Universiteit Utrecht

Bachelor Thesis

The stable gonality of finite graphs

Ragnar Groot Koerkamp

Supervisor:
Prof. dr. Gunther Cornelissen

January 2016

Abstract

This thesis is devoted to finding a finite algorithm to calculate the *stable gonality* of graphs. The stable gonality was first defined by Cornelissen et al. in [6]. A *finite harmonic morphism* φ is a map from a refinement H of a graph G to a tree T obeying several properties. Together with this comes a map $r_\varphi : E(H) \to \mathbb{N}$ which assigns *indices* to all edges of H. It turns out that we can assign a *degree* to every finite harmonic morphism. The stable gonality $\operatorname{sgon}(G)$ is the minimal degree of a finite harmonic morphism from a refinement of G to a tree.

The naive algorithm to calculate the stable gonality of a graph does not terminate, because we have to consider all refinements H, all trees T and all maps r_φ, which are three infinite loops. I describe a new algorithm to calculate $\operatorname{sgon}(G)$ in finite time, using several new theorems that reduce the set of refinements and trees we have to consider. First, I give a bound on the size of T is terms of G. Then, I replace T by a *minor-universal tree* containing all trees of a given size as a minor. Finally, I prove that we may assume that the indices of the edges of H have a nice property, and that they can be bounded from above by $|E(G)|$.

Contents

1 Introduction

The stable gonality of finite loopless graphs was first defined by Cornelissen et al. in [6] in 2015 as an analogy of the gonality of smooth projective curves. In this thesis, my goal is to find an algorithm to calculate this new invariant of graphs. Such an algorithm could be used to compare the stable gonality with other kinds of gonality, like the divisorial gonality and the treewidth of graphs. Furthermore, results on small graphs could give rise to new conjectures relating these and other kinds of graph invariants.

This thesis consists of three parts. In the first part, I will introduce finite harmonic morphisms and the stable gonality. For this, we need multisets and multigraphs, which I cover in the first two sections. These sections mostly contain a list of well known definitions, although some definitions are my own.

Next, we need refinements. These are special modifications graphs that we need in order to define the stable gonality. The notion of refinements is well known, but I define them in a formal inductive way that allows easy proofs of their properties. Also, using this inductive definition, it becomes easier to prove lemmas using refinements.

Using all this theory, I define finite harmonic morphisms, as defined in [6]. A finite harmonic morphism is a map φ from a refinement H of a graph G to a tree T, together with a map $r_\varphi : E(H) \to \mathbb{N}$ assigning indices to the edges of H. These maps φ and r have to obey some special properties to make φ a finite harmonic morphism. These properties allow us to define the degree of a finite harmonic morphism. Then, I define the stable gonality of a graph G as the minimal degree of a finite harmonic morphism from a refinement of G to a tree.

This is followed by a proof that the stable gonality is equal for refinement-equivalent graphs, Theorem 6.5.

The naive algorithm to calculate the stable gonality of a graph does not terminate in finite time, because, for example, we have to iterate over all refinements of the given input graph. In the second part of this thesis, Section 7, I describe the algorithm I developed to calculate the stable gonality in finite time. This section gives pseudo code of the algorithm, and I will explain why this algorithm works. It relies on some theorems about special properties of finite harmonic morphisms. If we assume that m, the number of edges of the input graph, grows at most polynomially in n, the number of vertices, the runtime of the algorithm is

$$n^{O(n \lg n + m)}$$

The new theorems I have come up with are explained in the third part, following the algorithm. These results are not only useful for the algorithm, but are also general results about properties of finite harmonic morphisms.

The most important results are Theorem 8.6 and Theorem 11.4. The former states that we can bound the number of nonleaf vertices (vertices with at least 2 edges incident to it) of the trees T by $|V(G)|$. Theorem 11.4 gives an upper bound on the indices of the vertices of H, which are defined together with finite harmonic morphisms. This theorem might be a first step towards a graph theoretic proof that

$$\operatorname{sgon}(G) \leq \left\lfloor \frac{|E(G)| - |V(G)| + 4}{2} \right\rfloor,$$

which Cornelissen et al. [6, Theorem B] have proven using theorems on smooth projective curves.

I thank Gunther Cornelissen for his valuable remarks and discussions while supervising me. Also, I thank my friends who proofread this thesis for finding several mistakes in grammar and maths, and suggesting clarifications where needed.

2 Multisets

Multisets are a generalization of sets where an element may occur multiple times. We will need this to define multigraphs in the next section.

This section is loosely based on Section 2 of [9]. Because multisets are less widely known than plain sets, the definitions and used notation vary among authors. The notation I introduce may not be compatible with those of other authors, but are most natural in the context of this thesis.

Definition 2.1 (Multiset)**.** A *multiset* M over Ω is a tuple $M = (\Omega, m)$ where Ω is some set and m is a function $m : \Omega \to \mathbb{Z}_{\geq 0}$. The set Ω is called the *domain* of M. For $\omega \in \Omega$, the *counting function* m gives the *multiplicity* of ω as $m(\omega)$.

We write m_M for the counting function of M. ♢

Remark 2.2. Since the codomain of m is $\mathbb{Z}_{\geq 0}$, every element of Ω has a finite multiplicity. However, Ω itself may be infinite.

Remark 2.3. A set S can also be seen as a multiset where every element in S has multiplicity 1, and all elements not in S have multiplicity 0. Thus m is just the indicator function $\mathbf{1}_S$ in this case. The domain Ω of m is implicit here, but it is usually clear from the context what the domain of m is.

Definition 2.4 (Empty multiset)**.** The empty multiset is written $\emptyset$, just like for normal sets. ♢

There are several ways to write down a multiset. Because every element can only occur finitely many times, we can explicitly list the contents of a multiset. This is done using a set like notation, where we use square brackets instead of curly braces. Like for sets, the order of the elements does not matter.

Example 2.5. The multiset $A = (\{a, b, c, d\}, f)$, with $f : a \mapsto 2, b \mapsto 3, c \mapsto 0, d \mapsto 1$ may also be written as

$$[a, a, b, b, b, d].$$

Because a multiset is basically just a function on a given domain, we may also write it as an ordinary relation as given by the graph of the function m. More precise, we can write

$$M = \big\{(\omega, m(\omega))\big|\omega \in \Omega\big\}.$$

Example 2.6. Continuing the previous example, we may also write A as

$$A = \big\{(a, 2), (b, 3), (c, 0), (d, 1)\big\}.$$

Remark 2.7. In the examples above, the multiplicity of c is 0. When we explicitly write down multisets, we often omit elements with multiplicity 0. Because of this, all elements for which no multiplicity is given may be assumed to have multiplicity 0. Hence, the last example could also have been written as

$$A = \big\{(a, 2), (b, 3), (d, 1)\big\}.$$

When dealing with multisets, we will mostly use the set like notation when the multisets only have a few elements. We will use the pairwise notation when there are a few elements with higher multiplicities. The more formal notation as given in

the definition will only be used in this section to define all common operations on multisets.

First, we define the element relation and the cardinality of multisets. Next, we define sub-multisets, the analogon of subsets.

Definition 2.8 (Element)**.** For a multiset $M = (\Omega, m)$, we write $x \in M$ when $x \in \Omega$ and $m(x) > 0$. We say that x is an *element* of M. ♢

Remark 2.9. Note that when $x \notin \Omega$, we can still write $x \notin M$, but usually the emphasis is on the fact that $m(x) > 0$. Otherwise, we could just have written $x \notin \Omega$.

Definition 2.10 (Cardinality)**.** The *cardinality* or *size* of a multiset $M = (\Omega, m)$ is

$$|M| = \sum_{\omega \in \Omega} m(\omega). \qquad \diamondsuit$$

Remark 2.11. In this thesis, we will only consider finite multisets, so $|M|$ is always finite. Hence the sum is well-defined.

Remark 2.12. When writing M as a subset of $\Omega \times \mathbb{N}$, it may be tempting to read $|M|$ as the number of elements of Ω, or the number of elements for which $m(\omega) > 0$. However, one should remember that M is a multiset.

Definition 2.13 (Sub-multiset)**.** A multiset $M = (\Omega, f)$ is a *sub-multiset* of a multiset $N = (\Omega, g)$, written $M \leq N$, if for all $\omega \in \Omega$ we have

$$f(\omega) \leq g(\omega).$$

If, additionally, there is an ω such that $f(\omega) < g(\omega)$, M is called a *proper* or *strict* sub-multiset of N and we write $M < N$. ♢

Naturally, multisets arise when elements of a given set must be counted more than once. We would therefore like some notation to express that *M is a multiset containing elements in the set S*. For this, we will first define the support of a multiset.

Definition 2.14 (Support)**.** The *support* of a multiset $M = (\Omega, m)$ is the *set*

$$\operatorname{supp}(M) = \{\omega | m(\omega) > 0\} = \{\omega | \omega \in M\}. \qquad \diamondsuit$$

Definition 2.15 (Multiset in set)**.** We write $M \lhd S$ when

$$\operatorname{supp}(M) \subset S.$$

This relation is written as *M is a multiset in S*. ♢

Remark 2.16. The definitions above allows us to say things like *let $M \lhd S$ be a multiset in S*. Often, S itself is a subset of a larger set Ω, for example a subset of the set of vertices of a graph. When this is the case, the domain of M is implicitly assumed to be Ω, and not S. Therefore, m is defined on all of Ω, and in particular $m(\omega)$ is also defined for $\omega \in \Omega \backslash S$.

Now that all relations between multisets are given, we are ready for the operations on multisets. These behave similar to the corresponding operations on normal sets.

The union and intersection of two multisets act as if the occurrences of an ω in two multisets is the same, and takes the maximum respectively minimum over the multiplicity in both multisets.

Definition 2.17 (Intersection)**.** The *intersection* of two multisets $M = (\Omega, f)$ and $N = (\Omega, g)$ is given by

$$M \cap N = (\Omega, \min(f, g)),$$

where $\min(f, g)(\omega) = \min(f(\omega), g(\omega))$. ♢

Definition 2.18 (Union)**.** Similarly, the *union* of two multisets M and N is given by

$$M \cup N = (\Omega, \max(f, g)),$$

where $\max(f, g)(\omega) = \max(f(\omega), g(\omega))$. ♢

We may also add and subtract multisets. For addition, the occurrences of an ω in both operands are considered to be different, and we add the multiplicities. The subtraction of multisets corresponds to the difference $M \backslash N$ of normal sets, where we remove all elements of N from M.

Definition 2.19 (Addition)**.** The *sum* of two multisets M and N is

$$M + N = (\Omega, f + g),$$

where $(f + g)(\omega) = f(\omega) + g(\omega)$. ♢

Remark 2.20. We may now write $\sum_{i \in I} M_i$ for the sum of several multisets. Similarly, $\bigcup$ and $\bigcap$ can be used for the union respectively intersection of several multisets.

For the difference, we have to be a bit more careful, since we can not have negative multiplicities.

Definition 2.21 (Subtraction)**.** The *difference* of two multisets M and N is

$$M - N = (\Omega, \max(0, f - g)),$$

where $\max(0, f - g)(\omega) = \max(0, f(\omega) - g(\omega))$. ♢

Remark 2.22. Now that we have defined subtraction for multisets, we will use $\backslash$ for the difference of two normal sets to avoid confusion.

Notation 1: Multisets

Formal notation	$M = (\Omega, m : \Omega \to \mathbb{Z}_{\geq 0})$
Set notation	$A = [a, a, a, b, b]$
Graph notation	$A = \{(a, 3), (b, 2)\}$
Counting function	m_M
Element	$x \in M$
Cardinality (size)	$\lvert M \rvert$
Sub-multiset	$M \leq N$
Support	$\operatorname{supp}(M)$
Multiset in set	$M \lhd S$
Union, intersection	$M \cup N$ and $M \cap N$
Addition, subtraction	$M + N$ and $M - N$
Repeated operations	$\sum, \bigcup, \bigcap$

3 Graphs

In this section, I will explain what graphs are. There are a lot of properties of graphs that will be defined. Unless otherwise stated, all graphs after this section are assumed to be connected undirected multigraphs.

3.1 Undirected graphs

An undirected graph is like a network of nodes (the vertices) connected to each other by lines (edges). An example is shown in Figure 3.1.

Definition 3.1 (Undirected multigraph)**.** An *undirected multigraph* G is a tuple $G = (V, E)$ consisting of a set of *vertices* or *nodes* V and a multiset of edges E. An *edge* is a multiset $e \lhd V$ with size $|e| = 2$. Thus E is a multiset containing multisets with two vertices,

$$E \lhd \big\{ e \lhd V \big| |e| = 2 \big\} = \big\{ [u, v] \big| u, v \in V \big\}.$$

The sets of vertices and edges of a graph are also denoted $V(G)$ and $E(G)$. $\diamondsuit$

Remark 3.2. Since I will only use multigraphs in this thesis, I will just call them graphs.

Remark 3.3. A multiset containing two elements is also called an *unordered pair*. These are very similar to sets containing two elements, with the only difference that both elements may be equal. Therefore, we will write the edges of a graph using the usual set notation. This is also to avoid confusion with intervals and more general multisets. We will also write uv for an edge between u and v, to avoid several layers of nested braces.

$$\{u, v\} = \{v, u\} = uv = vu$$

Remark 3.4. The two elements of an edge (an unordered pair) are also called the *endpoints* of that edge. Two vertices are *neighbours* when there is at least one edge having those vertices as endpoints.

Now we introduce some notation for some special sub-multisets of edges.

Definition 3.5 (Edge multisets)**.** The multiset of edges from a vertex $u \in V(G)$ to $v \in V(G)$ is given by

$$E(u, v) = \big\{ \big(uv, m_E(uv) \big) \big\},$$

where m_E is the counting function for the multiset of edges of G.

The multiset of edges adjacent to a given vertex u is

$$E(u) = \big\{ (e, m_E(e)) \big| u \in e \in M \big\} \qquad \diamondsuit$$

In a similar way, we define the set of neighbours of a vertex.

Definition 3.6 (Neighbours)**.** The set $N(v)$ of neighbours of $v \in V(G)$ is given by

$$N(v) = \{ u \in V(G) | (v, u) \in E(G) \}. \qquad \diamondsuit$$

Intuitively, the degree of a vertex is the total number of edges incident to it, where edges that have both ends at the same vertex are counted twice.

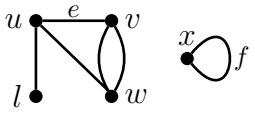


Figure 3.1: The black dots represent the *vertices* of the graph, and the lines between them the *edges*. Vertex x is only connected to itself by a *loop*, and therefore has *degree* 2. The leaf is l, because it is the only vertex that is connected to exactly one other vertex.

Definition 3.7 (Degree)**.** For a graph $G = (V, E)$, the *degree* of a vertex $u \in V$ is given by

$$\deg(u) = \sum_{\substack{(e, m_E(e)) \in E, \\ e = (V, m_e)}} m_E(e) \cdot m_e(u).$$

In each term we multiply the number of identical edge e with the number of times u occurs in e. This ensures that loops, which are defined below, are counted twice. ♢

Some types of nodes and edges have special names. When a node has degree 1 it is called a leaf.

Definition 3.8 (Leaf)**.** A vertex $u \in V$ in a graph G is called a *leaf* when $\deg(u) = 1$. ♢

Edges that have a single vertex as endpoint twice are called loops.

Definition 3.9 (Loops)**.** An edge $e = \{u, v\}$ is called a *loop* when $u = v$. ♢

Example 3.10. Suppose the graph G is given as

$$V(G) = \{u, v, w, l, x\},$$
$$E(G) = [uv, uw, ul, vw, vw, xx].$$

We may visualize this graph as a drawing as shown in Figure 3.1. The vertices of the graph are represented by dots. The edges are shown as lines connecting the dots corresponding to the endpoints of the edge. The edge $e = \{u, v\} = \{v, u\}$ connects vertices u and v, and is shown as a line connecting the dots labelled u and v.

Vertex l is a leaf because is has degree 1. The degree of x is 2, because the edge $f = \{x, x\}$ is a loop based at x. Because E is a multiset of edges, there may be multiple edges between vertices. This is the case for v and w.

Remark 3.11. It is important to remember that a drawing is only a visualization of a graph. This means that there can be very distinct drawings of the same graph. Thus, when two drawings differ, you can not conclude that the graphs they represent are different.

Now we define some more properties of graphs.

Definition 3.12 (Size)**.** The *size* of a graph $|G|$ is the number of vertices

$$|G| = |V(G)|.$$ ♢

Definition 3.13 (Finite graph)**.** A graph G is *finite* when both the number of vertices $|V(G)|$ and the number of edges $|E(G)|$ are finite. ♢

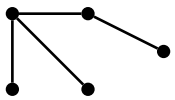


Figure 3.2: An example of a *tree*: a connected graph without cycles.

Remark 3.14. In fact, all graphs I have defined are finite, since we are only using finite multisets. If we were to allow infinite multisets, we could also obtain infinite graphs, but this thesis does not need them.

Graphs can be divided into several classes, depending on their properties. Here we give two of them.

Definition 3.15 (Loopless graph)**.** A graph is called *loopless* if it is contains no loops. ♢

Definition 3.16 (Simple graph)**.** A graph $G = (V, E)$ is *simple* if it is loopless and has no multiple edges. It has no multiple edges when for all edges $e \in E$ we have

$$m_E(e) \leq 1.$$ ♢

Now we define paths and cycles in graphs. A path is just a sequence of distinct vertices connected by edges. When a path ends in the vertex where is started, it is also called a cycle.

Definition 3.17 (Path)**.** A *path* is a sequence $u_0, \ldots, u_k$ of distinct vertices of $G = (V, E)$ such that for all $0 \leq i < k$ there is an edge from u_i to u_{i+1},

$$\{u_i, u_{i+1}\} \in E.$$

The vertices u_0 and u_k are the start and end of the path. We also say that u_0 and u_k are connected by this path. ♢

Definition 3.18 (Cycle)**.** A *cycle* is a path with $u_0 = u_k$, but otherwise distinct vertices. When $k = 2$, the cycle uses the same edge twice. This is only allowed when

$$m_E(\{u_0, u_1\}) = m_E(\{u_1, u_2\}) \geq 2,$$

to make sure all edges along the cycle are different too. ♢

Example 3.19. In Figure 3.1, the edges wu, uv, and one of the edges between v and w form a cycle.

Using these new concepts we define some more classes of graphs.

Definition 3.20 (Connected graph)**.** A *connected* graph is a graph in which there exists a path between every pair of vertices. ♢

Remark 3.21. From now on, we will assume all graphs are connected.

Definition 3.22 (Tree)**.** A *tree* is a connected graph without cycles. This happens precisely when there is exactly one path between each pair of vertices. ♢

Example 3.23. The graph in Figure 3.1 is not connected, because there is no path from x to any of the other vertices. Figure 3.2 shows an example of a tree.

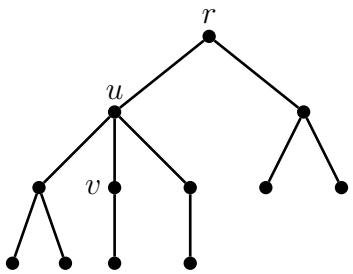


Figure 3.3: This is an example of a rooted tree. The root is r. u is one of the two children of r. The distance from v to r is 2, so v has depth 2.

Definition 3.24 (Rooted tree)**.** A *rooted tree* is a tree T together with a single special vertex r of T, which is called the root of T. ♢

Rooted trees have lots of interesting properties. Here, we define the depth of a vertex and the children of a vertex. An example is shown in **??**.

