Recognizing Hyperelliptic Graphs in Polynomial Time

Jelco M. Bodewes[1], Hans L. Bodlaender[1,3⋆], Gunther Cornelissen[2], and Marieke van der Wegen[1,2⋆⋆]

[1] Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands.
[2] Mathematical Institute, Utrecht University, P.O. Box 80.010, 3508 TA Utrecht, The Netherlands
[3] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. Based on analogies between algebraic curves and graphs, Baker and Norine introduced *divisorial gonality*, a graph parameter for multigraphs related to treewidth, multigraph algorithms and number theory. We consider so-called *hyperelliptic graphs* (multigraphs of gonality 2) and provide a safe and complete set of reduction rules for such multigraphs, showing that we can recognize hyperelliptic graphs in time $O(n \log n + m)$, where n is the number of vertices and m the number of edges of the multigraph. A corollary is that we can decide with the same runtime whether a two-edge-connected graph G admits an involution σ such that the quotient $G/\langle\sigma\rangle$ is a tree.

1 Introduction

Motivation In this paper, we consider a graph theoretic problem that finds its origin in algebraic geometry, and can be formulated in terms of a specific type of graph search, namely *monotone chip firing*. The case with two chips is of special interest in the application, and we show that we can decide this case in $O(n \log n + m)$ time on a multigraph with n vertices and m edges.

In algebraic geometry, a special role is played by so-called *hyperelliptic* curves; these are smooth projective algebraic curves possessing an involution, i.e. an automorphism of order two, for which the quotient is the projective line. Such curves can be described by an affine equation $y^2 = f(x)$, for some one-variable polynomial $f(x)$ without repeated roots. They are widely studied and used, for example in the study of moduli spaces of abelian surfaces, invariants of binary quadratic forms, diophantine problems (finding integer or rational solutions to such equations), and in so-called "hyperelliptic curve cryptography" (see, e.g., [15] and [28]).

⋆ This author was partially supported by the NETWORKS project, funded by the Netherlands Organisation for Scientific Research.

⋆⋆ Corresponding author. M.vanderWegen@uu.nl

Recognizing hyperelliptic curves is an important, decidable problem in algorithmic algebraic geometry; an algorithm has been implemented when the curve is given by some set of polynomial equations, e.g., in the computer algebra package MAGMA [13]. No exact runtime analysis is available, but, the method being dependent on Gröbner basis computations, worst-case performance is expected to be more than exponential in the input size.

In recent work of Baker and Norine [5], the notion of a "hyperelliptic graph" was introduced, based on an analogy between algebraic curves and multigraphs. We show that the recognition problem for hyperelliptic graphs can be solved in quasilinear time. This can be applied to the recognition of certain hyperelliptic curves, since if an algebraic curve has a non-hyperelliptic stable reduction graph, the curve itself cannot be hyperelliptic (see [4, 3.5]).

Divisorial gonality Hyperelliptic graphs are graphs with *divisorial gonality* at most two. The notion of divisorial gonality has several equivalent definitions; intuitively, we use a chip firing game: we have a graph and some initial configuration that assigns a non-negative number of "chips" to each vertex. We can fire a subset of vertices by moving a chip along each outgoing edge of the subset, *if* every vertex has sufficiently many chips. We say that an initial configuration reaches a vertex if a sequence of firings results in that vertex having at least one chip. The divisorial gonality of a graph is the minimum number of chips needed for an initial configuration to reach each vertex of the graph. It actually suffices to consider a 'monotone' variant of the chip firing procedure, in which the sequence of subsets that are fired to reach a vertex is increasing; this is similar to several other graph search games, where the optimal number of searchers does not increase when we require the search to be monotone, see e.g., [7,25].

Known results The termination of similar Mancala-style games was discussed by Björner, Lovász and Shor [8]. In the guise of "abelian sandpile model", they play an important role in the study of self-organized criticality in statistical physics [3,19]. The chip firing game introduced by Baker and Norine is relevant for classical combinatorial problems about graphs, relating to spanning trees [14], the uniqueness of graph involutions [5], and potential theory on electrical network graphs [6]. A polynomial bound on the minimal number of required firings to terminate the Björner, Lovász and Shor-game was given by Tardos [30].

We study the divisorial gonality of graphs from the point of view of computational complexity. The analogous problem of computing the gonality of an algebraic curve is decidable [29].

The divisorial gonality of a graph G is related to treewidth, $\mathrm{tw}(G)$, by an inequality [21]

$$\mathrm{dgon}(G) \geq \mathrm{tw}(G). \tag{1}$$

Since treewidth is insensitive to the presence of multiple edges while divisorial gonality is not, the parameters are different; actually, they are not "tied" in the

sense of Norin [27]: there exists G with $\mathrm{tw}(G) = 2$ but $\mathrm{dgon}(G)$ arbitrarily high [24]. We know that treewidth is FPT, and that computing divisorial gonality is NP-hard and in XP [22], [20, Section 5].

