set X and ignoring arc directions. Removing nodes with label \emptyset and contracting edges of the tree between nodes with equal labels, we obtain the following tree decomposition (Fig. 3).

5 Other notions of gonality

5.1 Stable divisorial gonality

The *stable divisorial gonality* of a graph G is the minimum of $\text{dgon}(H)$ over all subdivisions H of G (i.e., graphs H that can be obtained by subdividing zero or more edges of G). The bound for divisorial gonality can easily be transferred to one for stable divisorial gonality. If G is simple, then the treewidth of G equals the treewidth of any of its subdivisions. (This is well known.) If G is not simple, then either the treewidth of G equals the treewidth of all its subdivisions, or G is obtained by adding parallel edges to a forest (i.e., the treewidth of G equals 1), and we subdivide at least one of these parallel edges (thus creating a graph with a cycle; the treewidth will be equal to 2 in this case.) In the latter case, the (stable) divisorial gonality will be at least two. Thus, we have the following easy corollary.

Corollary 1 *The treewidth of a graph G is at most the stable divisorial gonality of G .*

Standard treewidth techniques allow us to transform a tree decomposition of a subdivision of G into a tree decomposition of G of the same width. (For each subdivided edge $\{v, w\}$ replace each occurrence of a vertex representing a subdivision of this edge by v in each bag.)

5.2 Stable gonality

Related to (stable) divisorial gonality is the notion of *stable gonality*; see Cornelissen et al. (2015). This notion is defined using finite harmonic morphisms to trees.

Let G and H be undirected nonempty graphs. We allow G and H to have parallel edges but not loops. A graph homomorphism from G to H is a map $f : V(G) \cup E(G) \rightarrow V(H) \cup E(H)$ that maps vertices to vertices, edges to edges, and preserves incidences of vertices and edges:

- $f(V(G)) \subseteq V(H)$,
- if e is an edge between vertices u and v , then $f(e)$ is an edge between $f(u)$ and $f(v)$.

A *finite morphism* from G to H (notation: $f : G \rightarrow H$) is graph homomorphism f from G to H together with an *index function* $r_f : E(G) \rightarrow \mathbb{Z}_{>0}$. A finite morphism $f : G \rightarrow H$ with index function r_f is *harmonic* if for every vertex $v \in V(G)$, there is a constant $m_f(v)$ such that for each edge $e \in E(H)$ incident to $f(v)$, we have

$$\sum_{e' \text{ incident to } v; f(e')=e} r_f(e') = m_f(v)$$

If H is connected and $|E(G)| \geq 1$, then there is a positive integer $\text{deg}(f)$, the *degree* of f , such that for all vertices $w \in V(H)$ and edges $e \in E(H)$, we have

$$\text{deg}(f) = \sum_{v \in V(G); f(v)=w} m_f(v) = \sum_{e' \in E(G); f(e')=e} r_f(e');$$

Fig. 4 A graph G with tree H and harmonic morphism f given by projecting the edges and vertices of G downward to H . The degree of f equals 3. The value of r_f resp. m_f is displayed at every edge resp. vertex of G

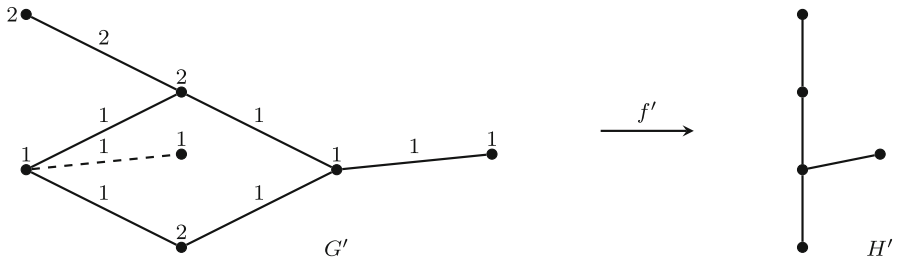
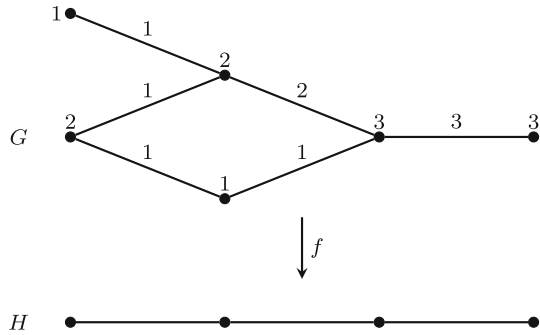


Fig. 5 A refinement G' of G and a harmonic morphism f' to a tree H' given by projecting the edges and vertices of G' to the right. The degree of f' equals 2. The values of $r_{f'}$ resp. $m_{f'}$ are displayed at every edge resp. vertex of G'

see Urakawa (2000, Lemma 2.12) and Baker and Norine (2009, Lemma 2.3). In particular, f is surjective in this case.