Definition 3.25 (Depth)**.** The *depth* of a vertex u in a rooted tree T is the number of edges in the unique path from the root r to u. The k^{th} *level* consists of all vertices with depth k. ♢

Definition 3.26 (Height)**.** The *height* of a tree is the maximum depth of the vertices in the tree. ♢

Definition 3.27 (Children)**.** The set of children of a vertex u with depth d consists of all neighbours of u with depth $d + 1$. ♢

Now we define subgraphs in order to define connected components.

Definition 3.28 (Subgraph)**.** A graph H is a subgraph of a graph G, written $H \subset G$, when

$$V(H) \subset V(G),$$
$$E(H) \leq E(G).$$

Because H is a graph itself, we furthermore have that $u \in V(H)$ for all $u \in e \in E(H)$. ♢

The induced subgraph of a given subset U of the vertices of G is the subgraph consisting of the vertices U and all edges between vertices in U.

Definition 3.29 (Induced subgraph)**.** Given a set of vertices $U \subset V(G)$, the induced subgraph $G[U]$ of G is defined as

$$V(G[U]) = U,$$
$$E(G[U]) = \{(e, m_E(e)) | e \in E(G),\ \text{supp}(e) \subset U\}. \qquad ♢$$

The connected component of a given vertex consists of all edges and vertices that are connected to the given vertex by a path.

Definition 3.30 (Connected component)**.** When v is a given vertex in a graph G, let U be the set of all vertices u for which there is a path that starts v and ends in u. Then, the connected component is the induced subgraph of U, $G[U]$.

For a given graph G and vertex $v \in V(G)$, $C(G, v)$ denotes the connected component of G containing v. ♢

Remark 3.31. A graph is connected, if and only if it has exactly one connected component.

Example 3.32. The graph shown in Figure 3.1 has two connected components. One component consists of the vertex x together with the edge f. The other connected component consists of all other vertices and edges of G.

Using this, we can define v-subgraphs of G in a similar way to the v-subtrees defined in [3]. Roughly, a v-subgraph is a connected component of $G\backslash\{v\}$, together with v.

Definition 3.33 (v-subgraph)**.** A *v-subgraph* of G is the induced subgraph of a connected component of G together with v. More precise, the v-subgraph containing $u \in V(G)\backslash\{v\}$, written $G_v(u)$, is given by

$$G_v(u) = G\Big[V\big(C(G\backslash\{v\}, u)\big) \cup \{v\}\Big].$$

When $G_v(u)$ is a tree, this is called a *v-subtree* of G. ♢

Remark 3.34. When we say v-subgraph, we will usually mean a v-subgraph that is *not* a v-subtree.

In a similar way, we define a uv-subtree of a tree T as the connected component of $T\backslash\{u, v\}$ that connects to both u and v. Because T is a tree, this component is unique.

Definition 3.35 (uv-subtree)**.** For a tree T and two different vertices u and v, the *uv-subtree* of T is defined as

$$T_{uv} = T'_v(u), \qquad \text{where} \qquad T' = T_u(v). \qquad ♢$$

Using connected components, we can also define subtrees of rooted trees.

Definition 3.36 (Rooted subtrees)**.** Given a rooted tree T with root r, the *subtree rooted at $u \in V(T)$* is the induced subgraph of the set of all vertices v for which u lies on the unique path from v to r.

The *subtrees of u* are all subtrees rooted at a child of u. ♢

We would also like a notion of subgraphs for rooted trees, where the root of the subgraph must be the root of the original graph. Because *subtree* is already used, we call this embedded trees, like in [7]. We first need the notion of graph isomorphisms.

Definition 3.37 (Graph isomorphism)**.** Two graphs G and H are isomorphic, written $G \simeq H$ when there exists a bijective function $\varphi : V(G) \to V(H)$ such that the number of edges between u and v in G is equal to the number of edges between $\varphi(u)$ and $\varphi(v)$ in H,

$$m_{E(G)}(uv) = m_{E(H)}(\varphi(u)\varphi(v)).$$

♢

Definition 3.38 (Embedded rooted trees)**.** A rooted tree A with root a is *embedded* in a rooted tree T with root r when A is isomorphic to a subgraph of A in such a way that the isomorphism maps a to r. ♢

Remark 3.39. We often consider functions between graphs that can act on both vertices and edges. For example, f may send vertices of G to vertices of H and similar for edges:

$$f : V(G) \to V(H), \qquad f : E(G) \to E(H).$$

Then, we often write $f : G \to H$ for both these functions at once. We define a function $f : G \to H$ by first defining the image of all vertices and then the image of all edges. When both the domain and codomain of a function are graphs, this should be read as two separate functions acting on the vertices respectively edges of the domain. Note that this is different from a function

$$f : V(G) \cup E(G) \to V(H) \cup E(H),$$

since that would allow edges to be send to vertices and the other way around. Functions which *do* map edges to vertices are considered in [1] for example, but we will not use them.

Notation 2: Undirected graphs

Graph	$G = (V, E)$
Undirected edge	$e = \{u, v\} = uv$
Vertices and edges of G	$V(G)$ and $E(G)$
Edges connecting u and v	$E(u, v)$
Edges incident to u	$E(u)$
Number of vertices	$\lvert G\rvert = \lvert V(G)\rvert$
Degree of vertex	$\deg(u)$
Subgraph	$G \subset H$
Induced subgraph	$G[U]$
Connected component of u	$C(G, u)$
v-subgraph containing u	$G_v(u)$
uv-subtree	T_{uv}

3.2 Operations on graphs

We define some useful shorthand notations for adding and removing vertices and edges to a graph. In order to define the removal of vertices, we will first define induced subgraphs.

Definition 3.40 (Modification of vertices)**.** Given a graph G and a set U of vertices, we write $G \cup U$ for the graph we get when we add all vertices of U to G,

$$V(G \cup U) = V(G) \cup U,$$
$$E(G \cup U) = E(G).$$

When $U \subset V(G)$, we can remove the vertices in U and all edges incident to them from G. This is written $G \backslash U$,

$$G \backslash U = G[V(G) \backslash U]. \qquad \diamondsuit$$

Definition 3.41 (Modification of edges)**.** When $F \lhd \{[u,v] | u,v \in V(G)\}$ is a multiset of edges between vertices of G, we write $G+F$ for the graph G together with all edges in F,

$$\begin{aligned} V(G+F) &= V(G), \\ E(G+F) &= E(G) + F. \end{aligned}$$

When $F \leq E(G)$ is a sub-multiset of the edges of G, we may remove those edges from G, resulting in $G-F$,

$$\begin{aligned} V(G-F) &= V(G), \\ E(G-F) &= E(G) - F. \end{aligned}$$ ♢

Instead of adding or removing edges, we may also contract edges. When we contract an edge $e = (u,v) \in E(G)$ in G, we identify the endpoints u and v of the edge by a single new vertex w and we remove all edges between u and v in G. This definition is a modification for multigraphs of the first definition in [10]. This paper also proves some intuitive lemmas about contractions in a formal way.

Definition 3.42 (Edge contraction)**.** Given a graph G and an edge $e = uv \in E(G)$, we define the edge contraction G/e of G by

$$\begin{aligned} V(G/e) &= V(G) - \{u,v\} + w_e, \\ E(G/e) &= \bigcup_{\substack{f \in E(G) \\ f \neq e}} (f/e, m(f)), \end{aligned}$$

where the contraction f/e of an edge f is defined as

$$f/e = \begin{cases} f & f \cap e = \emptyset \\ w_e + (f \cap e) & f \cap e \neq \emptyset. \end{cases}$$

Since the contracting of edges commutes, as shown in [10], we also define G/S as the result of contracting all edges in $S = \{e_1, \ldots, e_k\} \subset E(G)$,

$$G/S = G/e_1/\ldots/e_k.$$ ♢

Notation 3: Operations on graphs

Add vertices	$G \cup U$
Remove vertices	$G \backslash U$
Add or remove edges	$G \pm F$
Contract an edge e	G/e
Contract edges in S	G/S

3.3 Directed graphs

We can also consider *directed graphs*, where every edge has a given direction. These are defined analogously to undirected graphs, except that the edges are now *ordered pairs* of nodes, instead of unordered pairs.

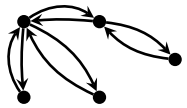


Figure 3.4: The *directed graph* $D(T)$ corresponding to the tree T in Figure 3.2. In this graph, we have replaced every undirected edge of T by two directed edges in opposite directions.

Definition 3.43 (Directed graph)**.** A *directed graph* G is a tuple $G = (V, E)$ consisting of a set of *vertices* V and a multiset of directed edges E. A *directed edge* is a pair $e \in V \times V$. Thus E is a multiset containing pairs of vertices of V,

$$E \lhd V \times V. \qquad \diamondsuit$$

Remark 3.44. Since directed edges are pairs of vertices, we write the edge from u to v as (u, v). Again, uv can be written for short when it is clear from the context that the edge is directed.

The definitions of the sets of edges between nodes needs to be modified too, because edges now have a direction.

Definition 3.45 (Edges multisets)**.** The multiset of edges from u to $v \in V(G)$ in a directed graph G is given by

$$E(u, v) = \big\{\!\big(uv, m_E(uv)\big)\!\big\}.$$

The set of edges *starting* in u is denoted $E(u)$ and defined by

$$E(u) = \big\{\!\big(e, m_E(e)\big)\big| e \in E, e \in \{u\} \times V\big\}. \qquad \diamondsuit$$

Because edges are directed, we may look at the reverse of an edge.

Definition 3.46 (Reverse)**.** The reverse $\overline{e}$ of an edge $e = (u, v)$ is given by

$$\overline{e} = (v, u)\,. \qquad \diamondsuit$$

Directed graphs often appear because undirected graphs are stored as directed graphs in most algorithms. Therefore, we now define the equivalent directed graph for a given undirected graph. In this graph, every undirected edge in the original graph is replaced by two directed edges in opposing directions.

Definition 3.47. The *equivalent directed graph* $D(G)$ of an undirected graph G is given by

$$\begin{aligned} V(D(G)) &= V(G), \\ E(D(G)) &= \sum_{uv \in E(G)} \{(uv, m_E(uv)), (vu, m_E(vu))\}. \end{aligned} \qquad \diamondsuit$$

Example 3.48. The directed graph of the tree in Figure 3.2 is shown in Figure 3.4.

Notation 4: Directed graphs

Equivalent directed graph	$D(G)$
Directed edge	$e = (u, v) = uv$
Edges from u to v	$E(u, v)$
Edges starting at u	$E(u)$
Reverse of edge e	$\overline{e}$

4 Refinements

A graph H is a *refinement* of G when it can be obtained from G by repeatedly adding leaves or subdividing edges a *finite* number of times. We first define two functions that describe these operations, and then define refinements rigorously.

The adding of leaves is shown in Figure 4.1. This is done by adding a new vertex l to G and connecting it to a given vertex $u \in V(G)$ via an undirected edge $\{u, l\}$. Together with this comes a natural inclusion ι of the original graph into the new graph. We need this inclusion to keep the structure of the graph, so that we can identify the vertices and edges of the original graph in the refinement.

Definition 4.1 (Adding leaves)**.** For a given vertex $u \in V(G)$ in a graph G, the *leaf addition tuple* is $\mathbf{L}_u = (L_u; \iota : G \to L_u(G))$. This tuple consists of a function L_u that adds a leaf to G and an inclusion ι of G into $L_u(G)$. The *leaf addition function* L_u is defined by

$$\begin{aligned} V(L_u(G)) &= V(G) \cup \{l\}, \\ E(L_u(G)) &= E(G) + [\{u, l\}]. \end{aligned}$$

The inclusion ι is defined by

$$\begin{aligned} \iota : G &\to L_u(G) \\ u &\mapsto u, \\ e &\mapsto e, \end{aligned}$$

and sends vertices and edges in G to the corresponding vertices respectively edges in $L_u(G)$.

When $S \lhd V(G)$ is a finite multiset of vertices of G, we write $\mathbf{L}_S$ for the tuple that adds $m_S(u)$ leaves to every vertex u. More precise, when $S = [u_1, \ldots, u_k]$ we have

$$L_S = L_{u_k} \circ \cdots \circ L_{u_1},$$

and

$$\iota = \iota_{u_k} \circ \cdots \circ \iota_{u_1}.$$

We will also write this as

$$\mathbf{L}_S = \mathbf{L}_{u_k} \circ \cdots \circ \mathbf{L}_{u_1}. \qquad \diamondsuit$$

The subdividing of an edge is shown in Figure 4.2. We replace an edge $e = \{u, v\}$ by a new vertex s connected to both old neighbours via new edges $e_u = \{u, s\}$ and $e_v = \{v, s\}$. Again we also have an inclusion of G into the new graph H. This time we have to be more careful, because the edge e does not exist any more in H. Therefore, we restrict the domain of ι to all of G except for this edge.

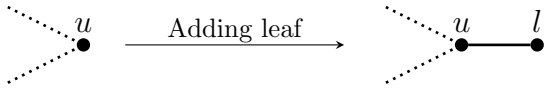


Figure 4.1: We add a leaf l to vertex u. The dotted edges indicate that the vertex u may be connected to other vertices not shown in the picture.

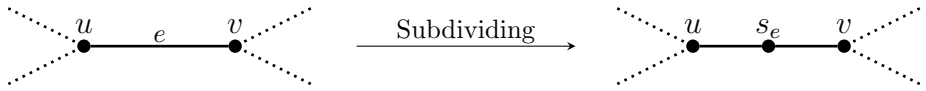


Figure 4.2: Here, we subdivide edge $e = \{u, v\}$, resulting in a new vertex s_e and two new edges connecting s_e to u and v. The (v, y) edge on the left has index 2.

Definition 4.2 (Subdividing edges)**.** For an edge $e = \{u, v\} \in E(G)$, the *subdivision tuple* is the pair of functions $\mathbf{S}_e = (S_e; \iota : G - [e] \to S_e(G))$. The function S_e subdivides edge e in two parts and ι is an inclusion of G minus the edge e into $S_e(G)$. The new vertex will be called s_e. The image of $S_e(G)$ is defined as

$$V(S_e(G)) = V(G) \cup \{s_e\},$$
$$V(S_e(G)) = E(G) - [uv] + [us_e, vs_e].$$

The domain of ι is all of G except for the edge e, because it does not exist in $S_e(G)$. This restriction will often be implicit and we will just write $\iota : G \to S_e(G)$.

$$\begin{aligned} \iota : G &\to S_e(G) \\ u &\mapsto u, \\ e &\mapsto e. \end{aligned}$$

When S is a finite multiset of edges of G, we may write $\mathbf{S}_S$ for the consecutive subdividing of the edges in S. More precise, when $S = [e_1, \ldots, e_k]$ we have

$$S_S = S_{e_k} \circ \cdots \circ S_{e_1},$$

and

$$\iota = \iota_{e_k} \circ \cdots \circ \iota_{e_1}.$$

The domain of this new inclusion is the intersection of the domains of all ι_{e_i}. This is just $G - (S, \mathbf{1}_S)$, where $(S, \mathbf{1}_S)$ is the multiset corresponding to the set S of edges we remove.

Again, we shorten the composition of S_{e_i} and ι_{e_i} to

$$\mathbf{S}_S = \mathbf{S}_{e_k} \circ \cdots \circ \mathbf{S}_{e_1}. \qquad \diamondsuit$$

Remark 4.3. As is clear from the definition of $\mathbf{S}_e$, we only remove the edge uv once from G. When the multiplicity of uv in G is greater then one, we only subdivide one occurrence of the edge. Thus, the new multiplicity of uv is $m_E(uv) - 1$. Since we consider all edges between u and v to be the same, it does not matter which one we choose to subdivide when viewing a graph as a drawing.

Remark 4.4. In Definition 4.1 above, we can add multiple leaves to a single vertex because the leaves do not interact with each other. However, after subdividing an edge, it does not exist any more. Therefore, we can not subdivide a single edge twice in one pass. When and edge $e = uv$ occurs twice in S, we do not subdivide one edge from u to v twice, but instead subdivide two edges from u to v once.

For completeness, we also define a function corresponding to edge contractions.

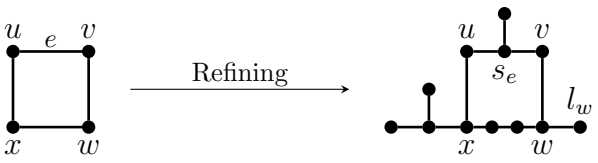


Figure 4.3: This is an example of a refinement. We subdivide edge e and add a leaf to the new node s_e. Furthermore, we subdivide $\{x, w\}$ twice, and add some more leaves recursively.

Definition 4.5. Given a graph G and an edge $e \in E(G)$, the *edge contraction tuple* is defined as $\mathbf{C}_e = (G/e; \iota : G \to G/e)$. Here, ι is the inclusion given by

$$\begin{aligned} \iota : V(G) &\to V(G/e) \\ \iota(u) &= \begin{cases} u & u \notin e \\ w_e & u \in e \end{cases} \\ \iota : E(G)\backslash\{(e, m_{E(G)}(e))\} &\to E(G/e) \\ \iota(f) &= f/e. \end{aligned}$$

As before, we also define $\mathbf{C}_S = (G/S; \iota : G \to G/S)$ for the function corresponding to the contraction of all edges in S. ♢

Now, a graph H is a *refinement* of G when we can make it from G by repeatedly adding leaves or subdividing edges.

Definition 4.6 (Refinements)**.** A graph H is a *refinement* of G, written $G \preceq H$, if and only if there is a finite sequence of graphs

$$G = G_0, G_1, \ldots, G_k \simeq H$$

such that for each $0 \leq i < k$ we have that

- either $G_{i+1} = L_u(G_i)$ for some vertex $u \in V(G_i)$,
- or $G_{i+1} = S_e(G_i)$ for an edge $e \in E(G_i)$.

Note that in the last step, we only need H to be isomorphic to G_k.

We write $\mathbf{R}_i = (R_i; \iota_i : G_i \to G_{i+1})$ for the i^{th} leaf addition or subdivision tuple. Furthermore, $\mathbf{R} = (R; \iota : G \to H)$ is the tuple associated with the complete refinement,

$$\mathbf{R} = \mathbf{R}_k \circ \cdots \circ \mathbf{R}_1. \qquad ♢$$

An example of a refinement can be seen in Figure 4.3.

Remark 4.7. We will often write $\mathbf{R} : G \to H$ instead of $\iota_{\mathbf{R}} : G \to H$. This allows us to draw commutative diagrams on graphs with refinements on the arrows. Hence, the bold functions should be read as the corresponding inclusions.

We will now state a lemma on the commutative behaviour of refinements.

Lemma 4.8 (Commutative refinements)**.** *Adding leaves u and v to a graph G can be done in any order when both u and v are vertices of G. Thus, $\mathbf{L}_u$ and $\mathbf{L}_v$ commute when v is not a leaf based at u or the other way around.*

Adding a leaf l to u commutes with the subdividing of an edge e when e is not $\{u, l\}$ and u is not s_e, the vertex created in the subdivision.

Finally, the subdividing of two different edges e and f commutes precisely when e is not an edge created by $\mathbf{S}_f$ and f is not an edge created by $\mathbf{S}_e$.

Remark 4.9. To summarize, when $\mathbf{R}_2 \circ \mathbf{R}_1$ is a refinement of G, $\mathbf{R}_1$ and $\mathbf{R}_2$ commute when $\mathbf{R}_2$ is also a well-defined refinement of G.