Our results Our main result is the following.

Theorem A (=Theorem 1). *There is an algorithm that decides whether a graph G is hyperelliptic in $O(n \log n + m)$ time.*

To obtain our algorithm, we provide a safe and complete set of reduction rules. Similar to recognition algorithms for graphs of treewidth 2 or 3 (see [1]), in our algorithm the rules are applied to the graph until no further rule application is possible; we decide positively if and only if this results in the empty graph. One novelty is that some of the rules introduce constraints on pairs of vertices, which we model by colored edges. To deal with the fact that some of the rules are not local, we use a data structure that allows us to find an efficient way of applying these rules, leading to the stated running time. Omitted proofs and details can be found in [9], in which we also consider other variants of gonality.

The computational complexity of the problem "Does a graph admit a non-trivial automorphism" (solvable in quasi-polynomial time [2]) is very sensitive to alterations of the question. For example, deciding whether a graph has a *fixed point free* automorphism of order two is NP-complete (see Lubiw [26]). Our main result implies the following result as corollary.

Corollary A (=Corollary 1). *There is an algorithm that, given a two-edge-connected graph G, decides in $O(n \log n + m)$ time whether G admits an involution σ such that the quotient $G/\langle\sigma\rangle$ is a tree.*

2 Preliminaries

2.1 Definitions

Whenever we write "graph" we refer to a multigraph $G = (V, E)$, where V is the set of vertices and E is a multiset of edges.

There is a number of different definitions of divisorial gonality. The one we use is shown to be equivalent to the chip firing procedure without the 'monotonicity' property by [20]. The definition given here allows us to prove correctness of the reduction rules in our algorithm, and avoids more heavy algebraic terminology.

A *divisor* D in a graph $G = (V, E)$ is a mapping $D\colon V \to \mathbb{Z}$ (a divisor represents a distribution of chips, see Section 1). We call a divisor D *effective* (notation $D \geq 0$) if $D(v) \geq 0$ for all $v \in V$. The degree, $\deg(D)$, of a divisor D equals $\sum_{v \in V} D(v)$.

Given an effective divisor D and a set of vertices $W \subseteq V$, we call W *valid for* D, if for each $v \in W$, $D(v) \geq |E(v, V \setminus W)|$ (i.e., v has at least as many chips as it has neighbors in $V \setminus W$). If W is valid for D, we can *fire* W starting from D, this yields another divisor: for $v \in W$, $D(v)$ is decreased by the number of edges from v to $V \setminus W$, and for $x \in V \setminus W$, $D(x)$ is increased by the number of edges

from W to x. Intuitively, firing W means moving a chip along all edges from W to $V \setminus W$. Note that the divisor obtained by firing is effective as well.

We call two effective divisors D and D' *equivalent*, in notation $D \sim D'$, if there is a sequence of subsets $A_1 \subseteq A_2 \subseteq \ldots \subseteq A_{k-1} \subset A_k = V$, such that for all i the set A_i can be fired when $A_1, \ldots, A_{i-1}$ are fired starting from D, and the divisor obtained by firing $A_1, \ldots, A_k$ is D'. This defines an equivalence relation on the set of effective divisors [20, Chapter 3]. For two equivalent effective divisors D and D', we call the difference of functions $D' - D$ the *transformation* from D to D', and the sequence $A_1 \subseteq A_2 \subseteq \ldots \subseteq A_{k-1} \subset A_k = V$ the *level set decomposition* of this transformation. This level set decomposition is unique [20, Remark 3.8].

We say that an effective divisor D *reaches* a vertex v, if there exists a D' such that $D \sim D'$ and $D'(v) \geq 1$. The *divisorial gonality*, $\mathrm{dgon}(G)$, of a graph G is the minimum degree of an effective divisor D that reaches each vertex of G.

Example 1. Let T be a tree. Then T has divisorial gonality 1. Let v be a vertex of T and consider the divisor D with $D(v) = 1$ and $D(x) = 0$ for all $x \neq v$. This divisor has degree 1 and reaches each vertex of T: Let w be a vertex of T. Let vu be the first edge on the unique path from v to w. Let A_v be the component that contains v of the cut induced by vu. Firing A_v yields the divisor $D(u) = 1$ and $D(x) = 0$ for all $x \neq u$, thus we moved a chip from v to u. Repeating this process yields a divisor with a chip on w.

