Constructing tree decompositions of graphs with bounded gonality

Bodlaender, Hans L.; van Dobben de Bruyn, Josse; Gijswijt, Dion; Smit, Harry

DOI
10.1007/978-3-030-58150-3_31

Publication date
2020

Document Version
Accepted author manuscript

Published in
Computing and Combinatorics

Citation (APA)
Bodlaender, H. L., van Dobben de Bruyn, J., Gijswijt, D., & Smit, H. (2020). Constructing tree decompositions of graphs with bounded gonality. In D. Kim, R. N. Uma, Z. Cai, & D. H. Lee (Eds.), *Computing and Combinatorics : 26th International Conference, COCOON 2020, Proceedings* (pp. 384-396). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 12273). Springer. https://doi.org/10.1007/978-3-030-58150-3_31

Constructing Tree Decompositions of Graphs with Bounded Gonality*

Hans L. Bodlaender[1], Josse van Dobben de Bruyn[2]**,
Dion Gijswijt[2], and Harry Smit[3]

[1] Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 YB Utrecht, the Netherlands. `h.l.bodlaender@uu.nl`
[2] Delft Institute of Applied Mathematics, Delft University of Technology, the Netherlands. `{j.vandobbendebruyn,d.c.gijswijt}@tudelft.nl`
[3] Department of Mathematics, Utrecht University, P.O. Box 80.010, 3508 TA Utrecht, the Netherlands. `h.j.smit@uu.nl`

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.

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 Briochi [4]. It was rediscovered several times since, under different names (see e.g. [5]). 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 [2] 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 [1]. To distinguish it from other notions of gonality (which we discuss briefly in Section 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

---

* 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. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-58150-3_31.

** Supported by NWO grant 613.009.127.

vertices can have 0 or more chips) is called an *effective divisor* of *degree k*. Under certain rules (see Section 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[4]. A non-constructive proof of this fact was given by van Dobben de Bruyn and Gijswijt [10]. Their proof is based on the characterization of treewidth in terms of brambles, due to Seymour and Thomas [13]. 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 Section 2. In Section 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 Section 4. In Section 5, we give constructive proofs that bound the treewidth of a graph in terms of two related other notions of gonality.

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 $\mathrm{outdeg}_U(v) = |E(\{v\}, V \setminus U)|$ 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 $v \in V$.

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

---

[4] Conversely, graphs of treewidth 2 can have arbitrarily high divisorial gonality, which can be seen by considering ‘chains of circuits’. See for instance [7,11].

can be viewed as a chip configuration with $D(v)$ chips on vertex v. If $U \subset V$ is such that $\mathrm{outdeg}_U(v) \le D(v)$ 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 $\mathrm{outdeg}_U(v) \le D(v)$ 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 $\mathrm{script}(D, D')$ and write $\mathrm{dist}(D, D') = \max\{x_v : v \in V\}$. Note that if $t = \mathrm{dist}(D, D')$, then $\mathrm{script}(D', D) = t\mathbf{1} - x$ and thus $\mathrm{dist}(D', D) = \mathrm{dist}(D, D')$. If D, D', D'' are pairwise equivalent, then we have the triangle inequality $\mathrm{dist}(D, D'') \le \mathrm{dist}(D, D') + \mathrm{dist}(D', D'')$ as $\mathrm{script}(D, D'') = \mathrm{script}(D, D') + \mathrm{script}(D', D'') - c\mathbf{1}$ for some nonnegative integer c.

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

$$\begin{aligned}\mathrm{rank}(D) = \max\{k \in \mathbb{Z}_{\ge 0} : {} & D - E \text{ is equivalent to an effective divisor} \\ & \text{for every effective divisor } E \text{ of degree at most } k\}.\end{aligned}$$

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

$$\mathrm{dgon}(G) = \min\{\deg(D) : \mathrm{rank}(D) \ge 1\}.$$

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 [10, Lemma 1.3]. (The proof can also be found in [6].)

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

(i) *There is a unique increasing chain $\emptyset \subsetneq U_1 \subseteq U_2 \subseteq \cdots \subseteq U_t \subsetneq V$ 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 $i = 1, \ldots, t$ we have $D_t = D'$ and D_i is effective for all $i = 0, \ldots, t$.*

(ii) *We have $t = \mathrm{dist}(D, D') \le \deg(D) \cdot |V|$.*

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

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

Lemma 2 ([2, 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.*

Let D be an effective divisor and let D_q be the q-reduced divisor equivalent to D. Suppose that $D \neq D_q$. By Lemma 1 we obtain D_q from D by firing on a chain of sets $U_1 \subseteq \cdots \subseteq U_t$ and, conversely, we obtain D from D_q by firing on the complements of $U_t, \ldots, U_1$. Since D_q is q-reduced, it follows that q is in the complement of U_t, and hence $q \notin U_1$. It follows that $x = \text{script}(D, D_q)$ satisfies $x_q = 0$ and $D_q(q) \geq D(q)$. In particular, a divisor D has positive rank if and only if for every $q \in V$ the q-reduced divisor equivalent to D has at least one chip on vertex q.