Currently, the refinement function $\mathbf{R}$ on a graph G is defined for all vertices of G and a subset of the edges of G. When an edge is subdivided once or more and leaves are added to the new vertices, we would like to define the image of an edge e (excluding endpoints) under the refinement as the subgraph of $\mathbf{R}(G)$ that originated in e. In fact, we can do this more generally and define a function that gives the image of a subgraph of G under $\mathbf{R}$.

This definition corresponds to the definition of the restricted refinement in Definition 5.2 in [6].

Definition 4.10 (Image of subgraph)**.** When we are given a subgraph A of G, we write $\mathbf{R}^G(A)$ for the image of A under the refinement $\mathbf{R}$ of G. We define this for leaf addition as

$$\mathbf{L}_u^G(A) = \begin{cases} \mathbf{L}_u(A) & u \in V(A) \\ A & u \notin V(A). \end{cases}$$

For subdivisions, we define

$$\mathbf{S}_e^G(A) = \begin{cases} \mathbf{S}_e(A) & e \in E(A) \\ A & e \notin E(A). \end{cases}$$

For an arbitrary refinement $\mathbf{R}$ with

$$\mathbf{R} = \mathbf{R}_k \circ \cdots \circ \mathbf{R}_1$$

we have

$$\mathbf{R}^G = \mathbf{R}_k^G \circ \cdots \circ \mathbf{R}_1^G. \qquad \diamondsuit$$

Remark 4.11. We are mostly interested in $\mathbf{R}^G(e)$, for some edge $e \in E(G)$. Here, A is not a real subgraph, because the endpoints of e are not included. This is not a problem however, because A does not have to be a subgraph of G.

We now define the set of refinements $\mathrm{Rf}(G)$ of a graph G.

Definition 4.12 (Set of refinements)**.** The set $\mathrm{Rf}(G)$ of refinements of G is defined as

$$\mathrm{Rf}(G) = \{H | G \preceq H\}. \qquad \diamondsuit$$

Remark 4.13. We should be calling $\mathrm{Rf}(G)$ the *class* of refinements of G, but I will just treat it as set instead.

The relation $\preceq$ also induces a poset on the set of all graphs.

Definition 4.14 (Poset)**.** The set of all graphs is a poset with respect to the binary relation $\preceq$. $\diamondsuit$

Proof. Reflexivity is trivial. Transitivity follows from the fact that we may compose refinements. Antisymmetry holds too, because $G \preceq H$ implies that the number of vertices of G is not larger than the number of vertices of H. The same holds for edges. Because we have $H \preceq G$ as well, they must have the same number of vertices and edges. Since strict refinements always increase the number vertices, we must have $G = H$. □

We may now call two graphs refinement-equivalent when they are in the same component of this poset.

Definition 4.15 (Refinement-equivalent)**.** We call two graphs G and H *refinement-equivalent*, or just *equivalent*, if there is a sequence of graphs $G = G_0, G_1, \ldots, G_k = H$ such that for each $0 \leq i < k$, either $G_i \preceq G_{i+1}$ or $G_i \succeq G_{i+1}$ holds. ♢

Remark 4.16. We will say that two graphs have the same *structure* when they belong to the same equivalence class. This allows us to talk about operations on a graph that do not change the structure of the graph.

4.1 Stable graphs

It turns out that every equivalence class of refinement-equivalent graphs has a single smallest element in the sense that every other graph in the equivalence class is a refinement of this graph. In this section, I give my own proof of this theorem. The definition of a stable graph is based on the one given in [6, Definition 3.4].

Definition 4.17 (Stable graph)**.** A *stable graph* is a connected graph with

- either a single vertex,
- or no vertices of degree less than 3. ♢

Theorem 4.18 (Unique stable graph)**.** *Every refinement-equivalent class of graphs contains exactly one stable graph, and every graph in the equivalence class is a refinement of this stable graph.*

We start with the definition of the stable degree. This counts the number of edges from a vertex v to v-subgraphs of G that contain a cycle.

Definition 4.19 (Stable degree)**.** The *stable degree* of a vertex $v \in V(G)$ is defined as

$$\operatorname{sdeg}(v) = \left|\left\{e \middle| e = uv, G_v(u) \text{ is not a tree}\right\}\right|$$

♢

Remark 4.20. Below, we show that the stable degree is constant under refinements. This justifies the name *stable*.

Lemma 4.21. *The stable degree is constant under refinements.*

Proof. Let $v \in V(G)$ be given. It suffices to show that the stable degree of v does not change we we add a leaf to G or subdivide an edge of G.

Case 1: leaf addition. If we add a leaf l to v, $(L_v(G))_v(l)$ is a tree, and hence vl is not counted. If we add l to a different vertex u, $\big(L_u(G)\big)_v(u)$ is a tree if and only if $G_v(u)$ is a tree. Hence, this does not change the stable degree of v.

(a) **(b)**

Figure 4.4: A graph with two cycles will always contain at least two vertices with stable degree 3, which are shown as red in these graphs. Furthermore, any graph with a vertex with stable degree at least three contains as least 2 cycles.

Case 2: edge subdivision. If we subdivide an edge $e = xy$ in G, we will never create a new cycle. Hence, every $G_v(u)$ is a tree after the subdivision if and only if it was a tree before the subdivision.

We conclude that the stable degree is constant under refinements. □

Definition 4.22 (Stabilisation)**.** A *stabilisation* of a connected graph G is a stable graph S such that G is a refinement of S. ♢

Theorem 4.23 (Unique stabilisation)**.** *Every graph has a unique stabilisation.*

Proof. We start with a constructive prove of the existence of a stabilisation for every graph G. Start with $S = G$. While S has at least two vertices, and at least one vertex v with degree less than 3, we contract this vertex to a neighbour. We can reverse this operation by adding a leaf or subdividing an edge. Hence, G will always be a refinement of S. We end with a stabilisation S of G.

For the uniqueness, we first consider the cases where all vertices of G have a stable degree less than 3. In this case, G has at most 1 cycle, because otherwise, there would be a vertex with stable degree at least 3. This is shown in Figure 4.4.

G is a tree. The trivial graph consisting of a single vertex is a stabilisation of G. Any stabilisation S of G with at least two vertices must have a leaf, and hence would not be stable. Furthermore, a graph with one vertex and at least one edge would have a cycle, so that G could not be a refinement of S. We conclude that trees have a single stabilisation.

G has 1 cycle. In this case, the graph S consisting of a single vertex with a loop is a stabilisation of G, because G is a refinement of this graph. Any other graph S will have at least two cycles or at least one vertex of degree less than 3, so the given graph is the only stabilisation of G.

G has a vertex with stable degree at least 3. We will prove that S contains all vertices of G with stable degree at least 3 and none of the other vertices. If we add a new vertex to S by a leaf addition, the stable degree of the new vertex will be 1. When we subdivide an edge, the new vertex has stable degree at most 2. Because both these degrees are less than 3, S must contain all vertices of G with stable degree at least 3. It remains to show that S contains no vertices with stable degree less than 3. Because S is stable, we know that all vertices have at least degree 3. A subgraph $S_v(u)$ is never a tree, because S has no leaves. This implies that all vertices of S have stable degree at least 3.

From this, we conclude that S contains precisely the vertices of G with stable degree at least 3. We still need to prove that the edges of S can be uniquely determined from G as well.

Let $e(u, v)$ denote the number of paths in G between two vertices $u, v \in V(G)$ with stable degree at least 3, such that no other vertices with stable degree at least 3 lie on these paths. It is easily seen that this quantity is constant under refinements, because we can not add or remove such vertices. This implies that $e(u, v)$ must be equal in S and G. Because every vertex in S has stable degree at least 3, $e(u, v)$ counts the number of edges from u to v in S. This implies that S can be uniquely determined from G. □

Now we prove that every equivalence class contains a unique stable graph.

Proof of Theorem 4.18. It is sufficient to prove that every pair of stable graphs is in a different equivalence class, because Theorem 4.23 implies the existence of at least one stable graph in every equivalence class.

We proceed with a proof by contradiction. Suppose S and S' are different but equivalent stable graphs. Then, there exist graphs $G_0, \ldots, G_k$ with $G_0 = S$, $G_k = S'$ and $G_i \preceq G_{i+1}$ or $G_i \succeq G_{i+1}$. Each of these graphs G_i has a unique stabilisation S_i. Suppose $G_i \preceq G_{i+1}$. Then, S_i is also a stabilisation of G_{i+1}. Thus, $S_i = S_{i+1}$. The same holds when $G_i \succeq G_{i+1}$. This implies that $S = S'$. Since we assumed that $S \neq S'$, this is a contradiction. We conclude that every equivalence contains a unique stable graph. □

We have introduced the following notations in this chapter.

Notation 5: Refinements

Leaf addition	$\mathbf{L}_u = (L_u; \iota : G \to L_u(G))$
Subdivision	$\mathbf{S}_e = (S_e; \iota : G - [e] \to S_e(G))$
Edge contraction	$\mathbf{C}_e = (G/e; \iota : G \to G/e)$
Refinement	$\mathbf{R} = (R; \iota : G \to R(G))$
Image of subgraph	$\mathbf{R}^G(A)$
Refinement relation	$G \preceq \mathbf{R}(G)$
Set of refinements	$\mathrm{Rf}(G)$
Stable degree	$\mathrm{sdeg}(u)$

5 Finite harmonic morphisms

In this section I present *finite harmonic morphisms* and the stable gonality, as first defined in chapter 3 of [6]. My definitions will be very similar to those in [6], except for some changes in wording and some new notations.

A *finite morphism* is a map from a graph to another graph sending vertices to vertices and edges to edges. Such a morphism is *harmonic* when it obeys some additional restrictions. We will see that because of these restrictions, a finite harmonic morphism has a lot of interesting properties.

Definition 5.1 (Finite harmonic morphism)**.** Let G and H be two loopless graphs. A *finite morphism* between G and H, denoted $\varphi : G \to H$ is a triple of maps

$$\begin{aligned} \varphi &: V(G) \to V(H), \\ \varphi &: E(G) \to E(H), \\ r_\varphi &: E(G) \to \mathbb{N} \end{aligned}$$

such that every edge $\{u, v\} \in E(G)$, maps to another edge $\{\varphi(u), \varphi(v)\} \in E(H)$ in H. The last map, $r_\varphi(e)$, is called the *index of* φ *at* e. When φ is clear from the context, we will usually just write *the index of* e and $r(e)$.

We define the *index* of a vertex $v \in V(G)$ at an edge $e' \in E(\varphi(v))$ neighbouring the image of v as

$$m_{\varphi,e'}(v) = \sum_{e \in E(v),\ \varphi(e)=e'} r_\varphi(e).$$

A finite morphism is called *harmonic* if $m_{\varphi,e'}(v)$ is independent of the edge e' in $E(\varphi(v))$ we choose. When φ is harmonic, we drop the e' from the subscript and write m_φ instead. When there are no edges adjacent to v', we define $m_\varphi(v) = 1$. The equation above is also called the *harmonic property*.

Lemma 5.5 below will show that for a finite harmonic morphism φ to a connected loopless graph H, the following number is independent of $v' \in V(H)$ or $e' \in E(H)$:

$$\deg \varphi = \sum_{v \in \varphi^{-1}(v')} m_\varphi(v) = \sum_{e \in \varphi^{-1}(e')} r_\varphi(e).$$

This is called the *degree* of φ. ♢

Remark 5.2. The degree of a finite harmonic morphism is only defined when the image of φ is connected. In practice, we will only consider finite harmonic morphisms to trees, so this is not a problem.

Definition 5.3 (Stable gonality, [6, Definition 3.7])**.** A connected graph G that has a refinement H with a degree d harmonic morphism $\varphi : H \to T$ to a tree T is called *stably d-gonal*. The *stable gonality* of a graph G is defined as

$$\operatorname{sgon}(G) = \min\{\deg \varphi | \varphi : H \to T,\ H \in \operatorname{Rf}(G),\ T \text{ a tree}\}$$

where φ is a finite harmonic morphism from a refinement H of G to a tree T. Note that, although not explicitly written down, we also have to consider all possible r_φ. ♢

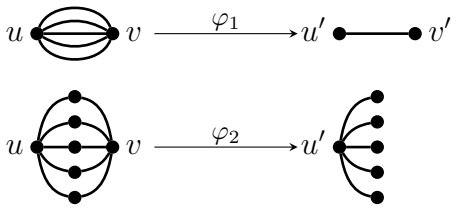


Figure 5.1: The first image shows a finite harmonic morphism from the banana graph B_5 to a single edge. The degree of this morphism is 5, because the single edge in the resulting graph has 5 edges mapping to it.
When we subdivide each edge in B_5 once, we obtain a finite harmonic morphism with degree 2. Since B_5 is not a tree, the stable gonality is at least 2, so it is 2.

Remark 5.4. The stable gonality is even defined for graphs containing loops, because we can refine these loops away.

Now, we will prove that the degree of a finite harmonic morphism is well defined. This lemma was already proven in [1], but I will present my own proof.

Lemma 5.5 (Degree)**.** *For a finite harmonic morphism $\varphi : G \to H$ to a connected graph H, we have that*

$$\deg \varphi = \sum_{v \in \varphi^{-1}(v')} m_\varphi(v) = \sum_{e \in \varphi^{-1}(e')} r_\varphi(e) \tag{5.1}$$

is independent of $v' \in V(H)$ or $e' \in E(H)$.

Proof. Since G and H are loopless, two neighbouring vertices in G never map to the same vertex in H, as that would imply that there is a loop in H. Thus, an edge $e \in E(G)$ mapping to $e' = \{u', v'\} \in E(H)$ will be adjacent to exactly one vertex mapping to u'. Therefore, we can partition the set $\varphi^{-1}(e')$ based on the preimage of one of the endpoints of e'.

$$\sum_{e \in \varphi^{-1}(e')} r_\varphi(e) = \sum_{u \in \varphi^{-1}(u')} \sum_{\substack{e \in E(u), \\ \varphi(e) = e'}} r_\varphi(e) = \sum_{u \in \varphi^{-1}(u')} m_\varphi(u)$$

Analogously, we could also have chosen the other end v' of e' to induce the partitioning. Therefore, we have

$$\sum_{u \in \varphi^{-1}(u')} m_\varphi(u) = \sum_{e \in \varphi^{-1}(e')} r_\varphi(e) = \sum_{v \in \varphi^{-1}(v')} m_\varphi(v).$$

This implies that the quantity $\sum_{v \in \varphi(v')} m_\varphi(v)$ is equal for neighbouring vertices in H. Since H is connected and every edge in H is adjacent to at least one vertex of H, the degree must be equal for all vertices of H, and hence for all edges in H. □

Remark 5.6. The proof I have given above only works if both G and H are loopless graphs. Since we are only interested in finite harmonic morphisms to trees, this is not a problem.

One property of finite harmonic morphisms is that we may subdivide an edge in H and all edges of G mapping to it without changing its properties. For a loop based

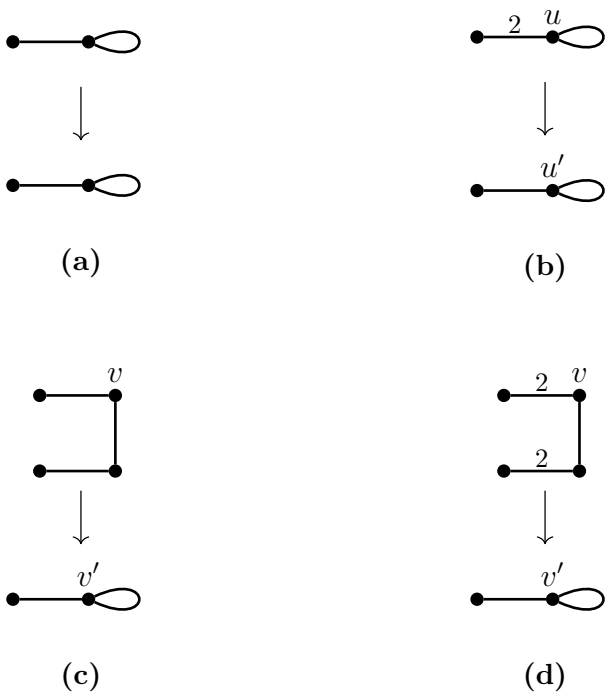


Figure 5.2: These are some examples of morphism that we might consider as finite harmonic morphisms if we were to allow loops. Only (a) satisfies the property that it is still a valid finite harmonic morphism after we refine the loop and all edges mapping to it. This is explained in Remark 5.6.

at u, this would create two edges from u to a new vertex l. If we look at the four candidates for finite harmonic morphisms shown in Figure 5.2, only Figure 5.2(a) satisfies this requirement. Case (b) fails because after subdividing the loops, u has index 2 to the left, but only one edge of index 1 maps to the new edges $u'l'$. Cases (c) and (d) fail because v' has degree 3 after subdividing the loop, while v still has degree 2. Thus, there is no way the edges surrounding v can cover all edges adjacent to v'.

We conclude that the only time when loops are possible in G and H is when all loops in G map to a loop in H and nonloops in G never map to loops in H. But then, we can just remove all loops from G and H, and the restriction of φ to the remainder of G would still be a finite harmonic morphism to the remainder of H.

The origin of these problems is that it is more intuitive to refine loops to a single edge to a new leaf, instead of two edges to a new vertex. To save ourselves from a lot of unintuitive edge cases, we disallow loops completely.

Remark 5.7. When all edges have index 1, the degree of a finite harmonic morphism is just the number of edges mapping to any edge $e' \in E(H)$, that is, $|\varphi^{-1}(e')| = |\{e : \varphi(e) = e'\}|$. The index $m_\varphi(v)$ of a vertex v is the number of times the edges in $E(V)$ cover the edges in $E(\varphi(v))$. The degree of φ is the sum of $m_\varphi(v)$ of all vertices mapping to a given $v' \in V(H)$.

Lemma 5.8. *The stable gonality of a tree G is* 1*, because we can use $H = T = G$, with the identity refinement and finite harmonic morphism which assigns index* 1 *to all edges. Trees are the only graphs with stable gonality* 1*, because a cycle in G will always cover some edge in T twice.*

Example 5.9. The banana graph B_n consists of two vertices u and v, both connected to n other vertices. The graph B_5 is shown in Figure 5.1. As shown in the picture, there is a finite harmonic morphism from B_5 to a tree with degree 5. By subdividing each edge once, we obtain a refinement of B_5. This new graph can be mapped to the tree shown on the bottom right. The degree of this new finite harmonic morphism is 2, because each edge is covered twice. Since B_5 is not a tree, there is no finite harmonic morphism from a refinement of B_5 to a tree with degree 1. Therefore, the stable gonality of B_5 is 2.

Often, it is useful to talk about all vertices of G that map to the same vertex of H. To make this easier, we now define an equivalence relation on G where two vertices are equivalent when they have the same image under φ.

Definition 5.10. Given a finite harmonic morphism $\varphi : G \to H$, we have an induced equivalence relation $\sim$ on the vertices of G defined by:

$$u \sim v \iff \varphi(u) = \varphi(v).$$

It is easy to check that this is indeed an equivalence relation. We write $[u] = \varphi^{-1}(\varphi(u))$ for the equivalence class of a vertex u in G.

Using the inclusion ι of $V(G)$ into $V(\widetilde{G})$ for some refinement $\widetilde{G}$ of G, we can partition the vertices of $\widetilde{G}$ that are in the image of ι by

$$\iota(u) \sim \iota(v) \iff u \sim v. \qquad \diamondsuit$$

In the algorithm I will describe, we will need the following lemma, which is Lemma 5.4 (i) from [6].