Example 2. Let G be a cycle, then G has divisorial gonality 2. First note that every set of vertices of G induces a cut of size at least 2. Hence for all degree 1 divisors, there are no valid sets. Hence a degree 1 divisor does not reach every vertex. To see that there is a divisor with 2 chips that reaches every vertex, number the vertices $v_1, v_2, \ldots, v_n$ and consider the divisor D with a chip on v_1 and a chip on v_n. To reach a vertex v_k with $k \leq \frac{n}{2}$, fire the set $\{v_i \mid 1 \leq i \leq j\} \cup \{v_i \mid n - j + 1 \leq i \leq n\}$ for $j = 1, 2, \ldots, i - 1$. Analogous for a vertex v_k with $\frac{n}{2} \leq k \leq n$.

Example 3. Consider the graph G in Figure 1. This graph has treewidth 1 and divisorial gonality 3. A divisor that reaches all vertices either has a chip on u and 2 more chips to reach both v and w, or has at least 3 chips to move along the three edges from v to u. See also [16, Table 3].

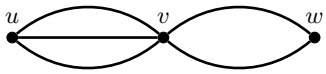

Fig. 1. Graph with divisorial gonality 3 and treewidth 1 (see Example 3).

Example 4. Consider the graph G in Figure 2. This graph has treewidth 2 and divisorial gonality 3. A divisor that reaches all vertices needs two chips to traverse the left cycle and 2 chips to traverse the right cycle. But we cannot move two chips from u to v, so these two chips on the left side cannot be the same as the two on the right side. Hence we need at least three chips.

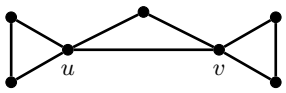

Fig. 2. Graph with divisorial gonality 3 and treewidth 2 (see Example 4).

2.2 Constraints

General Constraints In the process of applying reduction rules to a graph, we will need to keep track of certain restrictions otherwise lost by removal of vertices and edges. We will maintain these restrictions in the form of a set of pairs of vertices, called constraints, and then extend the notion of divisorial gonality to graphs with constraints.

Definition 1. *Given a graph $G = (V, E)$, a* constraint *on G is an unordered pair of vertices $v, w \in V$, usually denoted as (v, w), where v and w can be the same vertex.*

Constraints are, like edges, pairs of vertices, so we can consider them as an extra set of edges. We will use $\mathcal{C}$ to represent this set.

Checking whether a graph has gonality two or lower is the same as checking whether there exists a divisor on our graph with degree two that reaches all vertices. Our constraints place restrictions on what divisors we consider, as well as what sets we are allowed to fire.

Definition 2. *Given a set of constraints $\mathcal{C}$, and two equivalent effective divisors D and D'. We call D and D'* $\mathcal{C}$-equivalent *(in notation $D \sim_{\mathcal{C}} D'$), if for every set A_i of the level set decomposition of $D' - D$ and every constraint $(u, v) \in \mathcal{C}$, either $u, v \in A_i$ or $u, v \notin A_i$.*

Note that this defines a finer equivalence relation. Now we can extend the definition of *reach* using $\mathcal{C}$-equivalence: a divisor D *reaches* a vertex v, if there exists a D' such that $D \sim_{\mathcal{C}} D'$ and $D'(v) \geq 1$.

Definition 3. *Given a set of constraints $\mathcal{C}$. A divisor D* satisfies $\mathcal{C}$ *if for every constraint $(u, v) \in \mathcal{C}$ there is a divisor $D' \sim_{\mathcal{C}} D$ such that $D'(u) \geq 1$ and $D'(v) \geq 1$ if $u \neq v$ and $D'(u) \geq 2$ if $u = v$.*

Definition 4. *Given a graph $G = (V, E)$ with constraints $\mathcal{C}$, we call a divisor D* suitable *if it is effective, has degree 2, reaches all vertices using the $\mathcal{C}$-equivalence relation and satisfies all constraints in $\mathcal{C}$.*

Definition 5. *We will say that a graph with constraints has divisorial gonality 2 or lower if it admits a suitable divisor. Note that for a graph with no constraints this is equivalent to the usual definition of divisorial gonality 2 or lower. We will denote the class of graphs with constraints that have divisorial gonality two or lower as $\mathcal{G}_2$.*

Constraints & Cycles It will be useful to determine when constraints are non-conflicting locally:

Definition 6. *Let C be a cycle in a graph G with constraints $\mathcal{C}$. Let $\mathcal{C}_C \subseteq \mathcal{C}$ be the subset of the constraints that contain a vertex in C. We call the constraints $\mathcal{C}_C$* compatible *if the following hold.*

(i) *If $(v, w) \in \mathcal{C}_C$ then both $v \in C$ and $w \in C$.*
(ii) *For each $(v, w) \in \mathcal{C}_C$ and $(v', w') \in \mathcal{C}_C$, the divisor given by assigning a chip to v and w must be equivalent to the one given by assigning a chip to v' and w' on the subgraph consisting of C.*

2.3 Reduction Rules, Safeness and Completeness

A *reduction rule* is a rule that can be applied to a graph to produce a smaller graph. Our final goal with the set of reduction rules is to show that it can be used to characterize the graphs in a certain class, that of the graphs with divisorial gonality two, by reduction to the empty graph. For this we need to make sure that membership of the class is invariant under our reduction rules.