Given an effective divisor D and a vertex q, Dhar's algorithm [9] 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.
$U \leftarrow V \setminus \{q\}$;
while $\text{outdeg}_U(v) > D(v)$ for some $v \in U$ do
    $U \leftarrow U \setminus \{v\}$
end
return $U$
```

Lemma 3. *Dhar's algorithm is correct, and the output is the unique inclusion-wise 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. □

Note: in particular, Lemma 3 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 4. *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 $t = \max\{x_v : v \in V\}$ is positive. Let $W = \{v \in V : x_v = t\}$. By Lemma 1, we can fire on W, so by Lemma 3 we have $W \subseteq U$.

Let $x' = \text{script}(D', D_q)$ and let $t' = \max\{x'_v : v \in V\}$. 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 $\text{dist}(D, D_q) \leq \deg(D) \cdot |V(G)|$, 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 [12] 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 a 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 [13]. 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 [13]). *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 an 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 union[5] 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).*
  - (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.

---

[5] Here we deviate from the definition of position as stated in [13] in that we allow R to consist of zero X-flaps or more than one X-flap.

Lemma 5. *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 $f(X,R)\geq 0$. 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 $f(X,R)\geq f(X,R_1)+\cdots+f(X,R_k)+k$ as can be easily verified. 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 $1+f(\emptyset,V)=1+n^2$. □

In the construction of an MSS we will use the following lemma.

Lemma 6. *Let R be an X-flap. Let D be a positive rank effective divisor such that $X\subseteq \operatorname{supp}(D)$ and $R\cap\operatorname{supp}(D)=\emptyset$. Then we can find in polynomial time an effective divisor $D'\sim D$ such that $X\subseteq\operatorname{supp}(D')$, $R\cap\operatorname{supp}(D')=\emptyset$, and such that from D' we can fire a subset U with $U\cap R=\emptyset$ and $U\cap X\neq\emptyset$.*

Proof. Let $q\in R$. Let U be the set found by Dhar's algorithm. Since R is connected and U does not contain R, it follows that $U\cap R=\emptyset$ (otherwise $\operatorname{outdeg}_U(r)\geq 1>D(r)$ for some $r\in U\cap R$). If $U\cap X$ is nonempty, we set $D'=D$ and we are done. Otherwise, we set $D\leftarrow D-\mathbf{1}_U$. Then $X\subseteq\operatorname{supp}(D)$, $R\cap\operatorname{supp}(D)=\emptyset$ and we iterate. We must finish in no more than $\deg(D)\cdot|V|$ iterations by Lemma 1 and Lemma 4. Hence, we can find the required D' and U in time $|E(G)|\cdot|V(G)|\deg(D)$. □

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 $k=\deg(D)$. 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 $X=\operatorname{supp}(D)$. 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 $X\subseteq\operatorname{supp}(D')$ and $R\cap\operatorname{supp}(D')=\emptyset$. 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 6 we can find an effective divisor $D''\sim D'$ such that $X\subseteq\operatorname{supp}(D'')$, $R\cap\operatorname{supp}(D'')=\emptyset$ 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:

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

where we set $X'_0 = X$. Using (1) and the fact that $X'_0 \subseteq \operatorname{supp}(D'')$, it is easy to check that $|X'_i| \leq k$ 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 $(X, R) \to (X_1, R_1) \to (X'_1, R_1) \to \cdots \to (X'_t, R_t)$ 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 5, we are done in at most $|V(G)|^2$ steps. This completes the construction. By combining the construction described above with that of the lemma below, we obtain Theorem 2. Note that so far only a non-constructive proof that the divisorial gonality of a graph is an upper bound for the treewidth was known [10]. See [6] for the proof of the next lemma.

Lemma 7. *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 $k-1$.*

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 be the graph as in Figure 1. Let D be the divisor on G that has value 3 on vertex a and value 0 elsewhere. If we follow the construction of Section 3, we will end up with the monotone search strategy found in Figure 2. We start with the root node (X, R) with $X = \emptyset$ and $R = V$ and connect it to the node $(\operatorname{supp}(D), V \setminus \operatorname{supp}(D))$. The three ways of growing the tree (steps I, II, III) are indicated in the picture. The four occurrences of step III are explained below.