Lemma 5.11 (Refinement of finite harmonic morphism)**.** *For a given finite harmonic morphism $\varphi : H \to T$, and a refinement $\widetilde{T}$ of T, there exist a finite harmonic morphism $\tilde{\varphi} : \widetilde{H} \to \widetilde{T}$ for a refinement $\widetilde{H}$ of H with the same degree as φ.*

Proof. Suppose that $\mathbf{R} = \mathbf{R}_k \circ \cdots \circ \mathbf{R}_1$ is the refinement from T to $\widetilde{T}$. We will proceed by induction on k. When $k = 0$, $\widetilde{T} = T$ and we are done. Otherwise, let $T' = \mathbf{R}_{k-1} \circ \cdots \circ \mathbf{R}_1(T)$ be the refinement of T before applying $\mathbf{R}_k$, and H' and φ' the corresponding refinement of H and morphism. We know that $\mathbf{R}_k = \mathbf{L}_{u'}$ or $\mathbf{R}_k = \mathbf{S}_{e'}$ for some vertex u' respectively edge e' in T'.

Case 1: $\mathbf{R}_k = \mathbf{L}_{u'}$. We construct $\widetilde{H}$ by adding a leaf l_u to every vertex in $\varphi'^{-1}(u')$. We map all new leaves to $l_{u'}$, the new leaf in T', and the index of every new edge ul_u is given by $m_{\varphi'}(u)$, so that $\tilde{\varphi}$ is a harmonic morphism again.

Case 2: $\mathbf{R}_k = \mathbf{S}_{e'}$. Now, we subdivide every edge $e = uv$ in φ'^{-1} with a new vertex u_e mapping to e'. The indices of all new edges uu_e and vu_e are given by $r_{\varphi'}(e)$. Then, the harmonic property of $\tilde{\varphi}$ is satisfied in all u_e.

From the construction, it is obvious that $\deg(\tilde{\varphi}) = \deg(\varphi)$. □

Notation 6: Harmonic morphisms

Harmonic morphism	$\varphi : G \to H$
Index of φ of an edge	$r_\varphi(e)$
Index of a vertex	$m_\varphi(v)$
Degree of a morphism	$\deg(\varphi)$
Stable gonality	$\operatorname{sgon}(G)$
Equivalence class of a vertex	$[u] = \varphi^{-1}(\varphi(u))$

6 Properties of the stable gonality

The stable gonality has a lot of interesting properties. It is useful to take a look at them, because we might be able to use them to calculate the stable gonality.

The main result of this section will be that the stable gonality is invariant under refinements. We start by proving that it does not decrease under refinements, and next we show that it does not increase as well. For this, we proof that the stable gonality remains the same for both allowed operations to make refinements.

First we show that the stable gonality increases under refinements.

Lemma 6.1 (Refining)**.** *For a refinement H of G, we have* $\operatorname{sgon}(H) \geq \operatorname{sgon}(G)$.

Proof. When H is a refinement of G, it follows from the definition of refinements that $\operatorname{Rf}(H) \subset \operatorname{Rf}(G)$. Therefore

$$\begin{aligned}\operatorname{sgon}(H) &= \min\{\deg\varphi | \varphi : H \to T, H \in \operatorname{Rf}(H)\} \\ &\geq \min\{\deg\varphi | \varphi : H \to T, H \in \operatorname{Rf}(G)\} = \operatorname{sgon}(G).\end{aligned}$$ □

Now we know that the stable gonality does not decrease when we take refinements, it would be nice to prove it does not increase as well. For that, we first show that it does not increase when we add leaves.

Lemma 6.2 (Leaf addition)**.** *Adding leaves to a graph does not increase the stable gonality of the graph. Thus, for $u \in V(G)$, we have*

$$\operatorname{sgon}(L_u(G)) \leq \operatorname{sgon}(G).$$

Proof. Write $\widetilde{G}$ for $L_u(G)$. To prove that $\operatorname{sgon}(\widetilde{G}) \leq \operatorname{sgon}(G)$, we will show that there exists a finite harmonic morphism from a refinement of $\widetilde{G}$ to a tree with degree $\operatorname{sgon}(G)$.

Let H be a refinement of G such that there is a finite harmonic morphism φ from H to a tree T with $\deg\varphi = \operatorname{sgon} G$. For the given vertex $u \in V(G)$, we define $\widetilde{H} = L_{[u]}(H)$ and $\widetilde{T} = L_{\varphi(u)}(T)$. These are shown in Figure 6.1. Figure 6.2 shows the same diagram for explicit graphs G, H and T. We will show that we can define $\widetilde{\mathbf{R}}$ and $\tilde{\varphi}$ in such a way that this diagram commutes and that $\deg\tilde{\varphi} = \deg\varphi$.

First take a look at $\widetilde{\mathbf{R}}$. Since $u \in [u] \subset H$, $\mathbf{L}_{[u]}$ adds a leaf $l_u \in \widetilde{H}$ to $u \in H$. Hence, we can define

$$\widetilde{\mathbf{R}} = \mathbf{L}_{[u]\setminus\{u\}} \circ \mathbf{R}.$$

$$\begin{array}{ccccc} G & \xrightarrow{\mathbf{R}} & H & \xrightarrow{\varphi} & T \\ \downarrow{\scriptstyle \mathbf{L}_u} & & \downarrow{\scriptstyle \mathbf{L}_{[u]}} & & \downarrow{\scriptstyle \mathbf{L}_{\varphi(u)}} \\ \widetilde{G} & \xrightarrow{\widetilde{\mathbf{R}}} & \widetilde{H} & \xrightarrow{\tilde{\varphi}} & \widetilde{T} \end{array}$$

Figure 6.1: The commutative diagram we use in the proof of Lemma 6.2. The lower three graphs are defined in such a way that there exists a refinement $\widetilde{\mathbf{R}}$ and finite harmonic morphism $\tilde{\varphi}$ such that the diagram commutes.

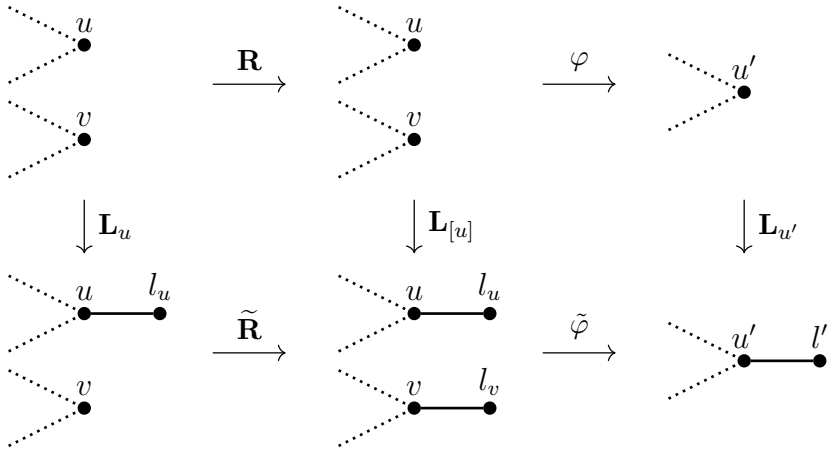


Figure 6.2: This is the same diagram as figure Figure 6.1, but with explicit graphs. The dotted lines indicate that the vertices may be connected to other parts of the graphs. We have $[u] = \{u, v\}$ and $\varphi(u) = \varphi(v) = u'$.

Because the domain of $\widetilde{\mathbf{R}}$ is $\widetilde{G}$ and not G, we extend the inclusion of $\widetilde{\mathbf{R}}$ by

$$\iota_{\widetilde{\mathbf{R}}}(l_u) = l_u, \qquad \iota_{\widetilde{\mathbf{R}}}(\{u, l_u\}) = \{u, l_u\}\,.$$

Because the addition of leaves to vertices of G commutes with other operations on refinements of G, we immediately have

$$\widetilde{\mathbf{R}} \circ \mathbf{L}_u = \mathbf{L}_{[u]\setminus\{u\}} \circ \mathbf{R} \circ \mathbf{L}_u = \mathbf{L}_{[u]\setminus\{u\}} \circ \mathbf{L}_u \circ \mathbf{R} = \mathbf{L}_{[u]} \circ \mathbf{R},$$

and thus the left square in the diagram commutes.

Now that we know that $\widetilde{H}$ is indeed a refinement of $\widetilde{G}$, we will define $\tilde{\varphi}$. It will be identical to φ on $H \subset \widetilde{H}$, and we extend it by sending all new leaves l_v with $v \in [u]$ to the newly added leaf l'. The corresponding edges $\{v, l_v\}$ are sent to the edges $\{u', l'\}$,

$$\begin{aligned} \tilde{\varphi}|_H &= \varphi, \\ \tilde{\varphi}(l_v) &= l', \\ \tilde{\varphi}(v l_v) &= u' l'. \end{aligned}$$

We also have to define the index of all new edges $\{v, l_v\}$,

$$r_{\tilde{\varphi}}(v l_v) = m_\varphi(v).$$

Since $\tilde{\varphi}|_H = \varphi$, it is clear that the right cycle in the diagram commutes too.

Now, the only thing left to show is that $\tilde{\varphi}$ is indeed a finite harmonic morphism. Thus, we have to show that $m_{\tilde{\varphi},e'}(v)$ is independent of e' for the vertices v in $[u] \subset \widetilde{H}$ and the new leaves l_v. In particular, we have to show that $m_{\tilde{\varphi},\{u',l'\}}(v)$ is equal to $m_\varphi(v)$, because $e' = \{u', l'\}$ is the only new edge surrounding $\tilde{\varphi}(v)$. Since $\{v, l_v\}$ is the only edge in $E(v)$ mapping to e', we have

$$m_{\tilde{\varphi},\{u',l'\}}(v) = \sum_{e \in E(v),\ \tilde{\varphi}(e) = e'} r_{\tilde{\varphi}}(e) = r_{\tilde{\varphi}}(\{v, l_v\}) = m_\varphi(v).$$

Hence, we have that $m_{\tilde{\varphi},e'}(v)$ is independent of e' and equal to $m_\varphi(v)$.

For the new leaves l_v, there is only one edge surrounding the image l'. Thus, the requirement that $m_{\tilde{\varphi},e'}$ is independent of e' surrounding l' is void.

This concludes the proof that $\tilde{\varphi}$ is a finite harmonic morphism. It remains to show that $\deg\tilde{\varphi} = \deg\varphi$. This is trivial, because the vertices $v \in [u]$ mapping to u', all have $m_{\tilde{\varphi}}(v) = m_\varphi(v)$, and $[u] = \varphi^{-1}(u') = \tilde{\varphi}^{-1}(u')$:

$$\deg\tilde{\varphi} = \sum_{v\in\tilde{\varphi}^{-1}(u')} m_{\tilde{\varphi}}(v) = \sum_{v\in\varphi^{-1}(u')} m_\varphi(v) = \deg\varphi. \qquad \square$$

Now we prove a similar lemma for the subdivision of edges. The proof is also very similar, although a bit more complicated. Therefore, we will only highlight the differences with the proof of the previous lemma.

Lemma 6.3 (Subdivision)**.** *Subdividing an edge does not increase the stable gonality of a graph. That is, for every $e \in E(G)$, we have*

$$\operatorname{sgon}(S_e(G)) \leq \operatorname{sgon}(G).$$

Proof. Again, let H be a refinement $\mathbf{R}$ of G with a degree $\operatorname{sgon}(G)$ finite harmonic morphism φ to a tree T. We will give a refinement $\widetilde{H}$ of $\widetilde{G} = S_e(G)$ and a finite harmonic morphism from $\widetilde{H}$ to $\widetilde{T}$ with the same degree as the stable gonality of G. Unlike before, we have to consider two cases, depending on whether the edge e that we subdivide in $\widetilde{G}$ is still present in H.

Case 1: $\mathbf{R}$ subdivides e. We will show that H already is a refinement of $\widetilde{G}$, so that φ is a finite harmonic morphism from a refinement of $\widetilde{G}$ to a tree with degree $\operatorname{sgon}(G)$.

Write

$$\mathbf{R} = \mathbf{R}_k \circ \cdots \circ \mathbf{R}_1.$$

where each of the $\mathbf{R}_i$ is of the form $\mathbf{L}_u$ or $\mathbf{S}_e$ for some $u \in V(G)$ or $e \in E(G)$. Since $\mathbf{R}$ subdivides the edge e, there is an index j such that $\mathbf{R}_j = \mathbf{S}_e$. Using Lemma 4.8 we may rewrite $\mathbf{R}$ as

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_k \circ \cdots \circ \mathbf{R}_{j+1} \circ \mathbf{S}_e \circ \mathbf{R}_{j-1} \circ \cdots \circ \mathbf{R}_1 \\ &= \mathbf{R}_k \circ \cdots \circ \mathbf{R}_{j+1} \circ \mathbf{R}_{j-1} \circ \cdots \circ \mathbf{R}_1 \circ \mathbf{S}_e \\ &= \left(\mathbf{R}_k \circ \cdots \circ \mathbf{R}_{j+1} \circ \mathbf{R}_{j-1} \circ \cdots \circ \mathbf{R}_1\right) \circ \mathbf{S}_e \\ &= \widetilde{\mathbf{R}} \circ \mathbf{S}_e.\end{aligned}$$

Here, $\widetilde{\mathbf{R}}$ is defined as the expression between parenthesis. Now, $\widetilde{\mathbf{R}}$ sends $\widetilde{G}$ to $\widetilde{H} = H$, and we can use φ again.

Case 2: e is present in H. This case is very similar to the proof of Lemma 6.2. We will focus on the definitions of $\widetilde{\mathbf{R}}$ and $\tilde{\varphi}$ and show that $\tilde{\varphi}$ is indeed a finite harmonic morphism . The new commutative diagram is shown in Figure 6.3 and an explicit example can be seen in Figure 6.4.

We have $\widetilde{H} = S_{[e]}(H)$. Since the refinement $\mathbf{S}_{[e]}$ subdivides $\{u_l, u_r\} = e \in [e]$, we can define

$$\widetilde{\mathbf{R}} = \mathbf{S}_{[e]\setminus\{e\}} \circ \mathbf{R}.$$

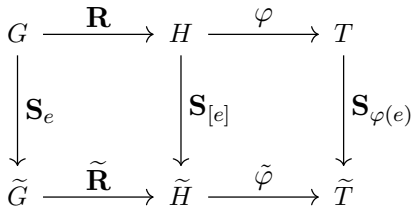


Figure 6.3: We use this diagram in the proof of Lemma 6.3. Instead of adding leaves, this time we subdivide edges.

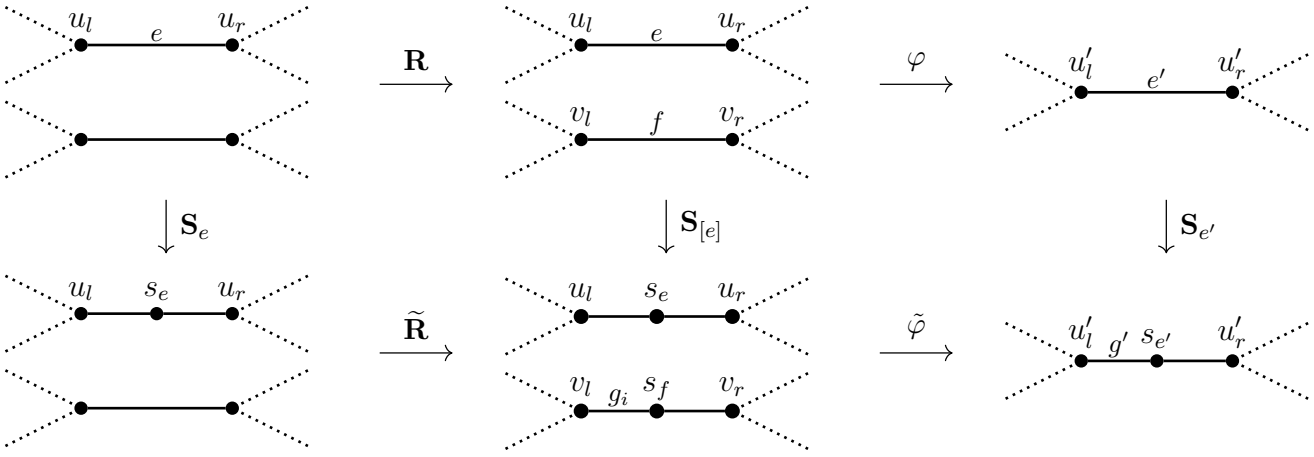


Figure 6.4: Again, this is the same diagram as figure Figure 6.3, but with explicit graphs. In this example, $\varphi(e) = \varphi(f) = e'$. Remark 6.4 explains why we cannot label f in the left two graphs.

Because $\mathbf{R}$ does not alter e, $\mathbf{R}$ and $\mathbf{S}_e$ commute and therefore left part of the diagram commutes,

$$\widetilde{\mathbf{R}} \circ \mathbf{S}_e = \mathbf{S}_{[e]\setminus\{e\}} \circ \mathbf{R} \circ \mathbf{S}_e = \mathbf{S}_{[e]\setminus\{e\}} \circ \mathbf{S}_e \circ \mathbf{R} = \mathbf{S}_{[e]} \circ \mathbf{R}.$$

Now $\widetilde{H}$ is a refinement of $\widetilde{G}$ and thus we can define $\tilde{\varphi}$. The image of $\tilde{\varphi}$ is the tree $\widetilde{T} = S_{e'}(T)$. Again, $\tilde{\varphi}$ coincides with φ on H. For any edge $\{v_l, v_r\} = f \in [e]$ we define

$$\begin{aligned}
\tilde{\varphi}(s_f) &= s_{e'}, \\
\tilde{\varphi}(v_l s_f) &= u'_l s_{e'}, \\
\tilde{\varphi}(v_r s_f) &= u'_r s_{e'}, \\
r_{\tilde{\varphi}}(v_l s_f) &= r_\varphi(f), \\
r_{\tilde{\varphi}}(v_r s_f) &= r_\varphi(f).
\end{aligned}$$

To show that $\tilde{\varphi}$ is a finite harmonic morphism, we must show that $m_{\tilde{\varphi},e'}(v)$ is independent of $e' \in E(\tilde{\varphi}(v))$ for v. When v is not an endpoint of an edge in $[e]$ or of the form s_f for $f \in [e]$, nothing changed so this is true.

When $v = s_f$, there are two edges adjacent to $\tilde{\varphi}(v)$, and they both have one edge neighbouring v mapping to it. Since both edges adjacent to v have index $r_\varphi(e)$, $m_{\tilde{\varphi}}(s_f)$ is well defined.

Now, let v be an endpoint of some edge $f \in [e] \subset H$. We would like to show that the index $m_{\tilde{\varphi}}(v)$ is well-defined and equal to the index $m_\varphi(v)$ of $v \in H$. Let $v' = \tilde{\varphi}(v)$

be the image of v in $\widetilde{T}$. Since the edges in $\widetilde{H}$ mapping to edges h' in $E(v') - [g']$, with $g' = \{u'_l, s_{e'}\}$, have not changed, the corresponding indices $m_{\varphi,h'}(v)$ just equal $m_\varphi(v)$. Hence, we must show that

$$m_{\tilde{\varphi},g'}(v) = m_\varphi(v).$$

Enumerate all edges in $E(v) \subset \widetilde{H}$ mapping to g' by $g_1, \ldots, g_k$. Then, we have that

$$m_{\tilde{\varphi},g'}(v) = \sum_{\substack{e \in E(v), \\ \tilde{\varphi}(e) = g'}} r_{\tilde{\varphi}}(e) = \sum_{i=1}^{k} r_{\tilde{\varphi}}(g_i).$$

We know that every g_i is an edge from v to some new vertex s_{f_i}, where f_i is an edge in $[e] \subset H$ adjacent to v. Since we defined $r_{\tilde{\varphi}}(\{v, s_{f_i}\}) = r_\varphi(f_i)$, we obtain

$$m_{\tilde{\varphi},g'}(v) = \sum_{i=1}^{k} r_{\tilde{\varphi}}(g_i) = \sum_{i=1}^{k} r_\varphi(f_i) = \sum_{\substack{f \in E(V), \\ \varphi(f) = e'}} r_\varphi(f) = m_\varphi(v).$$

We conclude that $\tilde{\varphi}$ is a finite harmonic morphism again, and as before, the equality of degree follows trivially. □

Remark 6.4. The edge f as shown in Figure 6.4 might not be present in G, because it can be part of a subdivided edge. This is not a problem, because $[e]$ contains only edges in H.