Definition 7. *Let $\boldsymbol{U}$ be a rule and $\boldsymbol{S}$ be a set of reduction rules. Let $\mathcal{A}$ be a class of graphs. We call $\boldsymbol{U}$* safe *for $\mathcal{A}$ if for all graphs G and H such that H can be produced by applying rule $\boldsymbol{U}$ to G it follows that $H \in \mathcal{A} \iff G \in \mathcal{A}$. We call $\boldsymbol{S}$* safe *for $\mathcal{A}$ if every rule in $\boldsymbol{S}$ is safe for $\mathcal{A}$.*

Apart from our rule sets being safe, we also need to know that, if a graph is in our class, it is always possible to reduce it to the empty graph.

Definition 8. *Let $\boldsymbol{S}$ be a set of reduction rules and $\mathcal{A}$ be a class of graphs. We call $\boldsymbol{S}$* complete *for $\mathcal{A}$ if for any graph $G \in \mathcal{A}$ it holds that G can be reduced to the empty graph by applying some finite sequence of rules from $\boldsymbol{S}$.*

For any rule set that is both complete and safe for $\mathcal{A}$ the rule set is suitable for characterizing $\mathcal{A}$: a graph G can be reduced to the empty graph if and only if G is in $\mathcal{A}$. Additionally it is not possible to make a wrong choice early on that would prevent the graph from being reduced to the empty graph: if $G \in \mathcal{A}$ and G can be reduced to H, then H can be reduced to the empty graph.

These properties ensure that we can use the set of reduction rules to create an algorithm for recognition of the graph class.

3 Reduction Rules for Divisorial Gonality

We will now show that there exists a set of reduction rules that is safe and complete for the class of graphs with divisorial gonality at most two. We will assume that our graph is loopless and connected. Loops can simply be removed from the graph since they never impact the divisorial gonality and a disconnected graph has divisorial gonality two or lower exactly when it consists of two trees, which can easily be checked in linear time. All reduction rules below maintain connectedness.

The Reduction Rules

We are given a connected loopless graph $G = (V, E)$ and a yet empty set of constraints $\mathcal{C}$. The following rules are illustrated in Figure 3, where a constraint is represented by a red dashed edge.

We start by covering the two possible end states of our reduction:

Rule E_1. *Given a graph consisting of exactly one vertex, remove that vertex.*

Rule E_2. *Given a graph consisting of exactly two vertices, u and v, connected to each other by a single edge, and $\mathcal{C} = \{(u, v)\}$, remove both vertices.*

Next are the reduction rules to get rid of vertices with degree one. These rules are split by what constraint applies to the vertex:

Rule T_1. *Let v be a leaf, such that v has no constraints in $\mathcal{C}$. Remove v.*

Rule T_2. *Let v be a leaf, such that its only constraint in $\mathcal{C}$ is (v, v). Let u be its neighbor. Remove v and add the constraint (u, u) if it does not exist yet.*

Rule T_3. *Let v_1 be a leaf, such that its only constraint in $\mathcal{C}$ is (v_1, v_2), where v_2 is another leaf, whose only constraint is also (v_1, v_2). Let u_1 be the neighbor of v_1 and u_2 be the neighbor of v_2 (these can be the same vertex). Then remove v_1 and v_2 and add the constraint (u_1, u_2) if it does not exist yet.*

Finally we have a set of reduction rules that apply to cycles containing at most 2 vertices with degree greater than two. The rules themselves are split by the number of vertices with degree greater than two.

Rule C_1. *Let C be a cycle of vertices with degree two. If the set of constraints $\mathcal{C}_C$ on C is compatible, then replace C by a new single vertex.*

Rule C_2. *Let C be a cycle with one vertex v with degree greater than two. If the set of constraints $\mathcal{C}_C$ on C plus the constraint (v, v) is compatible, then remove all vertices except v in C and add the constraint (v, v) if it does not exist yet.*

