Simplicial cycles and the computation of simplicial trees

Massimo Caboara[*] Sara Faridi[†] Peter Selinger[‡]

Abstract

We generalize the concept of a cycle from graphs to simplicial complexes. We show that a simplicial cycle is either a sequence of facets connected in the shape of a circle, or is a cone over such a structure. We show that a simplicial tree is a connected cycle-free simplicial complex, and use this characterization to produce an algorithm that checks in polynomial time whether a simplicial complex is a tree. We also present an efficient algorithm for checking whether a simplicial complex is grafted, and therefore Cohen-Macaulay.

1 Introduction

The main goal of this paper is to demonstrate that it is possible to check, in polynomial time, if a monomial ideal is the facet ideal of a simplicial tree.

Facet ideals were introduced in [F1] (generalizing results in [Vi1] and [SVV] on edge ideals of graphs) as a method to study square-free monomial ideals. The idea is to associate a simplicial complex to a square-free monomial ideal, where each facet (maximal face) of the complex is the collection of variables that appear in a monomial in the minimal generating set of the ideal (see Definition 2.4). The ideal will then be called the "facet ideal" of this simplicial complex. A special class of simplicial complexes are called "simplicial trees" (Definition 2.9). The definition of a simplicial tree is a generalization of the concept of a graph-tree. Facet ideals of trees have many properties; for example, they have normal and Cohen-Macaulay Rees rings [F1]. Finding such classes of ideals is in general a difficult problem. Simplicial trees also have strong Cohen-Macaulay properties: their facet ideals are always sequentially Cohen-Macaulay [F2], and one can determine under precisely what combinatorial conditions on the simplicial tree the facet ideal is Cohen-Macaulay [F3]. In [F4] it is shown that the theory is not restricted to square-free monomial ideals; via polarization, one can extend

[*]Department of Mathematics, University of Pisa, caboara@dm.unipi.it.

[†]Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada, faridi@mathstat.dal.ca. Research supported by NSERC.

[‡]Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada, selinger@mathstat.dal.ca. Research supported by NSERC.

many properties of facet ideals to all monomial ideals. All these properties, and many others, make simplicial trees useful from an algebraic point of view.

But how does one determine if a given square-free monomial ideal is the facet ideal of a simplicial tree? In Section 4, we give a characterization of trees that shows this can be done in polynomial time. This characterization is based on a careful study of the structure of cycles in Section 3. The study of simplicial cycles is indeed interesting in its own right. In graph theory, the concepts of a tree and of a cycle are closely linked to each other: a tree is a connected graph that does not contain a cycle, and a cycle is a minimal graph that is not a tree. Generalizing to the simplicial case, we use the latter property, together with the existing definition of a simplicial tree, to define the concept of a simplicial cycle. We then prove the remarkable fact that a simplicial cycle is either a sequence of facets connected in the shape of a circle, or a cone over such a structure. This in turns yields an alternative characterization of trees, given in Section 4.

This result enables us to produce a polynomial time algorithm to decide whether a given simplicial complex is a tree. The algorithm itself is introduced in Section 5, where the complexity and optimizations are also discussed. Section 6 focuses on the algebraic properties of facet ideals: in Section 6.1 we discuss a method of adding generators to a square-free monomial ideal (or facets to the corresponding complex) so that the resulting facet ideal is Cohen-Macaulay. This method is called "grafting" a simplicial complex. For simplicial trees, being grafted and being Cohen-Macaulay are equivalent conditions [F3]. We then introduce an algorithm that checks whether or not a given simplicial complex is grafted and discuss its complexity.

Implementations. The algorithms described in this paper have first been coded using CoCoAL, the programming language of the CoCoA system (please see http://cocoa.dima.unige.it/). These prototypical implementations can be downloaded from [CFS2]. Much more efficient (but less user friendly) C++ implementations have been developed for several versions of Algorithm 5.1 using the CoCoALib framework (http://cocoa.dima.unige.it/cocoalib/). The C++ code is also available at the website [CFS2].