Now we are finally ready to state the main result of this section.

Theorem 6.5. *Refinement-equivalent graphs have the same stable gonality.*

Proof. Lemma 6.2 and Lemma 6.3 together imply that the stable gonality does not increase when we take refinements. Lemma 6.1 states that is does not decrease, so we conclude that the stable gonality is constant under refinements. When G_1 and G_2 are refinement-equivalent they are refinements of the stable same graph H by Theorem 4.18. We thus have that

$$\operatorname{sgon} G_1 = \operatorname{sgon} H = \operatorname{sgon} G_2,$$

as required. □

7 The algorithm

Before proceeding with further theorems, I will explain the algorithm to calculate the stable gonality of graphs. This section describes the algorithm itself in full detail, but it omits arguments as to why it works. These arguments are presented in the sections following this one and I will refer to them when necessary. Also, I informally introduce some new notation which is formally defined later on.

7.1 Pseudocode

All pseudocode is explained in the paragraph below the code.

1: **procedure** STABLE GONALITY(G)

The input of the algorithm will be a connected multigraph G. The output will be the stable gonality of G, sgon(G). Furthermore, the algorithm results in two functions f and r, that can be used to construct a refinement H of G and a finite harmonic morphism to a tree.

2: **while** There is a $v \in V(G)$ with $1 \leq \deg(v) \leq 2$ **do**
3: Contract v to a neighbour
4: **end while**

The algorithm we will describe depends on the number of vertices and edges of G. Thus, we would like to make G as small as possible before proceeding. Since we know that the stable gonality of refinement-equivalent graphs is equal, we start by looking at the graph with the least number of vertices that is equivalent to G. This is the stabilisation of G (Definition 4.22), and it can be obtained by repeatedly contracting vertices of degree 1 and 2 to (one of) their neighbour(s). From now on, we will assume that G is a stable graph.

5: **if** $|E(G)| = 0$ **then**
6: **return** sgon$(G) = 1$
7: **end if**

When the input graph G is a tree, we are left with a single vertex with no edges. In this case, the stable gonality is 1.

8: **if** $|V(G)| = 1$ **then**
9: **return** sgon$(G) = 2$
10: **end if**

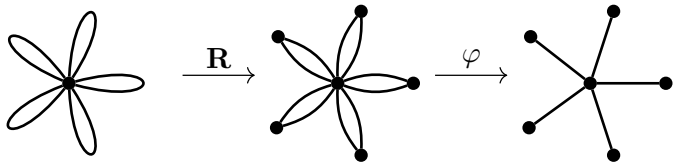


Figure 7.1: This shows the special case where G only has a single vertex after stabilizing it. We subdivide each edge once and map it to the star graph shown on the right. This finite harmonic morphism has degree 2.

When G is not a tree, but has only one vertex remaining after stabilizing it, all edges must be loops. We obtain a degree 2 finite harmonic morphism if we subdivide each loop once, and because G is not a tree, $\text{sgon}(G) = 2$. This is shown in Figure 7.1.

11: $T \leftarrow T_{|V(G)|}$ ▷ T_n is defined below
12: $r' \leftarrow$ the root of T
13: $sgon \leftarrow \infty$

Now, we consider a minor-universal tree T_n, where $n = |V(G)|$ is the number of vertices of G. The minor-universal tree on n vertices is the smallest tree containing all trees on at most n vertices as a minor. Section 9 gives some bounds regarding the size of minimal rooted-universal trees, which are rooted trees containing all rooted trees on at most n vertices as a rooted subtree. Also, we give a construction of R_n, the minimal rooted-universal tree. We will write $\alpha(n)$ for the number of vertices of a minimal minor-universal tree. Since every rooted-universal tree is also a minor-universal tree, equation 9.1 in Section 9 gives us

$$\alpha(n) \leq n^{\frac{1}{2\ln 2}(\ln n - 2\ln\ln n + O(1))}.$$

We write r' for the root of T_n, and initialize the stable gonality to infinity, because we will minimize it over all finite harmonic morphisms.

In Section 8 we will show that for any refinement $\mathbf{R}$ and finite harmonic morphism φ to a tree T, we can construct a new refinement and morphism such that φ' has the same degree as φ and T' has at most n nonleaf vertices. Because T_n contains all trees on n vertices as a minor, we know that the induced subtree of the nonleaf vertices of $\widetilde{T}$ will be a minor of T_n, or equivalently, that $\widetilde{T}$ is a minor of T_n when we add enough new leaves. There is only one case in the proof of this theorem where we actually need the vertices to be nonleaves, and we will deal with this case separately in the algorithm. Because of Lemma 5.11, we can extend any finite harmonic morphism to a tree T' to a finite harmonic morphism to T_n, provided that we add some extra leaves for the special case. This allows us to use T_n, instead of a larger universal tree that also accounts for the leaves we might need to add.

14: **for all** $(u, v) \in V(T)^2$ with $u \neq v$ **do**
15: $dir(u, v) \leftarrow$ the edge incident to u in the direction of v
16: **end for**

Since T is a tree, for every pair of distinct vertices $u, v \in V(T)$, there is a unique neighbour of u that is closer to v than u is. All these neighbours can be calculated in $V(T)^2$ time using a simple depth first search from every vertex in $V(T)$. The edges follow directly from these neighbours.

We will need these values later in the algorithm.

17: **for all** $f : V(G) \to V(T)$ **do**
18: **for all** $r : E(G) \to \{1, \ldots, \beta(G)$ **do**

Instead of iterating over all refinements H of G and finite harmonic morphisms φ from H to T, we iterate over functions $f : V(G) \to V(T)$ and $r : E(G) \to \{1, \ldots, \beta(G)\}$ for some $\beta(G)$ that will be specified later.

In Section 7.2 I explain how we find $\mathbf{R}$ and φ from f and r in more detail. We will use that $f = \varphi \circ \mathbf{R}_{V(G)}$ and that the indices of H can be calculated from r. We say that $(\mathbf{R}, \varphi)$ is the *reconstruction* of (f, r). Now, I will prove that this map

$(f, r) \mapsto (\mathbf{R}, \varphi)$ contains an optimal finite harmonic morphism in its image. Or, more precise, that there is a pair (f, r) such that the φ we create from it has degree $\operatorname{sgon}(G)$.

Suppose that $\tilde{\varphi}$ is a finite harmonic morphism from a refinement $\widetilde{H} = \widetilde{R}(G)$ to T with degree $\operatorname{sgon}(G)$.

Let $\tilde{f}$ be given by $\tilde{\varphi} \circ \widetilde{R}|_{V(G)}$, the image of the vertices of G in T. We iterate over all possible functions f, so this does not yet make any assumptions about $\tilde{\varphi}$.

In Section 10, I show that we may assume that $\widetilde{R}$ is such that for all $e \in E(G)$, all of $\widetilde{R}^G(e)$ (the image of e under $\mathbf{R}$, Definition 4.10) has the same index. Therefore, we iterate r over all possible assignments of indices to the edges of G, and construct H and φ in such a way that this condition is satisfied.

The actual shape of H is determined by T, and we will refine each edge $e = uv \in E(G)$ to $T_{f(u)f(v)}$ in H.

Next, I prove that we may assume there is some upper bound on the indices of $\tilde{\varphi}$. In particular, Corollary 11.5 bounds the indices of $\tilde{\varphi}$ at $\beta(G) = |E(G)|$.

```
19:     degree ← 0
20:     for all u ∈ V(G) do
21:         m_{φ,min}(u) ← 1
22:         for all e' ∈ E(f(u)) do
23:             m_{φ,e'}(u) ← 0
24:         end for
25:     end for
```

All we have to do now is to determine the degree of the finite harmonic morphism that corresponds to (f, r). We calculate the degree incrementally, so we initialize it to 0. Furthermore, we initialize the minimum value of $m_\varphi(u)$ to 1 for each u, because this can never be 0.

Finally, we initialize all values of $m_{\varphi,e'}(u)$ to 0, because they will also be calculated while processing all edges.

```
26:     for all e = uv ∈ E(G) do
27:         u' ← f(u)
28:         v' ← f(v)
29:         if u' = v' then
30:             if u = v then
31:                 m_{φ,min}(u) = 2
32:             end if
```

We loop over all edges $e \in E(G)$ we set $u' = f(u)$ and $v' = f(v)$. If $u = v$, e is a loop. Note that this is allowed, because G may have cycles after stabilizing it. This is the special case of Theorem 8.6. In this case, we subdivide the loop once with vertex s and add a new leaf s' to $u' = v'$. Then, we map the edges us and vs to $u's' = v's'$. Now, $m_{\varphi,u's'}(u)$ equals 2. Hence, $m_\varphi(u)$ must be at least 2, so we set $m_{\varphi,\min}(u) = 2$.

When $u' = v'$, but $u \neq v$, we also subdivide uv with s and add a new vertex s' that is connected to $u' = v'$. We extend φ as in the previous case. Now, $m_{\varphi,u's'}(u) = m_{\varphi,v's'}(v) = 1$, so we must have $m_\varphi(u) \geq 1$. This is a trivial condition, because $m_\varphi(u) = 0$ will never happen. Because of this, $m_{\varphi,\min}(u)$ is initialized to 1.

```
33:         else
34:             e' ← dir(u', v')
35:             m_{φ,e'}(u) ← m_{φ,e'}(u) + r(e)
36:             f' ← dir(v', u')
37:             m_{φ,f'}(v) ← m_{φ,f'}(v) + r(e)
```

If $u' \neq v'$, e is a normal edge. We refine e to the subtree $T_{u'v'}$ of T with index $r(e)$. We add $r(e)$ to the indices $m_{\varphi,e'}(u)$ and $m_{\varphi,f'}(v)$, with e' the edge leaving u' in the direction of v', and f the edge leaving v' in the direction of u'.

```
38:             if T_{u',v'} covers r' then
39:                 degree ← degree + index
40:             end if
41:         end if
42:     end for                                   ▷ End of loop over e
```

If the subtree we add covers the root r', we add $r(e)$ to the degree. Note that this does not interfere with the addition of new leaves when $u' = v'$, because we only have to know whether r' is contained in the subtree.

```
43:     for all u ∈ V(G) do
44:         m_φ(u) ← max(m_{φ,min}(u), max_{e'∈E(u')}{m_{φ,e'}(u)})
```

Now, we iterate over all vertices $u \in V(G)$ to make sure the harmonic property is satisfied everywhere. We calculate the final value of $m_\varphi(u)$ as the maximum over e' of all $m_{\varphi,e'}(u)$. If this maximum is 1, but $m_{\varphi,\min}(u)$ is 2, we set $m_\varphi(u)$ to 2.

```
45:         if u' = r' then
46:             degree ← degree + m_φ(u)
47:         else
48:             e' ← dir(u', r')
49:             degree ← degree + m_φ(u) − m_{φ,e'}(u)
50:         end if
51:     end for                                   ▷ End of loop over u
```

If u' is the root r' of T, we have to add $m_\varphi(u)$ to the degree. We have not counted this before, because we only increased the degree for $T_{u'v'}$ subtrees where r' is an internal node.

When u' is not r', we add subtrees to u for all $e' \in E(u')$ where $m_{\varphi,e'}(u) < m_\varphi(u)$. The only time this subtree covers r' is when e' is the edge from u' to $dir(u', r')$. Thus, we set e' to that edge and add the corresponding number of subtrees to the degree of φ.

```
52:     if degree < sgon then
53:         sgon ← degree
54:         Save f and r as optimal functions
55:     end if
```

If the degree of the finite harmonic morphism corresponding to the current (f, r) pair is strictly smaller than the best degree we have found so far, we update the stable gonality. We save f and r, so that they can be used to obtain the actual graph H and finite harmonic morphism φ.

```
56:         end for                                   ▷ End of loop over r
57:     end for                                       ▷ End of loop over f
58:     return sgon(G) = sgon
59: end procedure
```

We complete the algorithm by returning the stable gonality we have found.

7.2 Constructing H and φ from f and r

We are given

$$f : V(G) \to V(T_n)$$

and

$$r : E(G) \to \{1, \ldots, \beta(G)\}$$

for some $\beta(G)$. We would like to construct a refinement H of G and a finite harmonic morphism φ from H to T_n such that

$$\varphi \circ \mathbf{R}|_{V(G)} = f$$

and

$$r_\varphi(f) = r(e)$$

for all edges f in H that are in the image of $\mathbf{R}^G(e)$, the image under $\mathbf{R}$ of the subgraph consisting of the endpoints of e and e itself for some edge $e \in E(G)$.

Start with $H = G$. We process all edges of G separately.

Case 1: $f(u) = f(v)$. We add a new leaf w' to T_n at $f(u) = f(v)$, and subdivide uv once with a new vertex w. We set $\varphi(w) = w'$ and the indices of the edges uw and vw will be 1.

The tree obtained from T_n after adding the necessary new leaves will be denoted T, and the refinement from G to H will be written $\mathbf{R}$.

Case 2: $f(u) \neq f(v)$. For each edge $e = uv \in E(G)$ with $f(u) \neq f(v)$, we replace $e \in E(H)$ by

$$T_{f(u)f(v)},$$

which is the part of T_n between $f(u)$ and $f(v)$ (Definition 3.35). An example of this is shown in Figure 7.2 Note that this is a refinement of H, since we first subdivide the edge e, $k-1$ times, were k is the length of the path from $f(u)$ to $f(v)$. Then, we add leaves to match the subtrees along the path from $f(u)$ to $f(v)$. We set $\varphi(u) = f(u)$, $\varphi(v) = f(v)$, and φ is the identity on the subtree of T we used to replace e. Furthermore, we prove in Section 11 that we may assume that φ has the

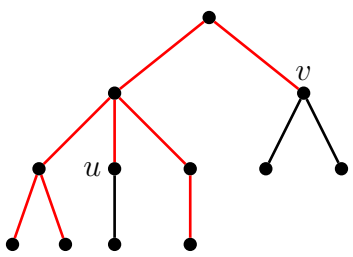


Figure 7.2: When we are given a tree T and two vertices u and v, the uv-subtree is the part of T between u and v, as show in red in this image.

same index on all edges that originated at the same edge in G. Thus, we assign index $r(e)$ to all edges in the newly added $T_{,f(u)f(v)}$.

At this moment, we can already prove that φ is harmonic at all vertices in H that are not in G. Let x be such a vertex. Assume that x was added in the refinement of an edge $e = uv \in G$. There are two cases, just like in the construction of H.

Case 1: $f(u) \neq f(v)$. The index of all edges surrounding x is a constant $r = r(e)$, because all edges surrounding x were created in a single replacement of an edge e in G. Also, we know that the degree of x equals the degree of $\varphi(x)$, because $\mathbf{R}^G(e)$ is a copy of $T_{f(u)f(v)}$. Hence, every edge incident to $\varphi(x)$ is the image of exactly on edge incident to u, and thus $m_\varphi(x) = r$ is independent of the edge e' incident to $\varphi(x)$ we choose.

Case 2: $f(u) = f(v)$. Since the image of x is a leaf w' in this case, the harmonic property is trivially satisfied at x.

We are left with the case that $x \in V(H)$ is also a vertex in G. We keep a counter $m_{\varphi,e'}(x)$ for every edge e' incident to $\varphi(x) = f(x)$. For every edge xy in H we will increment the counter $m_{\varphi,\varphi(xy)}(x)$ by $r(e)$, where e is the edge in G that created xy. Then, we will set $m_\varphi(x) = \max_{e' \in E(\varphi(x))} m_{\varphi,e'}(x)$. Now, for every edge $e' = \varphi(x)y' \in E(\varphi(x))$ we have $m_{\varphi,e'}(x) \leq m_\varphi(x)$. When the inequality is strict, we will add a subtree to H. More precise, we add a copy of $T_{\varphi(x)}(y')$ to H such that $\varphi(x)$ corresponds to x. This is again a refinement of H, since we can add the new tree by the repeated addition of leaves. The index of the edges in this new subtree will be $m_\varphi(x) - m_{\varphi,e'}(x)$.

Just like before, the harmonic property is satisfied for the vertices in the new subtree we added. In x itself, we added new subtrees to H is such a way that all $m_{\varphi,e'}(x)$ have become equal, so φ is harmonic in x too.

If we do this procedure for all vertices $x \in V(G) \subset V(H)$, we obtain the final refinement H of G and a finite harmonic morphism from H to T. We still have to determine the degree of the finite harmonic morphism. We do this by calculating the sum $\sum_{r \in \varphi^{-1}(r')} m_\varphi(r)$ for a certain vertex $r' \in T_n$. In practice, we choose the root r' of T.

In the actual implementation of the algorithm, we do not have to construct H, since this is not needed to calculate the degree of φ. Instead, we calculate the degree on the fly, by keeping track of $\sum_{r \in \varphi^{-1}(r')} m_\varphi(r)$, the sum of the indices of the vertices mapping to the root r' of T_n. When we add a subtree between $f(u)$ and $f(v)$ that covers r', $f(u) \neq r'$ and $f(v) \neq r'$, we increment the degree with the index of the subtree we add. When $f(u) = f(v)$ and $u \neq v$, we force $m_\varphi(u) \geq 1$ and $m_\varphi(v) \geq 1$, since we need some room in m_φ for the edges we add to the new leaf. When $f(u) = f(v)$ and $u = v$, that is, G has a loop at u, $m_\varphi(u)$ must be at least 2, because we need two edges in the direction of the new leaf we add. These constraints are only evaluated after the calculation of the maximum $m_{\varphi,e'}(u)$ for each u.

In the last phase, when we add the extra subtrees to make all $m_{\varphi,e'}(u)$ equal, we add the indices of the subtrees covering r' to the degree. Furthermore, we need to add $m_\varphi(u)$ for all vertices u in $\varphi^{-1}(r') \cap V(G)$.

7.3 Asymptotic runtime

Having described the full algorithm, we are ready to analyse its asymptotic runtime.

We assume that the input graph G is stable. Otherwise, we start by contracting

vertices in G until G is stable. This operation is linear in the number of vertices and edges of the input graph G, since the number of vertices of G decreases after each contraction. From now on, we will assume G is stable. We write $n = |V(G)|$ and $m = |E(G)|$ for the number of vertices and edges of the stable graph G. Furthermore, we assume that a minimal (minor-)universal tree can be constructed in linear time in its size $\alpha(n)$.

We start by iterating over all maps $f : V(G) \to V(T_n)$. There are $\alpha(n)^n$ such maps. Because each map can be constructed from the previous one, each function is constructed in amortized constant time, yielding

$$O(\alpha(n)^n)$$

for the loop over all maps f.

For each map f, we iterate over maps $r : E(G) \to \{1, \ldots, \beta(n)\}$. This takes

$$O(\beta(n)^m)$$

time.