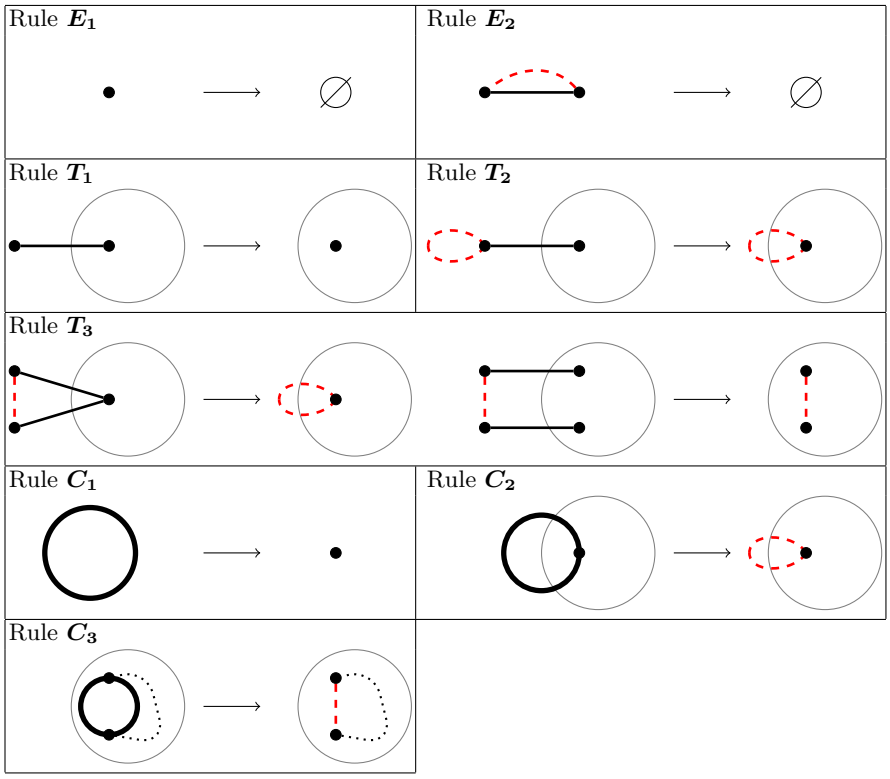

Fig. 3. The reduction rules for divisorial gonality

Rule C_3. *Let C be a cycle with two vertices v and u of degree greater than two. If there exists a path from v to u that does not share any edges with C and the set of constraints $\mathcal{C}_C$ on C plus the constraint (v, u) is compatible, then remove all vertices of C except v and u, remove all edges in C and add the constraint (v, u) if it does not exist yet.*

We denote by $\mathcal{R}$ the set consisting of all the above reduction rules: $\boldsymbol{E_1}$, $\boldsymbol{E_2}$, $\boldsymbol{T_1}$, $\boldsymbol{T_2}$, $\boldsymbol{T_3}$, $\boldsymbol{C_1}$, $\boldsymbol{C_2}$ and $\boldsymbol{C_3}$.

In the rest of this paragraph we will present safeness proofs for some of the rules and the more interesting parts of the proof of completeness. Details are found in [9, Section 4] .

Proposition 1 (Safeness). *The set of rules $\mathcal{R}$ is safe for $\mathcal{G}_2$.*

Proof. We need to proof that all rules are safe, we will show this for rules $\boldsymbol{T_3}$ and $\boldsymbol{C_3}$ below, for the other rules, see [9, Lemma 4.6, 4.7, 4.9, 4.10].
Claim 1: Rule $\boldsymbol{T_3}$ is safe. Let v_1 and v_2 be the vertices with degree one, such that their only constraint is (v_1, v_2) and let u_1 and u_2 be their (possibly equal) neighbors. We first assume that $H \in \mathcal{G}_2$, then there is a suitable divisor on H

with one chip on u_1 and another chip on u_2. Consider this divisor on G. Then by firing $V(G) \setminus \{v_1, v_2\}$ we can move a chip to v_1 and v_2. For every vertex $v \in V(G) \setminus \{v_1, v_2\}$ there is a sequence $A_0, A_1, \ldots, A_k \subseteq V(H)$ such that firing this sequence yields a divisor D' with a chip on v. Now add v_i to every set A_j that contains u_i. Firing these sets on G starting from D results in D' on G, so D reaches v. Moreover, every set we fired contains either both v_1 and v_2, or neither. We conclude that D is also suitable on G.

Assume that $G \in \mathcal{G}_2$, then the divisor on G with one chip on v_1 and v_2 is suitable. By firing $\{v_1, v_2\}$ we can create a divisor with a chip on u_1 and u_2 (or two on u_1 if $u_1 = u_2$). It follows that this divisor is suitable when considered on H.

Claim 2: Rule $\boldsymbol{C_3}$ *is safe.* Let C be our cycle and v, w the two vertices with degree greater than two in C. We first assume that $H \in \mathcal{G}_2$. From this it follows that the divisor on H with a chip on v and a chip on w is suitable. We know that in G all constraints on C plus (v, w) are compatible. From this we see that if we consider the divisor on G it will be able to satisfy all constraints on C. It is also clear that from v and w we can move chips along either of the two arcs between v and w in C. Therefore the divisor is also suitable on G and thus $G \in \mathcal{G}_2$.