A *refinement* of a graph G is a graph G' that can be obtained from G by zero or more of the following two operations: subdivide an edge; add a leaf (i.e., add one new vertex and an edge from that vertex to an existing vertex).

The *stable gonality* of a connected non-empty graph G is the minimum degree of a finite harmonic morphism of a refinement of G to a tree.

Example 1 We look at an example that illustrates the above definition and shows how taking refinements might lead to harmonic morphisms of low degree. Consider Fig. 4. Here the map $f : G \rightarrow H$ is given by projecting each edge and vertex downward. The value of the index function r_f is given at every edge and the value of m_f is given at every vertex. One can check that for every edge in H , the sum of the indices of the pre-images equals 3, and that the sum of the values of m_f also equals 3. The degree of f equals 3.

By refining G one obtains a harmonic morphism of degree 2, see Fig. 5. The map f' is given by projecting to the right, and again the values of the index function are given at every edge. The two horizontal edges in G are mapped to the horizontal edge in H' . This map has degree 2. As G is not a tree, the stable gonality of G is greater than 1, hence we conclude that the stable gonality of G is 2.

Lemma 9 *Let G be an undirected connected graph without loops and at least one edge. Given a tree T and a finite harmonic morphism $f : G \rightarrow T$ of degree k , a tree decomposition of G of width at most k can be constructed in $O(k^2|V(G)|)$ time.*

Before proving the lemma, we make some simple observations. Recall that indices $r_f(e)$ are positive integers. We thus have for each edge $e \in E(T)$:

$$|\{e' \in E(G) \mid f(e') = e\}| \leq \sum_{e' \in E(G); f(e')=e} r_f(e') = \deg(f).$$

Since G is connected and has at least one edge, it follows that $m_f(v) \geq 1$ for every $v \in V(G)$. Hence, for each vertex $i \in V(T)$:

$$|\{v \in V(G) \mid f(v) = i\}| \leq \sum_{v \in V(G); f(v)=i} m_f(v) = \deg(f).$$

Proof (of Lemma 9) We build a tree decomposition of G in the following way. For each edge $e \in E(T)$, we have that $|\{e' \in E(G) \mid f(e') = e\}| \leq k$. Call this number $\ell(e)$. We subdivide e precisely $\ell(e)$ times; that is, we add $\ell(e)$ new vertices on this edge. Let T' be the tree that is obtained in this way.

To the nodes i of T' , we associate sets X_i in the following way. If i is a node of T (i.e., not a node resulting from the subdivisions), then $X_i = f^{-1}(i)$, i.e., all vertices mapped by the morphism to i . By the observation above, we have that $|X_i| \leq \deg(f) = k$.

Consider an edge $\{i, j\}$ in T . Write $k' = \ell(\{i, j\})$. Recall that there are $k' \leq k$ edges of G that are mapped to $\{i, j\}$. Suppose these are $e_1 = \{v_1, w_1\}, \dots, e_{k'} = \{v_{k'}, w_{k'}\}$ with $f(v_1) = f(v_2) = \dots = f(v_{k'}) = i$ and $f(w_1) = f(w_2) = \dots = f(w_{k'}) = j$. Let $i_1, i_2, \dots, i_{k'}$ be the subdivision nodes of the edge $\{i, j\}$, with i_1 incident to i and $i_{k'}$ incident to j . Set $X_{i_r} = \{v_s \mid r \leq s \leq k'\} \cup \{w_t \mid 1 \leq t \leq r\}$ for $r \in \{1, \dots, k'\}$. The construction is illustrated in Fig. 6. We claim that this yields a tree decomposition of G of width at most k .