Finally, we have to determine the degree of the finite harmonic morphism. This is done by keeping track of all counters $m_{\varphi,e'}(u)$ for $u \in V(G)$ and $e' \in E(f(u))$. There are at most n^2 such counters. After initializing these counters, we loop over all edges of G to add the necessary subtrees. For each subtree of T_n between two vertices $f(u)$ and $f(v)$, we need to determine whether it contains r'. This can be done in constant time if, at the start of the algorithm, we create a $V(T_n)^2$ table, containing for each pair of vertices of T_n whether the subtree they induce contains r'.

Then, for each $v \in V(G)$ we have to calculate the maximum of all counters surrounding $f(v)$ to determine $m_\varphi(v)$. This costs $O(n^2)$ time. Finally, we calculate the degree of φ by a loop over all vertices of G, which takes $O(n)$ time.

All together, this results in the following run time

$$O\left(\alpha(n)^2 + \alpha(n)^n \cdot \beta(G)^m \cdot (n^2 + m)\right) = O\left((n^2 + m)\alpha(n)^n \beta(G)^m\right).$$

In this thesis (equation 9.1), we show that

$$\alpha(n) \leq \exp\left(\ln n \cdot \frac{\ln n - 2\ln\ln n + c}{2\ln 2}\right).$$

for some unknown constant c, using minimal rooted-universal trees. Also, we prove (Corollary 11.5) that the indices of the edges of the optimal finite harmonic morphism are bounded by

$$\beta(G) \leq |E(G)| = m.$$

Substituting these values results in

$$O\left((n^2 + m)\exp\left(n\ln n \cdot \frac{\ln n - 2\ln\ln n + c}{2\ln 2}\right) m^m\right).$$

Note that there is still a unknown constant c in the exponent, which means that we could be off by a power of n^n.

If we use the best bounds known in the literature (equation 9.2), we obtain

$$\alpha(n) \leq \frac{2\sqrt{2}}{n} n^{\ln n/2 \ln 2}.$$

by using minimal unrooted trees as in [5]. Cornelissen et al. have shown in [6] (Theorem B) that the stable gonality of a graph is bounded by

$$\operatorname{sgon}(G) \leq \left\lfloor \frac{m-n+4}{2} \right\rfloor.$$

This implies that the indices of all edges will be at most

$$\beta(G) \leq \left\lfloor \frac{m-n+4}{2} \right\rfloor$$

too. Because we assume G is stable, $m \geq \frac{3}{2}n$ (excluding trivial graphs), which implies that $m-n+4$ is asymptotically equal to m. Using these bounds, we obtain the following theorem.

Theorem 7.1 (Runtime)**.** *The stable gonality of a stable graph with n vertices and m edges can be calculated in*

$$O\left((n^2+m) n^{n \ln n/2 \ln 2 - n} m^m\right)$$

time.

If we assume that $m = O(n^k)$ for some constant k, which holds for simple graphs, this can be simplified to

$$n^{O(n \ln n + m)}$$

where the $n \ln n$ comes from the loop over maps f and the m comes from the loop over the indices r.

7.4 Performance

I made a C++ implementation of this algorithm, which is available upon request. In this section, I present some runtimes of the algorithm using the rooted-universal trees R_n as the image of φ. Because R_n contains all rooted trees on n vertices, we may assume that the image of a given vertex in G is r', the root of R_n. This reduces the runtime by a factor $\alpha(n)$. The upper bound on the indices I used is the one derived from the upper bound on the stable gonality given by Cornelissen et al.

The runtime for the complete graphs K_1, K_2 and K_3 is only a few milliseconds. Calculating $\operatorname{sgon}(K_4)$ takes 0.1 seconds. Calculating $\operatorname{sgon}(K_5)$ already takes more than an hour.

I also ran the algorithm on complete bipartite graphs $K_{a,b}$. When $a = 1$, the graph is a tree, and hence $\operatorname{sgon}(K_{a,b}) = 1$ is detected in linear time. When $a = 2$ and $b \leq 9$, the algorithm finishes within a second. When $a = b = 3$, the algorithm again takes much more than an hour.

Since the current algorithm already struggles with K_5 and $K_{3,3}$, it is quite unfeasible to use in practice. This is easily explained if we consider the sizes of the loops.

Because the number of maps we try for the indices r grows as $\left\lfloor \frac{m-n+4}{2} \right\rfloor^m$. For K_5, we have $n = 5$ and $m = \binom{n}{2} = 10$, so we have to try

$$\left\lfloor \frac{m-n+4}{2} \right\rfloor^m = 4^{10}$$

such maps for each map f. Since R_5 has 12 vertices, there are $\alpha(n)^{n-1} = 12^4$ maps we have to try for f. Multiplying these two numbers yields over $2 \cdot 10^{10}$ pairs of f and r we will consider. For each of these pairs, we have to calculate the degree of the corresponding finite harmonic morphism, which takes $O(n^2 + m)$ time too.

If we consider that a modern computer can do about 10^8 operations per second, it is not surprising that this takes quite a long time.

8 Bounding the tree size

Now, we will investigate ways to reduce the number of trees we have to consider as an image of φ. We start by proving a lemma concerning edge contractions in refinements, and use this lemma to bound the number of nonleaf nodes of the trees we have to try.

8.1 Contracting edges in finite harmonic morphisms

We start with the following lemma, which states that when we contract edges in T and H in a special way, the new morphism induced from φ is still harmonic and has at most the same degree as φ.

Lemma 8.1. *Let a finite harmonic morphism $\varphi : H \to T$ to a tree T and an edge $e' = u'v' \in E(T)$ be given. Then there exists a finite harmonic morphism $\tilde{\varphi} : H/\varphi^{-1}(e') \to T/e'$ with at most the same degree as φ, with equality when T contains at least 2 edges.*

Proof. The variables we use are shown in Figure 8.2. We will write $\widetilde{H} = H/\varphi^{-1}(e')$ and $\widetilde{T} = T/e'$. The new vertex in $\widetilde{T}$ will be denoted $\tilde{u}' = \tilde{v}'$ Let $\tilde{\varphi}$ be given by

$$\begin{aligned}
\tilde{\varphi} : V(\widetilde{H}) &\to V(\widetilde{T}) \\
\tilde{\varphi}(w) &= \begin{cases} \varphi(w) & \varphi(w) \notin e' \\ \tilde{u}' & \varphi(w) \in e', \end{cases} \\
\tilde{\varphi} : E(\widetilde{H}) &\to E(\widetilde{T}) \\
\tilde{\varphi}(f) &= \begin{cases} \varphi(f) & \varphi(f) \cap e' = \emptyset \\ \varphi(f) - [u', v'] + \tilde{u}' & \varphi(f) \cap e' \neq \emptyset, \end{cases} \\
r_{\tilde{\varphi}} : E(\widetilde{H}) &\to \mathbb{N} \\
r_{\tilde{\varphi}}(f) &= r_{\varphi}(f).
\end{aligned}$$

We will now show that $\tilde{\varphi}$ is a finite harmonic morphism and has at most the same degree as φ. Thus, consider a vertex $\tilde{w} \in V(\widetilde{H})$. We have to show that

$$m_{\varphi,f}(\tilde{w}) = \sum_{\substack{e \in E(\tilde{w}), \\ \varphi(e) = f}} r_{\tilde{\varphi}}(e)$$

is independent of $f \in E(\varphi(\tilde{w}))$.

$$\begin{array}{ccc}
H & \xrightarrow{\varphi} & T \\
{\scriptstyle \mathbf{C}_{\varphi^{-1}(e')}}\downarrow & & \downarrow{\scriptstyle \mathbf{C}_{e'}} \\
\widetilde{H} & \xrightarrow{\tilde{\varphi}} & \widetilde{T}
\end{array}$$

Figure 8.1: This is the commuting diagram containing the graphs used in the proof of Lemma 8.1.

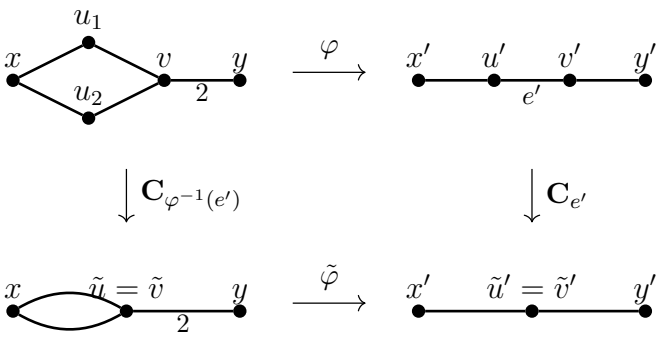


Figure 8.2: An example of the commuting diagram in Figure 8.1. The proof of the lemma uses the same labels as shown in this picture.

We first deal with the case where T consists of a single edge. In this case, all vertices of H are contracted to a single vertex. Hence, the degree of $\tilde{\varphi}$ becomes 1, which is smaller than $\deg \varphi$ when H is not a tree.

From now on, we assume T has at least two edges. We will show that the harmonic property is satisfied at all vertices in $\widetilde{H}$ by considering several cases.

Case 1: $\varphi(\tilde{w}) \neq \tilde{u}'$. In this case, $\tilde{w}$ was not merged with any neighbours, so $\tilde{w}$ is also present in H. We call the occurrence of $\tilde{w}$ in H just w. Since T is a tree, $\varphi(w)$ is not adjacent to both u' and v' and hence, w can not have neighbours mapping to u' and neighbours mapping to v'. If the set $N(w)$ of neighbours is disjunct from $\varphi^{-1}(u') \cup \varphi^{-1}(v')$, $\tilde{\varphi}$ is equal to φ on all neighbours of $\tilde{w} = w$, and thus the condition is satisfied at $\tilde{w}$. Otherwise, assume without loss of generality that there is a nonempty subset $S \subset N(w)$ of neighbours of w that map to u'. In $\widetilde{H}$, some of the points in S may have been merged to a single point. Also the edges to the points in S now map to $w'\tilde{u}'$ instead of $w'u'$. Since any edge $w\tilde{u} \in E(\widetilde{H})$ has the same index as the original edge $wu \in E(H)$, it is immediate that

$$m_{\tilde{\varphi},w'\tilde{u}'}(w) = m_{\varphi,w'u'}(w).$$

This proves the required equality when $\varphi(w) \neq \tilde{u}'$.

Case 2: $\varphi(\tilde{w}) = \tilde{u}' = \tilde{v}'$. We will show that

$$m_{\tilde{\varphi},\tilde{u}'x'}(\tilde{w}) = m_{\tilde{\varphi},\tilde{v}'y'}(\tilde{w})$$

for all vertices $x' \in N(u')\backslash\{v'\}$ and $y' \in N(v')\backslash\{x'\}$. If exactly one of these sets is empty, u' or v' is a leaf and only one side of this equality makes sense. When u' is a leaf, this means that $m_{\tilde{\varphi},\tilde{v}'y'}(\tilde{w})$ is constant over y', and $m_{\tilde{\varphi}}$ will still be well-defined. When y' is a leaf, this holds the other way around. Because T has at least 2 edges, u' and v' can not both be leaves, so at least one of these sets is nonempty.

Let $U \subset V(H)$ consist of those vertices u that have been contracted to $\tilde{w}$ with $\varphi(u) = u'$. Similarly, $V \subset V(H)$ consists of the vertices that have been contracted

to $\tilde{w}$ that originally mapped to v'. Using these sets, we have

$$\begin{aligned} m_{\tilde{\varphi},\tilde{u}'x'}(\tilde{w}) &= \sum_{u\in U} m_{\varphi,u'x'}(u) = \sum_{u\in U} m_{\varphi,u'v'}(u) \\ &= \sum_{e\in E(U,V)} r_\varphi(e) \\ &= \sum_{v\in V} m_{\varphi,v'u'}(v) = \sum_{v\in V} m_{\varphi,v'y'}(v) = m_{\tilde{\varphi},\tilde{v}'y'}(\tilde{w}). \end{aligned}$$

The first and last equalities are immediate from the contraction. The second and fifth equalities hold because φ is harmonic. The summation over the edges from U to V just counts the edges is a different way. This completes the proof that $m_{\tilde{\varphi},\tilde{u}'x'}$ is independent of the way we choose $x' \in E(\tilde{u}')$. Hence $\tilde{\varphi}$ is a finite harmonic morphism from $\widetilde{H}$ to $\widetilde{T}$ with the same degree as φ. □

8.2 Contraction lemma

Now we will prove the main result of this section, which states that there exists a finite harmonic morphism of minimal degree from a refinement H of G to a tree T where the number of nonleaf vertices of T is at most $|V(G)|$.

Before stating the theorem, we define some new terminology.

Definition 8.2 (Internal)**.** An *internal* vertex of a tree T is a vertex v that is not a leaf, that is,

$$\deg(v) > 1. \qquad \diamondsuit$$

Definition 8.3 (Native)**.** When we are given $G \xrightarrow{\mathbf{R}} H \xrightarrow{\varphi} T$ with $\mathbf{R}$ a refinement and φ a finite harmonic morphism, a vertex $v \in V(H)$ is called *native* when there exists a vertex $u \in V(G)$ such that $v = \mathbf{R}(u)$. A vertex $v' \in V(T)$ is called native when there exists a $u \in V(G)$ such that $v' = \varphi \circ \mathbf{R}(u)$. Thus, every native vertex is the image of some vertex in G. The sets of native vertices in H respectively T are given by

$$\mathbf{R}(V(G)) \qquad\qquad \varphi \circ \mathbf{R}(V(G)). \qquad \diamondsuit$$

Definition 8.4 (Alien)**.** A vertex in H or T that is not native is called an *alien* vertex. $\diamondsuit$

Remark 8.5. Native and alien vertices only exists in the context of a sequence $G \xrightarrow{\mathbf{R}} H \xrightarrow{\varphi} T$. For brevity, I will not always write this as a prerequisite.

Theorem 8.6 (No internal aliens)**.** *Given a finite harmonic morphism $\varphi : H \to T$, with H a refinement of G, there exists a refinement $\widetilde{H}$ of G, a tree $\widetilde{T}$, and a finite harmonic morphism $\tilde{\varphi} : \widetilde{H} \to \widetilde{T}$ such that $\widetilde{T}$ has no internal alien vertices, and* $\deg(\tilde{\varphi}) = \deg(\varphi)$.

Because trees in H are not very interesting (they can be removed without changing the equivalence class of H), we will only consider edges to parts of H that contain loops. For this, we use the stable degree defined in Definition 4.19.

Lemma 8.7. *The stable degree of an alien vertex $v \in H$ is at most* 2.

Proof. We know that $H = \mathbf{R}(G)$, and that $v \notin V(G)$. This implies that v was added somewhere in the refinement. If v was added as a leaf to vertex u, all v-subgraphs $H_v(w)$ with $w \notin V(H_v(u))$ must be v-subtrees in fact. This implies that there is at most one v-subgraph that is not a v-subtree, and thus, $\operatorname{sdeg}(v) \leq 1$.

Now suppose that v was added as a subdivision of an edge uw. All $H_v(x)$ with $w \notin V(H_v(u)) \cup V(H_v(w))$ are trees, since they were added after v was added. This implies that $\operatorname{sdeg}(v) \leq 2$, as required. □

Now we can prove Theorem 8.6.

Proof of Theorem 8.6. We will use a constructive proof where we remove all internal alien vertices of T one by one. We remove such a vertex from T by contracting it to a neighbour. We know that the resulting $\tilde{\varphi} : \widetilde{H} \to \widetilde{T}$ is a harmonic morphism because of Lemma 8.1, but that lemma does not say anything about the changes of H. In particular, $\widetilde{H}$ may not be a refinement of G any more. We will show that $\widetilde{H}$ *is* a refinement of G when we contract an internal alien vertex. Then, the theorem follows immediately.

Suppose that v' is the internal alien vertex in T that we would like to remove. We will contract it to a neighbour $u' \in N(v')$. To show that the structure of H does not change, we will consider two vertices $u, v \in V(H)$ that map to u' and u', and show that we can always make sure that $\widetilde{H}$ is equivalent to H. We will consider several cases. First we look at the stable degree of v, which can be 0, 1 or 2. The first two cases are quite straightforward, while the last case is more involved and requires several subcases.

Case 1: $\operatorname{sdeg}(v) = 0$. The component of v in H must be a tree, and since H is connected, H is a tree. This implies that G is a tree, and we know that trees have a stable gonality of 1.

Case 2: $\operatorname{sdeg}(v) = 1$. There is precisely one edge vu to a v-subgraph that is not a v-subtree. This implies that $H_u(v)$ is a u-subtree. When we contract edges in a tree, we can not create cycles, and thus will be left with a tree. Hence, $H_u(v)/uv$ is still a tree. Because all trees are equivalent, this contraction does not change the equivalence class of H.

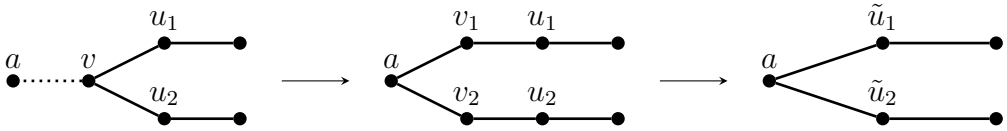
Case 3: $\operatorname{sdeg}(v) = 2$. This case is divided in three subcases which we will treat separately. These cases are based on whether the two v-subgraphs (non-v-subtrees) contain vertices mapping to u'. We will assume that the two v-subgraphs are different to simplify the proof, but in cases 1 and 3, where the two subgraphs could actually be the same, it does not matter. The three different cases are shown in Figure 8.3.

Subcase 3a. In this case, both v-subgraphs do not cover u, so that $H_v(u)$ is a v-subtree. This case is similar to the case where $\operatorname{sdeg}(v) = 1$, so we do not have to do anything here.

Subcase 3b. Now suppose that exactly one v-subgraph covers u. Let $H_v(u)$ be the v-subgraph containing u and let $H_v(a)$ be the other subgraph, not containing u, with a a neighbour of v. Again, we can just contract uv without changing the equivalence class of H. To see that the structure of H does not change, first note that it is invariant under the addition or removal of x-subtrees for vertices x. Then, the only difference between the graph H before the contraction of uv and the graph $\widetilde{H}$ after the contraction is that the edge $a\tilde{u} = a\tilde{v}$ in $\widetilde{H}$ is subdivided in H. Because

(a) Case 3a: both v-subgraphs do not cover u.

(b) Case 3b: only one v-subgraph covers u.



(c) Case 3c: both v-subgraphs cover vertices mapping to u'.

Figure 8.3: The proof of the $\text{sdeg}(v) = 2$ case of Theorem 8.6 requires three subcases, as shown in these graphs. The dotted lines indicate the start of a v-subtree, while bold lines indicate v-subgraphs. In the first case, a and b may actually coincide, as is the case for u_1 and u_2 in the third case. The graphs on the right show the result after applying the contraction.

subdivisions of edges do not change the equivalence class of a graph, we conclude that the contraction of uv does not change the equivalence class of H.

Subcase 3c. The last case we have to consider is when both v-subgraphs cover a vertex mapping to u'. Denote these vertices in H by u_1 and u_2. The vertices u_1 and u_2 may be equal, but that does not change the proof. We would like to contract v to u_1 and u_2 without changing the structure of H. This raises a problem, because these contractions will merge u_1 and u_2. When both u_1 and u_2 have a stable degree greater than 2, this will change the structure of H. It is here that we need that v' is an internal node. Because v' is internal, we know that the image of the neighbourhood of any vertex v mapping to v' contains at least two vertices. Hence, u_1 and u_2 can not be the only neighbours of v. Thus, suppose that a is a neighbour of v not mapping to u'. Then, we can do the transformation shown in Figure 8.3(c).