Let us now assume that instead $G \in \mathcal{G}_2$. Clearly there exists a suitable divisor D on G that has a chip on v. We will show that there is a suitable divisor that has a chip on both v and w: Assume that $D(w) = 0$, then there should be a suitable divisor D' with $D'(w) = 1$ and $D \sim_{\mathcal{C}} D'$. This implies there is a level set decomposition $A_0, \ldots, A_k$ of the transformation from D to D'.

Let A_i be the first subset that contains w and D_i the divisor before firing A_i. Note that we have $D_i(a) \geq |E(a, V(G) \setminus A_i)|$ for all $a \in A_i$, since all firing sets are valid. Since $\deg(D_i) = 2$ it follows that $\sum_{a \in A_i} |E(a, V(G) \setminus A_i)| \leq 2$. This is the same as the cut induced by A_i having size two or lower. The minimum cut between v and w is at least three, since they are both part of C and there exists an additional path outside of C between them. Therefore it follows that A_i can only induce a cut of size two or lower if $w \in A_i$. But this implies that $D_i(w) \geq 1$, since a vertex can not receive a chip after entering the firing set. We conclude that $D_i(v) = 1$ and $D_i(w) = 1$.

Also by the fact that the minimum cut between v and w is at least three it follows that a subset firing can only be valid if the subset contains either both v and w or neither (since otherwise the subset would have at least three outgoing edges). It follows we can satisfy the set of constraints including (v, w).

Therefore the divisor D_i gives us a suitable divisor when considered on H. We conclude that $H \in \mathcal{G}_2$. □

By the previous proposition we now have that membership in $\mathcal{G}_2$ is invariant under the reduction rules in $\mathcal{R}$. For the reduction rules to be useful however we will also need to confirm that any graph can be reduced to the empty graph by a finite sequence of rule applications.

Proposition 2 (Completeness). *The set of rules $\mathcal{R}$ is complete for $\mathcal{G}_2$.*

Proof. Let $G \in \mathcal{G}_2$ be a non-empty graph.

Claim 1: A rule in $\mathcal{R}$ can be applied to G. Assume instead that no rule in $\mathcal{R}$ can be applied to G.

Claim 2: G contains no vertices of degree 1 [9, Lemma 4.14]. It follows that all vertices of G have degree at least 2. Consider the minor H of G created by contracting each path of only degree 2 vertices to an edge. Then any edge in H was either created by contraction of a path of any number of vertices with degree 2 in G or it already was an edge in G.

If H contains a loop, there is a path of degree 2 vertices in G going from a degree 3 or greater vertex to itself (since G contains no loops), so this path plus the vertex it is attached to forms a cycle with exactly one vertex of degree 3 or greater. Since we cannot apply Rule $\boldsymbol{C_2}$ to G, it follows that the constraints $\mathcal{C}_C$ are not compatible. This contradicts the following claim:

Claim 3: For every cycle C in G, the constraints $\mathcal{C}_C$ are compatible [9, Lemma 4.15]. Hence H contains no loops.

Now we find a subgraph H' of H with no multiple edges. If H contains no multiple edges, simply let $H' = H$. Otherwise let v and w be two vertices such that there are at least two edges between v and w. Suppose that v and w are still connected to each other after removing two edges e_1, e_2 between them. The removed edges each represent a single edge or a path of degree 2 vertices in G. Thus v, w plus these paths form a cycle C in G with exactly two vertices of degree 3 or greater, where there is also a path between v and w that does not share any edges with C. By Claim 3 we have that the constraints on this cycle are compatible and so we are able to apply Rule $\boldsymbol{C_3}$ to C. Since we cannot apply any rules to G, it follows that G must be disconnected after removing e_1 and e_2. So any multiple edge in H consists of a double edge, whose removal splits the graph in two connected components. Let H' be the connected component of minimal size over all possible removals of a double edge in H. Note that H' cannot contain any double edge, since this would imply a smaller connected component.

We now have a minor H' of G, which is a simple graph since it has no loops or multiple edges. Also, each vertex of H' has degree at least 3 with at most one exception, namely the vertex that was incident to the two parallel edges that were removed to obtain H'. Since a graph with treewidth at most two has at least two vertices of degree at most two, it follows that $\mathrm{tw}(H') \geq 3$ [12, Lemma 4]. Since treewidth is closed under taking minors we get $\mathrm{tw}(G) \geq 3$. But then by Equation 1 it follows that $\mathrm{dgon}(G) \geq 3$, creating a contradiction, since $G \in \mathcal{G}_2$. We conclude that our assumption must be wrong and there must be a rule in $\mathcal{R}$ that can be applied to G.