2 Simplicial complexes and trees

We define the basic notions related to facet ideals. More details and examples can be found in [F1, F3].

Definition 2.1 (Simplicial complex, facet). A *simplicial complex* Δ over a finite set of vertices V is a collection of subsets of V, with the property that if $F \in \Delta$ then all subsets of F are also in Δ. An element of Δ is called a *face* of Δ, and the maximal faces are called *facets* of Δ.

Since we are usually only interested in the facets, rather than all faces, of a simplicial complex, it will be convenient to work with the following definition:

Definition 2.2 (Facet complex). A *facet complex* over a finite set of vertices V is a set Δ of subsets of V, such that for all $F, G \in \Delta$, $F \subseteq G$ implies $F = G$. Each $F \in \Delta$ is called a *facet* of Δ.

Remark 2.3 (Equivalence of simplicial complexes and facet complexes). The set of facets of a simplicial complex forms a facet complex. Conversely, the set of subsets of the facets of a facet complex is a simplicial complex. This defines a one-to-one correspondence between simplicial complexes and facet complexes. In this paper, we will work primarily with facet complexes.

We define facet ideals, giving a one-to-one correspondence between facet complexes (or, equivalently, simplicial complexes) and square-free monomial ideals.

Definition 2.4 (Facet ideal of a facet complex, facet complex of an ideal).

- Let Δ be a facet complex over a vertex set $\{v_1, \dots, v_n\}$. Let k be a field, and let $R = k[x_1, \dots, x_n]$ be the polynomial ring with indeterminates $x_1, \dots, x_n$. The *facet ideal of* Δ is defined to be the ideal of R generated by all the square-free monomials $x_{i_1} \dots x_{i_s}$, where $\{v_{i_1}, \dots, v_{i_s}\}$ is a facet of Δ. We denote the facet ideal of Δ by $\mathcal{F}(\Delta)$.
- Let $I = (M_1, \dots, M_q)$ be an ideal in the polynomial ring $k[x_1, \dots, x_n]$, where k is a field and $M_1, \dots, M_q$ are square-free monomials in $x_1, \dots, x_n$ that form a minimal set of generators for I. The *facet complex of* I is defined to be $\delta_{\mathcal{F}}(I) = \{F_1, \dots, F_q\}$, where for each i, $F_i = \{v_j \mid x_j | M_i, 1 \leqslant j \leqslant n\}$.

From now on, we often use $x_1, \dots, x_n$ to denote both the vertices of Δ and the variables appearing in $\mathcal{F}(\Delta)$. We also sometimes ease the notation by denoting facets by their corresponding monomials; for example, we write xyz for the facet $\{x, y, z\}$.

We now generalize some notions from graph theory to facet complexes. Note that a graph can be regarded as a special kind of facet complex, namely one in which each facet has cardinality 2.

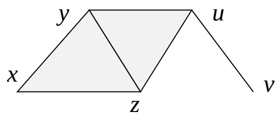
Definition 2.5 (Path, connected facet complex). Let Δ be a facet complex. A sequence of facets $F_1, \dots, F_n$ is called a *path* if for all $i = 1, \dots, n-1$, $F_i \cap F_{i+1} \neq \emptyset$. We say that two facets F and G are *connected* in Δ if there exists a path $F_1, \dots, F_n$ with $F_1 = F$ and $F_n = G$. Finally, we say that Δ is *connected* if every pair of facets is connected.

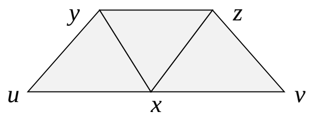
Notation 2.6. If F, G and H are facets of Δ, $H \leqslant_F G$ means that $H \cap F \subseteq G \cap F$. The relation $\leqslant_F$ defines a preorder (reflexive and transitive relation) on the facet set of Δ.