For all edges $\{v, w\} \in E(G)$, we have $\{f(v), f(w)\} \in E(T)$. Suppose without loss of generality that $f(v)$ has the role of i , $f(w)$ the role of j , $v = v_r$ and $w = w_r$ in the construction above. Then $v, w \in X_{i_r}$.

Finally, for all $v \in V$, the sets X_i to which v belongs are the following: v is in $X_{f(v)}$, and for each edge incident to $f(v) \in T$, v is in zero or more successive bags of subdivision nodes of this edge, with the first one (if existing), incident to $f(v)$. Thus, the bags to which v belongs form a connected subtree.

The first condition of tree decompositions follows from the second and the fact that G is connected. Hence T' , with bags as defined above, yields a tree decomposition of G .

Finally, note that each set X_i is of size at most $k + 1$: vertices in T have a bag of size k and subdivision vertices have a bag of size $k' + 1 \leq k + 1$. So, we have a tree decomposition of G of width at most k .

It is straightforward to see that the construction in the proof can be carried out in $O(k^2|V(G)|)$ time. (Use that $|V(T)| \leq |V(G)|$, since f is surjective.) \square

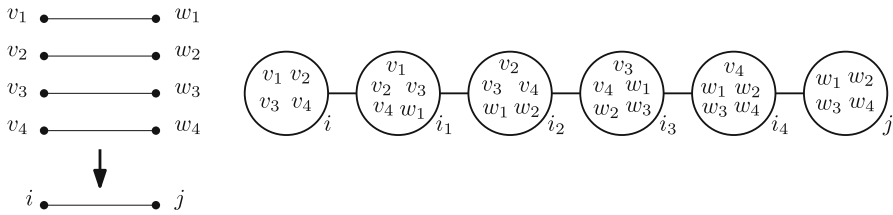


Fig. 6 Example of a step in the proof of Lemma 9. Here $k' = 4$. Left: four edges are mapped to the edge $\{i, j\}$ by the finite harmonic morphism. Right: the corresponding bags in the tree decomposition

Theorem 3 *Let G be an undirected connected graph without loops. Suppose that G has stable gonality k . Then G has treewidth at most k . Given a refinement G' of G and a finite harmonic morphism $f : G' \rightarrow T$ of degree k , a tree decomposition of G of width at most k can be constructed in $O(k^2|V(G')|)$ time.*

Proof The degenerate case that G has no edges must be handled separately; here we have that the treewidth of G is 0, which is equal to its stable gonality.

Suppose G has at least one edge. By Lemma 9, we obtain a tree-decomposition of G' of width k in $O(k^2|V(G')|)$ time. Standard treewidth techniques allow us to transform a tree decomposition of a refinement of G into a tree decomposition of G of the same or smaller width. Added leaves can just be removed from all bags where they occur. For each subdivided edge $\{v, w\}$, replace each occurrence of a vertex representing a subdivision of this edge by v in each bag. □

5.3 Comparison of the tree decompositions

Divisorial gonality and stable gonality are unbounded in one another (Gijswijt et al. 2020, Sect. 5). In particular, the existence of a harmonic morphism $G \rightarrow T$ of degree k does not imply the existence of a positive rank divisor of degree k on G and vice versa. Thus one cannot expect the minimal width of the tree decompositions of a graph G obtained through harmonic morphisms to be equal to the minimal width obtained through positive rank divisors, let alone that the tree decompositions themselves are equal.

However, we can relate the two constructions as follows. Given a graph G and a harmonic morphism $f : G \rightarrow T$ of degree k one can obtain a positive rank divisor on a graph G' with $V(G) = V(G')$: replace every edge e of G by $r_f(e)$ parallel edges to obtain G' . Choose any vertex $t \in T$ and let D_t be the divisor with $m_f(v)$ chips on every vertex $v \in f^{-1}(t)$ and no chips on the other vertices. Then D_t is a positive rank divisor on G' of degree k . Indeed, given a neighbor $t' \in T$ of t , let $T_{t,t'}$ be the vertex set of the connected component of $T - \{t, t'\}$ that contains t . We can fire the set $f^{-1}(T_{t,t'})$, which moves all chips from vertices in $f^{-1}(t)$ to vertices in $f^{-1}(t')$. As $v \in f^{-1}(t)$ has exactly $m_f(v)$ edges to vertices in $f^{-1}(t')$ and $v' \in f^{-1}(t')$ has exactly $m_f(v')$ edges to vertices in $f^{-1}(t)$, we find that the resulting divisor has $m_f(v')$ chips on every vertex $v' \in f^{-1}(t')$ and no chips on other vertices. That is, it is precisely the divisor $D_{t'}$. Repeating this process shows that D_t is a positive rank divisor.