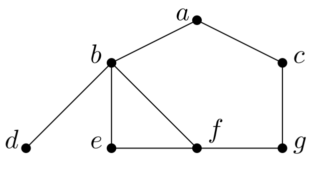

Fig. 1. An example graph G. It has divisorial gonality equal to 3.

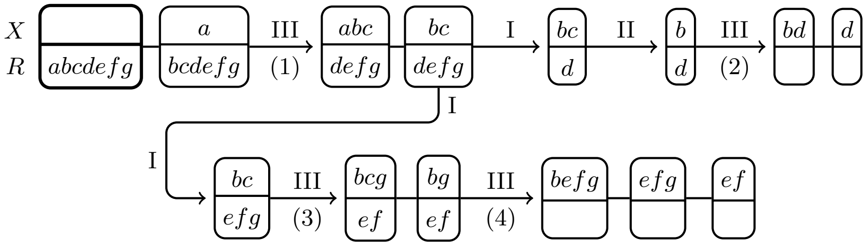

Fig. 2. The monotone search strategy obtained from G with divisor $D = 3a$. Each node shows the corresponding pair (X, R) with the root being $(\emptyset, \{a, b, c, d, e, f, g\})$. The labels I–III refer to the steps in the construction.

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

(1) Divisor D' is equal to $3a$. We fire the set $\{a\}$ and obtain the new divisor $a + b + c$.
(2) Divisor D' is equal to $a + b + c$. We fire the set $\{a, b, c, e, f, g\}$ and obtain the new divisor $a + c + d$.
(3) Divisor D' is equal to $a + b + c$. We fire the set $\{a, c\}$ and obtain the new divisor $2b + g$.
(4) Divisor D' is equal to $2b + g$. We fire the set $\{a, b, c, d, g\}$ and obtain the new divisor $e + 2f$.

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) divisiorial gonality is the notion of *stable gonality*; see [8]. 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 $f : V(G) \cup E(G) \to V(H) \cup E(H)$ 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: $f : G \to H$) is graph homomorphism f from G to H together with an *index function* $r_f : E(G) \to \mathbb{Z}_{>0}$.

A finite morphism $f : G \to H$ 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 $\deg(f)$, the *degree* of f, such that for all vertices $w \in V(H)$ and edges $e \in E(H)$, we have

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

see [14, Lemma 2.12] and [3, 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.

Lemma 8. *Let G be an undirected connected graph without loops and at least one edge. Given a tree T and a finite harmonic morphism $f : G \to T$ 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 8). 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 $e_1 = \{v_1, w_1\}, \ldots, e_{k'} = \{v_{k'}, w_{k'}\}$ with $f(v_1) = f(v_2) = \cdots = f(v_{k'}) = i$ and $f(w_1) = f(w_2) = \cdots = f(w_{k'}) = j$. Let $i_1, i_2, \ldots, i_{k'}$ 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 $r \in \{1, \ldots, k'\}$. The construction is illustrated in Figure 3. We claim that this yields a tree decomposition of G of width at most k.

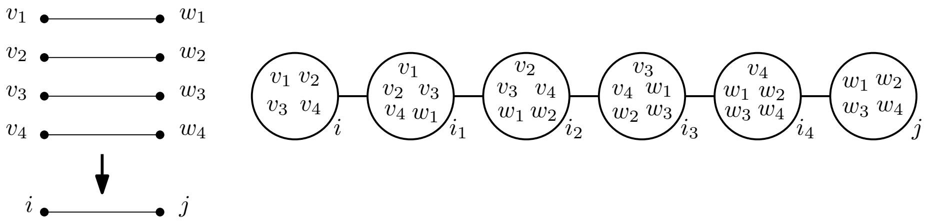

Fig. 3. Example of a step in the proof of Lemma 8. Here $k' = 4$. Left: four edges are mapped to the edge $\{i, j\}$ by the finite harmonic morphism. Right: the corresponding bags in the tree decomposition.

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, $f(v)$ 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.) $\square$

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 $f : G' \to T$ 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 8, 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. ☐

Acknowledgements

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

References

1. M. Baker. Specialization of linear systems from curves to graphs. *Algebra & Number Theory*, 2(6):613–653, 2008.
2. M. Baker and S. Norine. Riemann–Roch and Abel–Jacobi theory on a finite graph. *Advances in Mathematics*, 215(2):766–788, 2007.
3. M. Baker and S. Norine. Harmonic morphisms and hyperelliptic graphs. *International Mathematics Research Notices*, 2009(15):2914–2955, 2009.
4. U. Bertele and F. Brioschi. *Nonserial Dynamic Programming.* Academic Press, New York, 1972.
5. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.
6. H. L. Bodlaender, J. van Dobben de Bruyn, D. Gijswijt, and H. Smit. Constructing tree decompositions of graphs with bounded gonality, 2020. ArXiV e-prints, 2005.05569.
7. F. Cools, J. Draisma, S. Payne, and E. Robeva. A tropical proof of the Brill–Noether theorem. *Advances in Mathematics*, 230(2):759–776, 2012.
8. G. Cornelissen, F. Kato, and J. Kool. A combinatorial Li-Yau inequality and rational points on curves. *Mathematische Annalen*, 361(1-2):211–258, 2015.
9. D. Dhar. Self-organized critical state of sandpile automaton models. *Physical Review Letters*, 64(14):1613, 1990.
10. J. van Dobben de Bruyn and D. Gijswijt. Treewidth is a lower bound on graph gonality, 2014. ArXiV e-prints, 1407.7055v2.
11. K. Hendrey. Sparse graphs of high gonality. *SIAM Journal on Discrete Mathematics*, 32(2):1400–1407, 2018.
12. N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of treewidth. *Journal of Algorithms*, 7:309–322, 1986.
13. P. D. Seymour and R. Thomas. Graph searching and a minimax theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58:239–257, 1993.
14. H. Urakawa. A discrete analogue of the harmonic morphism and Green kernel comparison theorems. *Glasgow Mathematical Journal*, 42(3):319–334, 2000.