Definition 2.7 (Leaf, joint). Let F be a facet of a facet complex Δ. Then F is called a *leaf* of Δ if either F is the only facet of Δ, or else there exists some $G \in \Delta \setminus \{F\}$ such that for all $H \in \Delta \setminus \{F\}$, we have $H \leqslant_F G$. The facet G above is called a *joint* of the leaf F if $F \cap G \neq \emptyset$.

It follows immediately from the definition that every leaf F contains at least one *free vertex*, i.e., a vertex that belongs to no other facet.

Example 2.8. In the facet complex $\Delta = \{xyz, yzu, uv\}$, xyz and uv are leaves, but yzu is not a leaf. Similarly, in $\Delta' = \{xyu, xyz, xzv\}$, the only leaves are xyu and xzv.

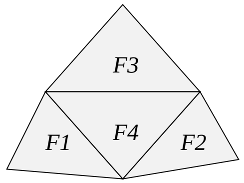
$\Delta =$ 

$\Delta' =$ 

Definition 2.9 (Forest, tree). A facet complex Δ is a *forest* if every nonempty subset of Δ has a leaf. A connected forest is called a *tree* (or sometimes a *simplicial tree* to distinguish it from a tree in the graph-theoretic sense).

It is clear that any facet complex of cardinality one or two is a forest. When Δ is a graph, the notion of a simplicial tree coincides with that of a graph-theoretic tree.

Example 2.10. The facet complexes in Example 2.8 are trees. The facet complex pictured below has three leaves F_1, F_2 and F_3; however, it is not a tree, because if one removes the facet F_4, the remaining facet complex has no leaf.



The following property is proved in [F3, Lemma 4.1]:

Lemma 2.11 (A tree has two leaves). *Every tree with two or more facets has at least two leaves.* $\square$

3 Cycles

In this section, we define a simplicial cycle as a minimal complex without leaf. This in turns characterizes a simplicial tree as a connected cycle-free facet complex. We further show that cycles possess a particularly simple structure: each cycle is either equivalent to a "circle" of facets with disjoint intersections, or to a cone over such a circle.

Definition 3.1 (Cycle). A nonempty facet complex Δ is called a *cycle* if Δ has no leaf but every nonempty proper subset of Δ has a leaf.

Equivalently, Δ is a cycle if Δ is not a forest, but every proper subset of Δ is a forest. If Δ is a graph, Definition 3.1 coincides with the graph-theoretic definition of a cycle. The next two remarks are immediate consequences of the definitions of cycle and forest:

Remark 3.2. A cycle is connected.

Remark 3.3. A facet complex is a forest if and only if it does not contain a cycle.

In the remainder of this section, we provide a complete characterization of the structure of cycles.

Definition 3.4 (Strong neighbor). Let Δ be a facet complex and $F, G \in \Delta$. We say that F and G are *strong neighbors*, written $F \sim_\Delta G$, if $F \neq G$ and for all $H \in \Delta$, $F \cap G \subseteq H$ implies $H = F$ or $H = G$.

The relation $\sim_\Delta$ is symmetric, i.e., $F \sim_\Delta G$ if and only if $G \sim_\Delta F$. Note that if Δ has more than two facets, then $F \sim_\Delta G$ implies that $F \cap G \neq \emptyset$.

Example 3.5. For the facet complex Δ' in Example 2.8, $xyu \not\sim_{\Delta'} xzv$, as their intersection x lies in the facet xyz. However, $xyz \sim_{\Delta'} xzv$ and similarly $xyz \sim_{\Delta'} xyu$.

Remark 3.6. Suppose Δ is a facet complex, and $\Delta' \subseteq \Delta$. Let $F, G \in \Delta'$. If $F \sim_\Delta G$, then $F \sim_{\Delta'} G$. The converse is not in general true.

Remark 3.7. We have $F \sim_\Delta G$ if and only if G is strictly maximal with respect to $\leqslant_F$ on $\Delta \setminus \{F\}$, i.e., for all $H \neq F$, $G \leqslant_F H$ implies $G = H$. This is a simple restatement of the definition.

It turns out that a cycle can be described as a sequence of strong neighbors. The following lemma follows directly from Definition 3.4.