We can use the divisors D_t and firing sets $T_{t,t'}$ to construct a monotone search strategy as in Sect. 3. Indeed, suppose we are in Step III of the construction of a monotone search strategy, and $D' = D_t$ for some node $t \in T$. Let $r \in R \cap N(X)$. Then $t' = f(r)$ is a neighbor of t in T . Since R is connected, $f(R) = T_{t',t} = V(T) \setminus T_{t,t'}$. Hence, we can take $D'' = D_t$ and fire on $U = T_{t,t'}$.

In this way we obtain a decomposition that is basically the same as the tree decomposition obtained from the harmonic morphism f . Indeed, if t and t' are neighbors in T , we let $f^{-1}(t) = \{v_1, \dots, v_n\}$ and denote the neighbors of v_i in $f^{-1}(t')$ by $v'_{i,1}, \dots, v'_{i,a_i}$. Then the tree decomposition obtained from D_t will have a path of bags

$$\begin{aligned}
 f^{-1}(t) = \{v_1, \dots, v_n\} &\rightarrow \{v_1, \dots, v_n, v_{1,1}, \dots, v_{1,a_1}\} \\
 &\rightarrow \{v_2, \dots, v_n, v_{1,1}, \dots, v_{1,a_1}\} \\
 &\rightarrow \{v_2, \dots, v_n, v_{1,1}, \dots, v_{1,a_1}, v_{2,1}, \dots, v_{2,a_2}\} \\
 &\rightarrow \dots \\
 &\rightarrow \{v_{1,1}, \dots, v_{1,a_1}, \dots, v_{n,1}, \dots, v_{n,a_n}\} = f^{-1}(t').
 \end{aligned}$$

Numbering the edges⁴ $\{v_i, v_{i,j}\} =: e_{a_1+\dots+a_{i-1}+j}$, the tree decomposition obtained from f will have a path of bags

$$\begin{aligned}
 f^{-1}(t) = \{v_1, \dots, v_n\} &\rightarrow \{v_1, \dots, v_n, v_{1,1}\} \\
 &\rightarrow \dots \\
 &\rightarrow \{v_1, \dots, v_n, v_{1,1}, \dots, v_{1,a_1}\} \\
 &\rightarrow \{v_2, \dots, v_n, v_{1,1}, \dots, v_{1,a_1}, v_{2,1}\} \\
 &\rightarrow \dots \\
 &\rightarrow \{v_2, \dots, v_n, v_{1,1}, \dots, v_{1,a_1}, v_{2,1}, \dots, v_{2,a_2}\} \\
 &\rightarrow \dots \\
 &\rightarrow \{v_{1,1}, \dots, v_{1,a_1}, \dots, v_{n,1}, \dots, v_{n,a_n}\} = f^{-1}(t').
 \end{aligned}$$

The second sequence of bags can be easily contracted to obtain the first. Hence, a tree decomposition constructed from a harmonic morphism can also be obtained from a positive rank divisor on an associated graph.

It is unknown to the authors whether or not any tree decomposition obtained from a positive rank divisor can be obtained from a finite harmonic morphism on some associated graph, as it is unclear how to obtain a harmonic morphism from a positive rank divisor.

⁴ For simplicity, we assume here that G has no parallel edges. If it does, the path will have additional, repeated, bags.