First note that we may assume that all v-subtrees have edges with index $m_\varphi(v)$, since k v-subtrees with index 1 can be replaced with one v-subtees with index k. (Here we assume that there a no v-subtrees covering u'.)

Because the index of v is $r_\varphi(vu_1) + r_\varphi(vu_2)$, we can split it to two new vertices v_1 and v_2, one connecting to u_1 and a, and the other connecting to u_2 and a. Thus, v_1u_1 and v_1a have index $r_\varphi(vu_1)$, and v_2u_2 and v_2a have index $r_\varphi(vu_2)$. Then, we split the v-subtrees over v_1 and v_2, such that there is a copy based at v_1 and a copy at v_2, with degrees $r_\varphi(vu_1)$ respectively $r_\varphi(vu_2)$. It is easy to see that this new graph, which is shown in the center of Figure 8.3(c) is equivalent to H, because, up to subtrees, we only subdivided two edges, together with a relabeling of the vertices. Now, v_1 and v_2 are vertices like those in case 3b, which we may contract into u_1 and u_2 respectively.

In all these cases, we have omitted any other v-subtrees potentially covering u'.

These do not pose any problems for the contraction of the uv edge, because edges in v-subtrees can always be contracted, as we have seen before. □

Again, we summarize all newly introduced notation.

Notation 7: Contraction lemma

Internal vertex	$\deg(v) > 1$
Native vertex	$v \in \varphi \circ \mathbf{R}(V(G))$
Alien vertex	$v \notin \varphi \circ \mathbf{R}(V(G))$

9 Minimal universal trees

In this section we will take a look at trees containing all trees on n vertices. Such trees are called *universal trees*.

We first define the class of trees containing at most n vertices.

Definition 9.1 (Class of trees)**.** The class of graphs $\mathcal{T}_n$ contains all trees with n vertices. The class $\mathcal{R}_n$ contains all rooted trees on n vertices. ♢

Remark 9.2. Some papers [3, 5] define $\mathcal{T}_n$ as the class containing all trees with n edges, while others do not explicitly define this class, but calculate the size of universal trees for trees containing n vertices.

Because the focus is more on vertices in this thesis, we let n denote the number of vertices of a tree, and not the number of edges.

Now, we will define several kinds of universal trees.

Definition 9.3 (Universal trees)**.** A *universal tree* is a tree containing all trees in $\mathcal{T}(n)$ as a subtree. The minimal number of vertices of a universal tree is written $u(n)$.

A *rooted-universal tree* is a rooted tree containing all rooted trees in $\mathcal{R}_n$ as a rooted subtree. Its minimal size is written $r(n)$.

Finally, a *minor-universal tree* is a tree containing all trees in $\mathcal{T}_n$ as a minor. ♢

In this section, we will show that $r(n)$ satisfies

$$r(1) = 1$$
$$r(n+1) = 1 + \sum_{i=1}^{n} r\left(\left\lceil \frac{n}{i} \right\rceil\right).$$

using a proof by Gol′dberg and Livšic [7]. They were the first to prove this recursion, and they argue that

$$\ln r(n) \sim \frac{\ln^2 n}{2 \ln 2}.$$

By a more careful analyses of the recursion involved, Chung and Graham [4] sharpened this to

$$r(n) = \exp\left(\ln n \cdot \frac{\ln n - 2 \ln\ln n + O(1)}{2 \ln 2}\right). \tag{9.1}$$

Furthermore, in [5] they prove that unrooted-universal trees also satisfy this equation, but are in fact polynomially smaller than rooted-universal trees,

$$u(n) \leq \frac{2\sqrt{2}}{n} \exp\left(\frac{\ln^2 n}{2 \ln 2}\right). \tag{9.2}$$

Also, they stated in [4] that it can be shown that

$$u(n) = n^{-3/2+O(1)} r(n),$$

but they never published the proof.

For the purposes of our algorithm, we only need a tree in which all trees of up to n vertices are a minor. Sadly, no research has been done on minor-universal trees yet. Therefore, I will give a construction of minimal rooted-universal trees. Then, I will give a proof of their minimality.

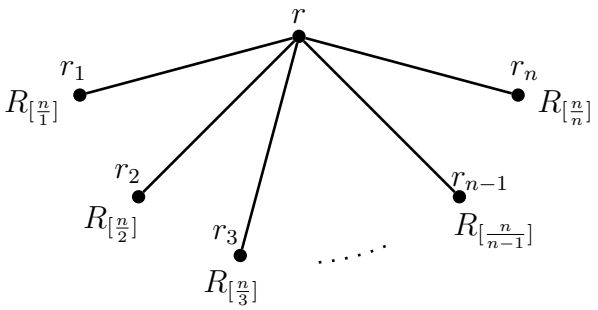


Figure 9.1: This figure shows the inductive construction of the minimal rooted-universal tree R_{n+1}. The root is r, and the subtrees below r are smaller rooted universal trees.

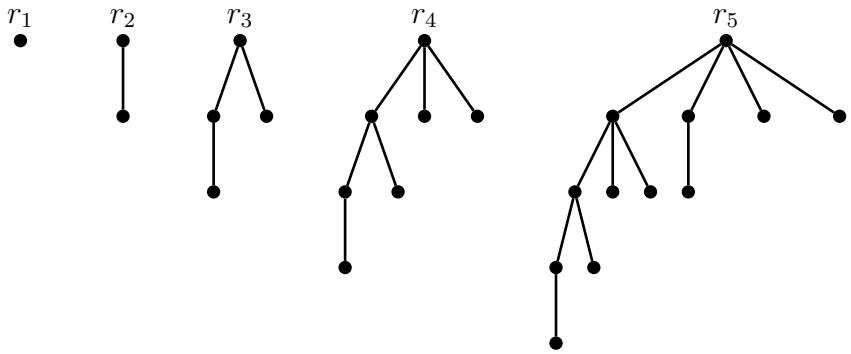


Figure 9.2: This shows the first five rooted universal trees. As can be seen in R_5, we add $R_{\frac{5-1}{2}} = R_2$ as second subtree.

Definition 9.4 (Construction of R_n)**.** The rooted-universal tree for $n = 1$, R_1, is given by a single point. For $n \geq 1$, R_{n+1} is the rooted tree with root r and n subtrees $R_{\left[\frac{n}{k}\right]}$ for $k = 1, \ldots, k = n$. Here, $[x]$ denotes the entier of x. The root of the k^{th} subtree is called r_k. This construction is shown in Figure 9.1. Some small examples are shown in Figure 9.2. ♢

We will first show that R_n is indeed a universal tree. Then, we prove that R_n is in fact a minimal rooted-universal tree.

Lemma 9.5. *All rooted trees on n vertices are embedded in R_n.*

Proof. The tree R_1 only has to cover the tree consisting of a single vertex, which it does because it R_1 has 1 vertex.

We will show using induction that R_{n+1} is a universal tree when R_k is a universal tree for $1 \leq k \leq n$. Let $T \in \mathcal{R}_{n+1}$ be a given rooted tree on $n+1$ vertices. We prove that T is embedded in R_{n+1}. The root of T must be mapped to r. Now, we are left with n other vertices of T. Let c be the number of components of $T \backslash \{r\}$. (We will identify the vertices in T with those in R_{n+1} via the embedding we construct.) Order the c components in decreasing order of the number of vertices. We know that the k^{th} largest component has at most $\left\lceil \frac{n}{k} \right\rceil$ vertices, which is can be proven using a proof by contradiction:

Suppose to the contrary the k^{th} component has more than $\left\lceil \frac{n}{k} \right\rceil$ vertices. Then, the first k components would have at least $\left\lceil \frac{n}{k} \right\rceil + 1$ vertices each. This gives a total

n	1	2	3	4	5	6	7	8	9	10	11
$r(n)$	1	2	4	7	12	18	28	39	55	74	100

Table 9.3: The first values of $r(n) = r'(n)$, the number of vertices of R_n. This is sequence A03318 in the Online Encyclopedia of Integer Sequences.

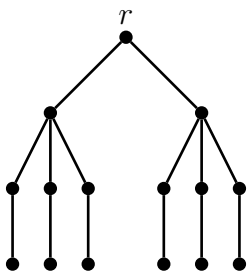


Figure 9.4: An example of a uniform tree. In every layer, all vertices have the same number of children. The characteristic sequence is $(2, 3, 1)$.

of at least

$$k\left(\left[\frac{n}{k}\right]+1\right) > k \cdot \frac{n}{k} = n$$

nonroot vertices. But we know that there are only n nonroot vertices in T, which is a contradiction.

Since the k^{th} component has at most $\left[\frac{n}{k}\right]$ vertices, we know that it is embedded in $R_{\left[\frac{n}{k}\right]}$, because each component of $T\backslash\{r\}$ is a rooted subtree. Because the k^{th} largest subtree of R_{n+1} is the rooted-universal tree on $\left[\frac{n}{k}\right]$ vertices, the k^{th} subtree of T can be embedded in the k^{th} subtree of R_{n+1}. Hence, T can be embedded in R_{n+1}, completing the proof. □

In the end, we are interested in the sizes of the universal trees we have constructed. The recurrence relation in the following lemma follows directly from their construction.

Lemma 9.6 (Recurrence relation for r')**.** *Writing $r'(k) = |V(R_k)|$, we have $r'(1) = 1$ and for $n \geq 1$*

$$r'(n+1) = 1 + \sum_{i=1}^{n} r'\left(\left[\frac{n}{i}\right]\right).$$

Now we know that R_n is a universal tree, it would be nice if we could show its minimality. Gol′dberg and Livšic have shown this in [7] using uniform trees. Below, I present this proof. Some small values of $r' = r$ are shown in Table 9.3.

We start with the definition of uniform trees, which form the basis of this proof.

Definition 9.7 (Uniform trees)**.** A rooted tree is *uniform* if in each level of the tree, all vertices have the same number of children. ♢

Figure 9.4 shows an example of a uniform tree.

We will write $a(n)$ for the number of uniform trees with at most n vertices. Obviously, $a(1) = 1$. For $a(n+1)$, we have a single root and n remaining vertices. The number of children of the root can be at most n, and when we fix it at $1 \leq k \leq n$,

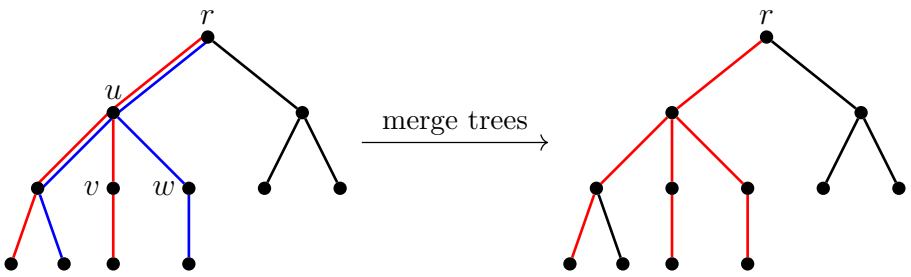


Figure 9.5: The tree on the left contains two different embeddings of the uniform tree with characteristic sequence $(1, 2, 1)$, A_1 is red and A_2 is blue. To create the uniform tree B with characteristic sequence $(1, 3, 1)$, which is shown in red on the right side, we keep the path to u, all of A_1 below u, and the remaining subtrees of A_2.

all k subtrees must be the same uniform tree of at most $\left\lceil \frac{n}{k} \right\rceil$ vertices. This leads to

$$a(n+1) = 1 + \sum_{k=1}^{n} a\left(\left\lceil \frac{n}{k} \right\rceil\right).$$

A uniform tree is characterised by the number of children of the nodes in every layer.

Definition 9.8 (Characteristic sequence)**.** The *characteristic sequence* of a uniform tree T of height h is a sequence

$$(t_0, \ldots, t_{h-1})$$

where $t_i > 0$ is the number of children of a node at depth i. ♢

Uniform trees having the same characteristic sequence are isomorphic.

We will now put an order on the characteristic sequences of the same length.

Definition 9.9. A characteristic sequence $(a_0, \ldots, a_{h-1})$ is larger than a characteristic sequence $(b_0, \ldots, b_{h-1})$ when there exists an $0 \leq i < h$ such that $a_i > b_i$ and $a_j = b_j$ for all $i < j < h$. ♢

Lemma 9.10. *Given a rooted tree T, if a uniform tree A can be embedded in T in two different ways, there exists a uniform tree B with a larger characteristic sequence that can be embedded in T.*

Proof. Suppose that A_1 and A_2 are two different rooted subtrees of T that are isomorphic to A. There must be a pair of vertices $u, v \in T$ such that v is a child of u, u lies in both A_1 and A_2, and v is only present in A_1. Since u has the same number of children in A_2 as in A_1, there must be some other child w of u such that $w \in A_2$ and $w \notin A_1$. These are shown in **??**. We can construct B as the rooted tree consisting of the path from r to u, together with the subtrees of A_1 rooted at u and the subtrees of A_2 rooted at u that are disjoint from A_1. It is easy to see that B is uniform:

Let d be the depth of u. For all depths d' at most d there is only a single vertex in level d', so that the tree is trivially uniform at those levels. At levels d' larger

than d, we merged two uniform trees with the same characteristic sequence, so that the characteristic sequence of B is equal to that of A. Since B has more children in u then A_1 or A_2, we have $b_d > a_d$ and $b_{d'} = a_{d'}$ for $d' > d$. Thus, the characteristic sequence of B is larger than the characteristic sequence of A. □

Corollary 9.11. *The above lemma implies that the uniform tree with the largest characteristic sequence that can be embedded in a rooted tree T has a unique embedding.*

Now we prove the most important result of [7].

Proposition 9.12. *The number of pairwise nonisomorphic uniform trees embedded in a tree T equals the number of vertices of T.*

Proof. We will prove this by induction on the number of vertices of T. When T has only a single vertex, the statement is true. When T is larger, assume that M is the uniform tree embedded in T with maximal characteristic sequence. Let h be the height of T. Then the height of M will be h too. Let x be a vertex of M at level h. We will show that the number of uniform trees that can be embedded in $T' = T - x$ is exactly one less than the number of uniform trees in T.

Since there exists only one embedding of M in T, and $x \in M$, M can not be embedded in T'. It remains to show that all other uniform trees that can be embedded in T can also be embedded in T'. Thus, suppose a given uniform tree A different from M can be embedded in T, and assume one such embedding. When $x \notin A$, A can also be embedded in T' and we are done. When $x \in A$, the path from the root r to x must lie completely in A and the height of A will be h. Let $(a_0, \ldots, a_{h-1})$ and $(m_0, \ldots, m_{h-1})$ the characteristic sequences of A and M. Because M is the maximal uniform tree in T and A has a smaller characteristic sequence, we must have $a_d < m_d$ with $a_i = m_i$ for $d < i < h$ for some $0 \leq d < h$. Now consider the vertex u on the path from r to x at depth d, which lies in both M and A. The uniform tree with characteristic sequence $(m_d, m_{d+1}, \ldots, m_{h-1})$ can be embedded in the subtree S of T rooted at u. All m_d subtrees of M at u have characteristic sequence $(m_{d+1}, \ldots, m_{h-1})$. Also, all a_d subtrees of A at u have the same characteristic sequence, because $a_i = m_i$ for $i > d$. Because A has fewer subtrees at u than M, we can choose a new embedding A' of A in T that is equal to A, except that in u we do not use the subtree of M that contains x, but instead use another subtree of u.

Thus, we can construct an embedding of A in $T - x$ that does not contain x for all uniform trees $A \neq M$ that can be embedded in T. By induction, we conclude that the number of nonisomorphic uniform trees embedded in T equals the number of vertices of T. □

Theorem 9.13 (Size of minimal rooted-universal tree)**.** *We have $r(1) = 1$ and*

$$r(n+1) = 1 + \sum_{i=1}^{n} r\left(\left\lceil \frac{n}{i} \right\rceil\right).$$

Proof. The construction of R_n gives an upper bound $r(n) \leq r'(n)$.

Because all rooted trees on at most n vertices are embedded in a minimal rooted-universal tree, all uniform trees on at most n vertices must be embedded in a minimal rooted-universal tree as well. Hence, the above proposition implies that

$$r(n) \geq a(n).$$

Since $a(n) = r'(n)$, we must have $a(n) = r(n) = r'(n)$. □

Notation 8: Minimal universal trees

Minimal rooted-universal tree	R_n
Size of minimal rooted-universal tree	$r(n) = \lvert R_n \rvert$

10 Uniform indices on refinements

In this section we prove that the following theorem

Theorem 10.1 (Uniform indices)**.** *When we are given a refinement $H = \mathbf{R}(G)$ of G and a finite harmonic morphism φ to a tree T, there exists a finite harmonic morphism $\tilde{\varphi}$ from a refinement $\widetilde{H}$ to T with degree at most* $\deg\varphi$*, such that the index of the edges in $\mathbf{R}^G(e)$ is constant, for every edge e in G.*

Proof. For every edge in G, we will modify H to satisfy the stated requirement. Consider $e = xy \in E(G)$ and $\mathbf{R}^G(e)$ in H. Suppose that r is the minimal index on the path from x to y in $\mathbf{R}^G(e)$. Let the path from x to y be given by $x = x_0, \ldots, y = x_k$. Denote $e_x = x_0x_1$ and $e_y = x_{k-1}x_k$. We have $r_\varphi(e_x) \geq r$ and $r_\varphi(e_y) \geq r$. This is show in the upper half of Figure 10.1. Now replace $\mathbf{R}^G(e)$ by $T_{\varphi(x)\varphi(y)}$, the part of T between $\varphi(x)$ and $\varphi(y)$. We assign index r to all the edges added in this way. The harmonic property is satisfied in all vertices in the newly added subgraph, apart form x and y. In x, add $T_{\varphi(x)}(y)$ as a new subtree with index $r_\varphi(e_x) - r$ on all edges if this quantity is at least 1. Otherwise, the harmonic property was already satisfied at x. Analogously, add $T_{\varphi(y)}(x)$ as a subtree to y with index $r_\varphi(e_y) - r$ if this is at least 1. We define $\tilde{\varphi}$ equal to φ on all of H except $\mathbf{R}^G(e)$. On the three newly added subgraphs, we let $\tilde{\varphi}$ be the identity.

Because of the way we have replaced these subgraphs, $\tilde{\varphi}$ is again a finite harmonic morphism. We still have to show that the degree of $\tilde{\varphi}$ is at most the degree of φ. We do this by looking at the changes in

$$\sum_{u\in[x]} m_\varphi(u).$$

By construction, $m_\varphi(x)$ is the same in H and $\widetilde{H}$. Because we added the $T_{\varphi(y)}(x)$ subtrees in $\widetilde{H}$, we added a new term of $r_\varphi(e_y) - r$ to the sum. We also removed $\mathbf{R}^G(e)$ from H. Every time $r_\varphi(x_ix_{i+1})$ is larger than $r_\varphi(x_{i-1}x_i)$ on the path from x to y, there must be one or more x_i-subtrees in H going out of x_i that covers x. The sum of the indices on these subtrees wil be at least $r_\varphi(x_ix_{i+1}) - r_\varphi(x_{i-1}x_i)$. Because

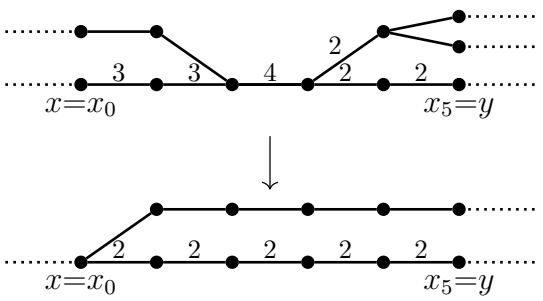


Figure 10.1: This is an example of the replacement we do in H in the proof of Theorem 10.1. We only show the path from x to y, which is the image of a single edge e in G. We replace all indices on the path by the minimum index of $r = 2$. Then, we add a subtree at x to make up for the difference with the index of x_0x_1. It can be seen that the degree of the new harmonic morphism will one lower than before.

r was chosen as the minimal index on the edges between x and y, the sum of these (positive) differences will be at least $r_\varphi(e_y) - r$. Thus, the sum decreases with *at least* $r_\varphi(e_y) - r$, which compensates for the increase by this amount.