Assume that $G \in \mathcal{G}_2$. By Claim 1 and Proposition 1 we can keep applying rules from $\mathcal{R}$ to G as long as G has not been turned into the empty graph yet. Observe that each rule removes at least one vertex or at least two edges, while never adding more vertices or edges. Since G is finite, rules from $\mathcal{R}$ can only be applied a finite number of times. When no more rules can be applied, it follows that the graph has been reduced to the empty graph. Therefore $\mathcal{R}$ is complete. □

4 Main Algorithm

In this section, we discuss how the reduction rules of Sections 3 lead to an efficient algorithm that recognize graphs with divisorial gonality 2 or lower.

Theorem 1 (= Theorem A). *There is an algorithm that, given a graph G, decides whether* $\operatorname{dgon}(G) \leq 2$ *in* $O(m + n \log n)$ *time.*

Proof. We introduce a new rule that shortcuts repeated applications of Rule $\boldsymbol{C_3}$.:

Rule $\boldsymbol{M}$. *Let u, v be vertices, such that $|E(u,v)| \geq 3$. Remove $2\left\lfloor \frac{|E(u,v)|-1}{2} \right\rfloor$ edges between u and v and add a constraint (u,v).*

All applications of this rule can be done in $O(m)$ at the start of the algorithm, after which we know that no pair of vertices has more than two edges between them.

Since treewidth is a lower bound on divisorial gonality (Equation 1), it follows that if $\operatorname{tw}(G) > 2$, the algorithm can terminate. Checking whether treewidth is at most 2 can be done in linear time. Hereafter, we assume our graph has treewidth at most 2.

The remainder of the algorithm is of the following form: repeatedly try to apply a safe rule, until none is possible. If no rule is applicable, we can directly decide, as safeness and completeness of our set of rules implies that $\operatorname{dgon}(G) \leq 2$, if and only if the resulting graph is empty. We now discuss how this can be done in $O(n \log n)$ time.

As graphs of treewidth k and n vertices have at most kn edges, the underlying simple graph has at most $2n$ edges. There are at most 2 edges between a pair of vertices and no loops, so at most $4n$ edges in total. Note that each rule application decreases the sum of the number of vertices and the number of edges by at least one, so $O(n)$ rules can be applied before we reach the empty graph.

For most rules, standard data structures allow to find applicable rules in amortized constant time. For Rules $\boldsymbol{C_2}$. and $\boldsymbol{C_3}$. we employ a technique used in [10]: we use a formulation in monadic second order logic (MSOL), and a data structure, based upon a tree decomposition of G of logarithmic depth and constant width allows to perform queries and updates in $O(\log n)$ time each. (See also [23,18].)

The main idea is as follows: by [11, Lemma 2.2], we can build in $O(n)$ time a tree decomposition of G of width 8, such that the tree T in the tree decomposition is binary and has $O(\log n)$ depth. We augment the graph by labels that express for vertices and edges whether they are contracted, deleted, or carry a constraint. For each of the Rules $\boldsymbol{C_2}$. and $\boldsymbol{C_3}$., we can express the property that these can be applied to the graph obtained after a number of rule applications as a sentence in MSOL on the original graph G augmented with the labeling relations Contracted, Deleted and Carry-a-Constraint. The sentences have free variables that allow to find where in the graph the modification can take place. A modification of Courcelle's algorithm [17] gives that each query and each graph update can be

done in time linear in the depth of the tree decomposition, i.e., $O(\log n)$ time. More details are given in [9, Section 7].

As the time per application of a safe rule is bounded by $O(\log n)$, and we execute $O(n)$ rule applications, the total time is bounded by $O(n \log n)$. □

Corollary 1 (= Corollary A). *There is an algorithm that, given a two-edge-connected graph G, decides whether or not G admits an automorphism σ of order two such that the quotient $G/\langle\sigma\rangle$ is a tree, in $O(n \log n + m)$ time.*

Proof. This follows from Theorem 1, since Baker and Norine [5, Thm. 5.12] have shown that a two-edge-connected graph G is hyperelliptic precisely if G admits an automorphism as stated in the theorem. □

5 Conclusion

In this text, we have focused on *divisorial* gonality, defined by analogy with the theory of divisors on algebraic curves and described in terms of chip-firing games. We gave a quasilinear detection algorithm for dgon ≤ 2. Different flavours of gonality exist, based on analogies with the theory of coverings of algebraic curves; or "stable" versions (in which the graph can be refined), based on ideas from the theory of tropical curves (see [16]). In [9], we give quasilinear time detection algorithms for these variants being two, too.