6 Conclusions

In this paper, we gave a constructive proof that the divisorial gonality of a graph is an upper bound for its treewidth. Before this, only a non-constructive proof was known (van Dobben de Bruyn and Gijswijt 2020). There are several graph theoretic notions with the property that they give an upper bound for treewidth; see e.g. Bodlaender (1998) for an older overview. One consequence of our result is that problems which can be solved efficiently (e.g., in linear time) on graphs of bounded treewidth (e.g., when they can be formulated in Monadic Second Order Logic (see Courcelle (1990) or Cygan et al. (2015, Chapter 7)) also can be solved efficiently on graphs of bounded divisorial gonality. It is a natural question to ask whether there are problems that are intractable even when the treewidth is small, but become tractable for small divisorial gonality; however, it appears hard to come with natural examples for this. The same applies to the notions stable divisorial gonality

An interesting open problem is to establish the parameterized complexity of divisorial gonality, and the other gonality notions. It is known that the DIVISORIAL GONALITY problem (i.e., given a graph G , and an integer k , decide if $\text{dgon}(G) \leq k$) is in XP (Baker and Shokrieh 2013; Dobben de Bruyn 2012), but it is unknown if the problem is in FPT, i.e., if there is an algorithm with running time of the form $O(f(k) \cdot n^c)$ for constant c . Likewise, are STABLE DIVISORIAL GONALITY or STABLE GONALITY in XP or even in FPT? (See e.g. Cygan et al. (2015)) for an introduction to parameterized complexity.)

Acknowledgements We thank Gunther Cornelissen, Bart Jansen, Erik Jan van Leeuwen, Marieke van der Wegen, and Tom van der Zanden for helpful discussions.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Baker M (2008) Specialization of linear systems from curves to graphs. *Algebra Number Theory* 2(6):613–653
- Baker M, Norine S (2007) Riemann–Roch and Abel–Jacobi theory on a finite graph. *Adv Math* 215(2):766–788
- Baker M, Norine S (2009) Harmonic morphisms and hyperelliptic graphs. *Int Math Res Not* 15:2914–2955
- Baker M, Shokrieh F (2013) Chip-firing games, potential theory on graphs, and spanning trees. *J Comb Theory Ser B* 120:164–182
- Bertele U, Brioschi F (1972) *Nonserial dynamic programming*. Academic Press, New York
- Bodlaender HL (1998) A partial k -arboretum of graphs with bounded treewidth. *Theor Comput Sci* 209(1–2):1–45. [https://doi.org/10.1016/S0304-3975\(97\)00228-4](https://doi.org/10.1016/S0304-3975(97)00228-4)
- Bodlaender HL, van Dobben de Bruyn J, Gijswijt D, Smit H (2020) Constructing tree decompositions of graphs with bounded gonality. In: Kim D, Uma RN, Cai Z, Lee DH (eds) *Computing and combinatorics*. Springer, Cham, pp 384–396

- Cools F, Draisma J, Payne S, Robeva E (2012) A tropical proof of the Brill–Noether theorem. *Adv Math* 230(2):759–776
- Cornelissen G, Kato F, Kool J (2015) A combinatorial Li–Yau inequality and rational points on curves. *Mathematische Annalen* 361(1–2):211–258. <https://doi.org/10.1007/s00208-014-1067-x>
- Courcelle B (1990) The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform Comput* 85(1):12–75. [https://doi.org/10.1016/0890-5401\(90\)90043-H](https://doi.org/10.1016/0890-5401(90)90043-H)
- Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) *Parameterized algorithms*. Springer, Berlin. <https://doi.org/10.1007/978-3-319-21275-3>
- Dhar D (1990) Self-organized critical state of sandpile automaton models. *Phys Rev Lett* 64(14):1613
- Dobben de Bruyn JV (2012) Reduced divisors and gonality in finite graphs. Bachelor thesis, Leiden University. <https://www.universiteitleiden.nl/binaries/content/assets/science/mi/scripties/bachvandobbendebruyn.pdf>
- van Dobben de Bruyn J, Gijswijt D (2020) Treewidth is a lower bound on graph gonality. *Algebr Combin* 3(4):941–953. <https://doi.org/10.5802/alco.124>
- Gijswijt D, Smit H, van der Wegen M (2020) Computing graph gonality is hard. *Discrete Appl Math* 287:134–149
- Hendrey K (2018) Sparse graphs of high gonality. *SIAM J Discrete Math* 32(2):1400–1407
- Robertson N, Seymour PD (1986) Graph minors. II. Algorithmic aspects of tree-width. *J Algorithms* 7:309–322
- Seymour PD, Thomas R (1993) Graph searching and a minimax theorem for tree-width. *J Combin Theory Ser B* 58:239–257
- Urakawa H (2000) A discrete analogue of the harmonic morphism and Green kernel comparison theorems. *Glasgow Math J* 42(3):319–334

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.