Thus, we conclude that $\tilde{\varphi}$ is a finite harmonic morphism with degree not larger than $\deg \varphi$. □

11 Upper bound on the indices

In this section we prove that it suffices to consider finite harmonic morphisms with indices (both $r_\varphi(e)$ and $m_\varphi(u)$) at most $|E(G)|$. We start with an upper bound of $|E(H)|$, and later use a similar argument for the $|E(G)|$ bound.

Lemma 11.1. *If we are given a finite harmonic morphism $\varphi : H \to T$, there exists a finite harmonic morphism $\tilde{\varphi} : H \to T$ with $\deg \tilde{\varphi} \leq \deg \varphi$ and $m_{\tilde{\varphi}}(u) \leq |E(H)|$ for all $u \in V(H)$.*

We first need the following lemma.

Lemma 11.2. *When we are given a finite harmonic morphism $\varphi : H \to T$ and a vertex $u \in V(H)$ with index $m = m_\varphi(u)$, there exist m subtrees $T_1, \ldots, T_m$ of H such that*

- *$u \in V(T_i)$ for all i;*
- *$\varphi|_{T_i}$ is a bijection from T_i to T;*
- *for every edge $f \in E(H)$, the number of trees such that $f \in T_i$ is at most $r_\varphi(f)$.*

Proof. We will prove this using a greedy argument. Start the construction of T_1 by choosing u. Let $u' = \varphi(u)$ be the image of u in T. Now, repeated the following process for all vertices in $V(T)\backslash\{u'\}$.

Choose a vertex $y' \in T$ such that y' is a neighbour of some vertex $x' \in V(\varphi(T_1))$, but y' is not yet covered by T_1. These variables are show in Figure 11.1. Let x be the preimage of x' in T_1. There must be some edge $xy \in E(H)$ such that $\varphi(y) = y'$, because φ is harmonic. Thus, extend T_1 by adding the edge xy and the vertex y to it.

If we repeat this process until T_1 covers all of T, we know φ restricted to T_1 is a bijection from T_1 to T, as required.

Now we construct the other trees T_i in a similar way. We have to be a bit careful to satisfy the third requirement in the lemma. It is sufficient to show that we can always choose vertex y in such a way that at most $r_\varphi(xy)$ trees contain xy.

Thus, consider vertices x and y' again. If $x = u$, we know that the sum of the indices of the edges from x to $\varphi^{-1}(y')$ is $m_\varphi(x)$. Since $i \leq m_\varphi(x)$, we can always choose some vertex $y' \in \varphi^{-1}(y)$ such that xy was not covered $r_\varphi(xy)$ time yet.

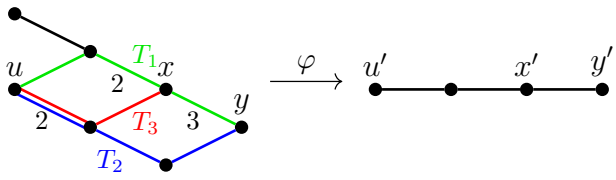


Figure 11.1: The left of this image shows an example of the construction of the trees in a graph H in Lemma 11.2. Only indices different from 1 are given. The red and green trees are already done. The next edge we will add to the red tree is xy. The right part shows the morphism from H to T, which is just a vertical projection.

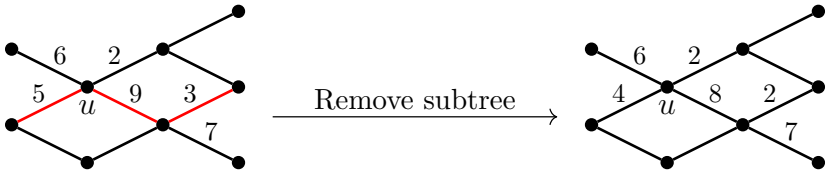


Figure 11.2: The left shows an example of a graph H with a vertex u. Only indices greater than 1 are shown. All edges in the red subgraph have index at least 2, so it may be removed from H, like in the proof of Lemma 11.1 below. This results in the graph on the right. Note that we assume that φ is the harmonic morphism corresponding to a vertical projection of the vertices and edges.

Now let $x \neq u$. There might already be some other trees containing x. However, we also know that the current tree T_i has an edge, say wx, going to x. Now, let c be the total number of edges in the trees T_1 till T_i from a vertex in $[w]$ to x. The number of edges leaving x in the direction of y' is only $c-1$, because each of the trees T_j with $j < i$ has a single edge from x to $\varphi^{-1}(y')$, but T_i has no such edge. Because φ is harmonic, the sum of the indices from x to $\varphi^{-1}(y')$ must be equal to the same sum for $[w]$. Because the latter sum is at least c, we know that we can always choose a vertex $y \in \varphi^{-1}(y')$ such that xy is not yet covered $r_\varphi(xy)$ times. Hence, we extend T_i by adding y and the edge xy. □

Proof of Lemma 11.1. Suppose that there is an vertex $u \in E(H)$ with index $m_\varphi(u) > |E(H)|$. Write $m = m_\varphi(u)$. We will then show that we can modify φ in such a way that this index decreases and $\deg \tilde{\varphi} \leq \deg \varphi$.

Let $T_1, \ldots, T_m$ be the m subtrees of H we obtain by applying the previous lemma on the vertex u. If there is an i such that the index of all edges in T_i is at least 2, we can decease the index of all edges in T_i by 1, which is explained below. Suppose to the contrary that all trees T_i contain an edge with index 1. Each of those edges can only occur in one of the T_i, because of property three in Lemma 11.2. Hence, there must be m edges with index 1 in H. However, we know that H has only m edges in total. Since H has only $|E(H)| < m$ edges in total, there can not be m edges with an index of 1, and thus we have a contradiction. Hence, there will always be a tree T_i with all indices at least 2.

Now, let T_i be a tree with all indices at least 2, such as the red subtree in Figure 11.2. After decreasing the index of all edges in T_i by 1, we obtain a new morphism $\tilde{\varphi}$. This is again a finite harmonic morphism, because for each vertex $u \in V(T_i)$ and edge e', we have

$$\sum_{e \in \tilde{\varphi}^{-1}(e') \cap E(u)} r_{\tilde{\varphi}}(e) = \sum_{e \in \varphi^{-1}(e') \cap E(u)} r_\varphi(e) - 1,$$

and thus $m_{\tilde{\varphi}}(u) = m_\varphi(u) - 1$ when $u \in V(T_i)$. When $u \notin V(T_i)$, nothing changes. Note that $\deg \tilde{\varphi} = \deg \varphi - 1$, so the degree of the new finite harmonic morphism is not larger than what we started with. □

Corollary 11.3. *Since we may now assume that $m_\varphi(u) \leq |E(H)|$ while searching the optimal finite harmonic morphisms, we may also assume that $r_\varphi(e) \leq |E(H)|$ for all edges e, because $r_\varphi(uv) \leq m_\varphi(u) \leq |E(H)|$.*

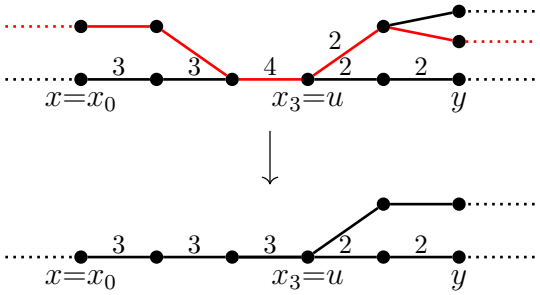


Figure 11.3: Here we show $\mathbf{R}^G(e)$, the subtree of H that $e = xy$ refined to. In case 1 of the proof of Theorem 11.4 we construct the red subtree. We may remove this subtree from H, because it does not contain any index 1 edge on the path from x to y. Again, we assume a vertical projection to a tree.

The previous corollary implies that in order to find the optimal finite harmonic morphism in the algorithm we describe, we only have to consider indices at most $|E(H)|$. This is not a bounded quantity however, because H can be an arbitrary refinement of G. Therefore, we would like an upper bound of $|E(G)|$. This is established in the following lemma, which uses a similar proof.

Theorem 11.4 (Upper bound on indices)**.** *When we are given a nontrivial graph G, refinement H and finite harmonic morphism φ to a tree T, we can construct a new finite harmonic morphism $\tilde{\varphi}$ from $\widetilde{H}$ to T with degree at most $\deg\varphi$ such that $m_{\tilde{\varphi}}(u) \leq |E(G)|$ for all vertices u in $\widetilde{H}$.*

Proof. For every edge $e = uv \in E(G)$, let $r(e)$ be the smallest index on the path from $\mathbf{R}(u)$ to $\mathbf{R}(v)$ in $\mathbf{R}^G(e)$, the subgraph in H created by e including u and v. Now, suppose there is a vertex u in H with $m = m_\varphi(u) > |E(G)| \geq 1$. We consider two cases.

Case 1: $u \notin V(G) \subset V(H)$. Let u be in the image of the edge $e = xy$ in G. If $m_\varphi(x) \geq m$ or $m_\varphi(y) \geq m$, we could choose $u = x$ or $u = y$, so that we are in the second case of this proof. Thus, suppose $m_\varphi(x) < m$ and $m_\varphi(y) < m$. We will construct a subtree T_1 of H starting in u just like in Lemma 11.2. We do however impose some extra restrictions. Since T_1 is the first tree we can construct, and $m \geq 2$ because $m > |E(G)| \geq 1$, we can always avoid to choose an edge with index 1 on the path from x to y, because there is at most 1 such edge in every direction. Also, we can avoid covering x and y, because their indices are less than m. This is easily seen by considering a path $x = x_0,\ldots,x_k = u$. We know that $m_\varphi(x) < m = m_\varphi(x_k)$. Hence, there is an i such that x_i has an edge to a vertex in $[x_{i-1}]\backslash\{x_{i-1}\}$. Choosing this edge prevents us from covering x with T_1. A similar argument holds for y.

Now, T_1 is a subtree of $\mathbf{R}^G(e)$ isomorphic to T, and it does not contain any index 1 edges on the path from x to y. Hence, we can decrease the index of all edges in T_1 by 1. If the index of any edge becomes 0, we simply remove it. Also, vertices that become disconnected from all other vertices are removed. An example of this is shown in Figure 11.3. This always leaves a connected graph, since we will remove complete w-subtrees of H for several vertices w.

Case 2: $u \in V(G) \subset V(H)$. Now, consider $m = m_\varphi(u) > |E(G)|$ subtrees T_1, $\ldots$, T_m of H, starting in u, as given by Lemma 11.2. We start by proving that there

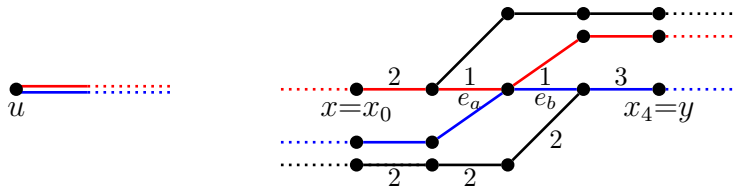


Figure 11.4: In case 2 of the proof of Theorem 11.4, we prove that a situation like the one shown here can not occur. The blue and red subtree both contain an index 1 edge on the path from x to y and both contain u. Intuitively, once the blue subgraph has detached from xy, there is no way it can reattach.

must be a tree T_i such that T_i contains no edges of index 1 on any path between vertices in $V(G) \subset V(H)$.

Suppose that all T_i contain at least one index 1 edge on a path between two vertices in $V(G) \subset V(H)$. As before, every index 1 edge can only be part of at most one tree T_i. If we can prove that in fact, all index 1 edges on any xy path in H with $x, y \in V(G)$ can be part of at most one tree T_i, we know that we need at least m such paths to have an index 1 edge in every tree T_i. Since the number of edges of G is strictly less than m, this can not happen, and there must be a tree T_i that does not contain an index 1 edge on a xy path. Then, we can remove that tree from H, which decreases the degree of φ by 1, resulting in $\widetilde{H}$ and $\tilde{\varphi}$.

It remains to be shown that on every path in H between two vertices $x, y \in V(G) \subset V(H)$, there can be at most 1 tree covering index 1 edges. Suppose to the contrary that two trees T_a and T_b both cover an index 1 edge. Let the path be given by

$$x = x_0, x_1, \ldots, x_k = y.$$

Suppose T_a covers edge e_a and T_b covers edge e_b. Obviously, $e_a \neq e_b$. Suppose that e_a comes before e_b. Then, T_a contains x but can not cover e_b, and thus, $y \notin V(T_a)$. Analogously, $x \notin V(T_b)$ while $y \in V(T_b)$. Now, suppose without loss of generality that in T, the distance from $\varphi(u)$ is to $\varphi(x)$ is less than the distance to $\varphi(y)$. We will show that a tree like T_b can not be constructed by the procedure from Lemma 11.2. Here, a "tree like T_b" is a tree not containing x which *does* contain u and an edge on the path from x to y.

Please have a look at Figure 11.4 for an example. We know that $u, y \in V(T_b)$. Furthermore, $e_b \in E(T_b)$. Thus, the edge $\varphi(e_b)$ in T is already covered by T_b. Since $\varphi|_{T_b}$ must be a bijection from T_b to T, every path from u to y in T_b must go through e_b, and hence through x. Thus, a tree like T_b can not occur. □

Like before, we can derive an upper bound on the indices of the edges of H using this theorem.

Corollary 11.5. *Given a finite harmonic morphism from a refinement H of G to a tree T, there exits a finite harmonic morphism $\tilde{\varphi}$ from a refinement $\widetilde{H}$ of G to H with degree at most* $\deg \varphi$ *and $r_{\varphi}(e) \leq |E(G)|$ for all $e \in E(G)$.*

Remark 11.6. Note that for the purposes of our algorithm, this corollary could also be stated as:

There exists a finite harmonic morphism of degree $\operatorname{sgon}(G)$ *such that all indices $r_{\varphi}(e)$ are at most $|E(G)|$.*

In the algorithm, we could also use the bound on the indices of the vertices while iterating over all possible assignments of indices of the edges of H, but in practice, just bounding the indices of the edges themselves is easier. It is always possible to skip a certain function r when the index at one of the vertices is more than $|E(G)|$.

Now that we have proven this upper bound on the indices, I would like to state a theorem of Cornelissen et al. [6] which implies an even better bound. They prove this using a similar theorem in algebraic geometry, proven by Kleiman and Laksov in [8].

Theorem 11.7 (Theorem B from [6])**.** *For any graph G, we have the Brill-Noether bound,*

$$\operatorname{sgon}(G) \leq \left\lfloor \frac{g+3}{2} \right\rfloor.$$

Here, g is the genus *of G. For connected graphs, this is defined as*

$$g = |E(G)| - |V(G)| + 1.$$

Remark 11.8. In their paper, they need $g \geq 2$. Here, we assume G is connected. When $g = 0$, G is a tree, and $\operatorname{sgon}(G) = 1$. When $g = 1$, G has exactly one cycle, so $\operatorname{sgon}(G) = 2$. Both these values satisfy the bound given in the theorem, so we do not impose the restriction $g \geq 2$.

We conclude this section with the corresponding bound on the indices, which is much better than our own bound of $|E(G)|$. Note however, that no graph theoretic proof of this result is known. Theorem 11.4 might be a first step towards such a proof.

Corollary 11.9. *Given a graph G, there exists a finite harmonic morphism of degree* $\operatorname{sgon}(G)$ *such that all indices $r_\varphi(e)$ of the edges in H are at most*

$$r_\varphi(e) \leq \left\lfloor \frac{|E(G)| - |V(G)| + 3}{2} \right\rfloor.$$

12 Discussion

The algorithm runs in finite time, but it is very slow. However, there is room for further optimizations, some of which I explain here.

Currently, the optimal universal trees we use are unrooted-universal trees. No research has been done yet on minor-universal trees, which is all we actually need. They are not much smaller for small n, but in practice, all small improvements speed up the algorithm. Furthermore, it might be faster to make a list of small minor-universal trees using a brute force algorithm, instead of using larger universal trees.

The loop over indices could be improved as well. Currently, we loop all the way up to the upper bound of $\lfloor (m-n+4)/2 \rfloor$, but in practice, this is to high for most graphs. We could also use a dynamic upper bound that is at most the lowest degree we have found so far. This should speed up the calculations for graphs with a very low stable gonality.

Finally, there is no graph theoretic proof of the bound on the stable given above. I did prove that we may assume that $r_\varphi \leq m$, but this only gives a bound on the indices, not on the stable gonality itself.

References

[1] Matthew Baker and Serguei Norine. "Harmonic morphisms and hyperelliptic graphs". In: *Int. Math. Res. Not. IMRN* 15 (2009), pp. 2914–2955. ISSN: 1073-7928. DOI: 10.1093/imrn/rnp037. URL: http://dx.doi.org/10.1093/imrn/rnp037.

[2] Lucia Caporaso. "Gonality of algebraic curves and graphs". In: *Algebraic and complex geometry*. Vol. 71. Springer Proc. Math. Stat. Springer, Cham, 2014, pp. 77–108. DOI: 10.1007/978-3-319-05404-9_4. URL: http://dx.doi.org/10.1007/978-3-319-05404-9_4.

[3] F. R. K. Chung and R. L. Graham. "On graphs which contain all small trees". In: *J. Combinatorial Theory Ser. B* 24.1 (1978), pp. 14–23.

[4] F. R. K. Chung, R. L. Graham, and D. Coppersmith. "On trees containing all small trees". In: *The theory and applications of graphs (Kalamazoo, Mich., 1980)*. Wiley, New York, 1981, pp. 265–272.

[5] F. R. K. Chung, R. L. Graham, and N. Pippenger. "On graphs which contain all small trees. II". In: *Combinatorics (Proc. Fifth Hungarian Colloq., Keszthely, 1976), Vol. I*. North-Holland, Amsterdam, 1978, 213–223. Colloq. Math. Soc. János Bolyai, 18.

[6] Gunther Cornelissen, Fumiharu Kato, and Janne Kool. "A combinatorial Li-Yau inequality and rational points on curves". In: *Math. Ann.* 361.1-2 (2015), pp. 211–258. ISSN: 0025-5831. DOI: 10.1007/s00208-014-1067-x. URL: http://dx.doi.org/10.1007/s00208-014-1067-x.

[7] M. K. Gol′dberg and È. M. Livšic. "Minimal universal trees". In: *Mat. Zametki* 4 (1968), pp. 371–379. ISSN: 0025-567X.

[8] Steven L. Kleiman and Dan Laksov. "On the existence of special divisors". In: *Amer. J. Math.* 94 (1972), pp. 431–436. ISSN: 0002-9327.

[9] Apostolos Syropoulos. "Mathematics of multisets". In: *Multiset processing*. Vol. 2235. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, pp. 347–358. DOI: 10.1007/3-540-45523-X_17. URL: http://dx.doi.org/10.1007/3-540-45523-X_17.

[10] Thomas Wolle and Hans L. Bodlaender. *A Note on Edge Contraction*. 2004.