Finally, we mention some interesting open questions on (divisorial) gonality from the point of view of algorithmic complexity: (a) Can hyperelliptic graphs be recognized in linear time? (b) Which problems become fixed parameter tractable with gonality as parameter? (c) Is there an analogue of Courcelle's theorem for bounded gonality? (d) Is divisorial gonality fixed parameter tractable?

References

1. Stefan Arnborg and Andrzej Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Algebraic Discrete Methods*, 7(2):305–314, 1986.
2. László Babai. Graph isomorphism in quasipolynomial time. Preprint arXiv: 1512.03547v2, 2016.
3. Per Bak, Chao Tang, and Kurt Wiesenfeld. Self-organized criticality. *Phys. Rev. A*, 38(1):364, 1988.
4. Matthew Baker. Specialization of linear systems from curves to graphs. *Algebra Number Theory*, 2(6):613–653, 2008. With an appendix by Brian Conrad.
5. Matthew Baker and Serguei Norine. Harmonic morphisms and hyperelliptic graphs. *Int. Math. Res. Not. IMRN*, 15:2914–2955, 2009.
6. Matthew Baker and Farbod Shokrieh. Chip-firing games, potential theory on graphs, and spanning trees. *J. Combin. Theory Ser. A*, 120(1):164–182, 2013.
7. Daniel Bienstock and Paul Seymour. Monotonicity in graph searching. *J. Algorithms*, 12:239–245, 1991.
8. Anders Björner, László Lovász, and Peter W. Shor. Chip-firing games on graphs. *European J. Combin.*, 12(4):283–291, 1991.

9. Jelco M. Bodewes, Hans L. Bodlaender, Gunther Cornelissen, and Marieke van der Wegen. Recognizing hyperelliptic graphs in polynomial time. Preprint arXiv: 1706.05670, 2017.
10. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.
11. Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.*, 27(6):1725–1746, 1998.
12. Hans L. Bodlaender and Arie M.C.A. Koster. Treewidth computations II. Lower bounds. *Information and Computation*, 209(7):1103–1119, 2011.
13. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.
14. Melody Chan, Darren Glass, Matthew Macauley, David Perkinson, Caryn Werner, and Qiaoyu Yang. Sandpiles, spanning trees, and plane duality. *SIAM J. Discrete Math.*, 29(1):461–471, 2015.
15. Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2012.
16. Gunther Cornelissen, Fumiharu Kato, and Janne Kool. A combinatorial Li-Yau inequality and rational points on curves. *Math. Ann.*, 361(1-2):211–258, 2015.
17. Bruno Courcelle. The monadic second-order logic of graphs. I: Recognizable sets of finite graphs. *Inform. and Comput.*, 85(1):12–75, 1990.
18. Bruno Courcelle and Rémi Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Appl. Math.*, 131(1):129 – 150, 2003.
19. Deepak Dhar. Self-organized critical state of sandpile automaton models. *Phys. Rev. Let.*, 64(14):1613, 1990.
20. Josse van Dobben de Bruyn. Reduced divisors and gonality in finite graphs. Bachelor thesis, Leiden University, 2012. URL: `https://www.universiteitleiden.nl/binaries/content/assets/science/mi/scripties/bachvandobbendebruyn.pdf`.
21. Josse van Dobben de Bruyn and Dion Gijswijt. Treewidth is a lower bound on graph gonality. Preprint arXiv:1407.7055, 2014.
22. Dion Gijswijt. Computing divisorial gonality is hard. Preprint arXiv:1504.06713, 2015.
23. Torben Hagerup. Dynamic algorithms for graphs of bounded treewidth. *Algorithmica*, 27(3):292–315, 2000.
24. Kevin Hendrey. Sparse graphs of high gonality. Preprint arXiv:1606.06412, 2016.
25. Andrea S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245, April 1993.
26. Anna Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM J. Comput.*, 10(1):11–21, 1981.
27. Sergey Norin. New tools and results in graph minor structure theory. In *Surveys in combinatorics 2015*, volume 424 of *London Math. Soc. Lecture Note Ser.*, pages 221–260. Cambridge Univ. Press, 2015.
28. Bjorn Poonen. Computing rational points on curves. In *Number theory for the millennium, III (Urbana, IL, 2000)*, pages 149–172. A K Peters, Natick, MA, 2002.
29. Josef Schicho, Frank-Olaf Schreyer, and Martin Weimann. Computational aspects of gonal maps and radical parametrization of curves. *Appl. Algebra Engrg. Comm. Comput.*, 24(5):313–341, 2013.
30. Gábor Tardos. Polynomial bound for a chip firing game on graphs. *SIAM J. Discrete Math.*, 1(3):397–398, 1988.