Lemma 3.8. *If Δ is a facet complex with distinct facets F, G_1, G_2 such that $F \sim_\Delta G_1$ and $F \sim_\Delta G_2$, then F is not a leaf of Δ.*

Proof. If F is a leaf, there exists a facet $H \neq F$ such that $G_1 \leqslant_F H$ and $G_2 \leqslant_F H$, which by Remark 3.7 implies that $G_1 = G_2 = H$, a contradiction. $\square$

Corollary 3.9. *Let Δ be a facet complex, and let $F_1, \dots, F_n$ be distinct facets with $n \geqslant 3$, such that $F_1 \sim_\Delta F_2 \sim_\Delta \dots \sim_\Delta F_n \sim_\Delta F_1$. Then $\{F_1, \dots, F_n\}$ has no leaf.*

Proof. This follows directly from Remark 3.6, and Lemma 3.8. $\square$

Lemma 3.10. *Suppose Δ is a facet complex and $F, G \in \Delta$. If F is a leaf of $\Delta \setminus \{G\}$, but not a leaf of Δ, then $F \sim_\Delta G$.*

Proof. Suppose H is some facet such that $F \cap G \subseteq H$, but $H \neq F$ and $H \neq G$. Since F is a leaf for $\Delta \setminus \{G\}$, there exists a facet $H' \in \Delta \setminus \{G\}$ such that $L \cap F \subseteq H'$ for all $L \in D \setminus \{F, G\}$, and so $F \cap H \subseteq H'$. But now we have $F \cap G \subseteq F \cap H \subseteq H'$, which implies that F is a leaf of Δ, a contradiction. $\square$

Proposition 3.11 (A cycle is a sequence of strong neighbors). *Suppose Δ is a cycle, and let $n = |\Delta|$. Then $n \geqslant 3$, and the facets of Δ can be enumerated as $\Delta = \{F_1, \dots, F_n\}$ in such a way that*

$$F_1 \sim_\Delta F_2 \sim_\Delta \dots \sim_\Delta F_n \sim_\Delta F_1,$$

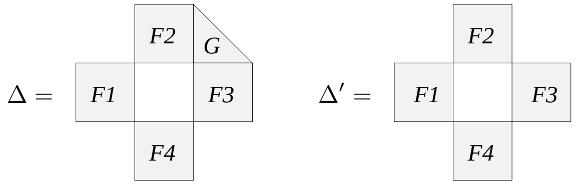
and $F_i \not\sim_\Delta F_j$ in all other cases, so that each facet is a strong neighbor of precisely two other facets.

Proof. First note that since Δ is not a forest, $n \geqslant 3$. We begin by showing that each facet has at least two distinct strong neighbors. Let $F \in \Delta$ be a facet. Since Δ is a cycle, $\Delta \setminus \{F\}$ is a tree. The subset $\Delta \setminus \{F\}$ also has cardinality at least two, and therefore has two distinct leaves, say G and H, by Lemma 2.11. Since neither G nor H are leaves of Δ (because Δ is a cycle), we have $F \sim_\Delta G$ and $F \sim_\Delta H$ by Lemma 3.10.

Now we can simply choose F_1 arbitrarily, then choose $F_2 \neq F_1$ such that $F_1 \sim_\Delta F_2$, then for every $i \geqslant 3$ choose F_i such that $F_{i-1} \sim_\Delta F_i$ and $F_i \neq F_{i-1}, F_{i-2}$. Since Δ is finite, there will be some smallest i such that $F_i = F_j$ for some $j < i$. Then $\Delta' = \{F_j, \dots, F_{i-1}\}$ has no leaf by Corollary 3.9, so $\Delta' = \Delta$. It follows that $j = 1$ and $i - 1 = n$. Finally, suppose that $F_k \sim_\Delta F_l$ for some $k \leqslant l - 2$, where $k > 1$ or $l < n$. Then $\{F_1, \dots, F_k, F_l, \dots, F_n\}$ has no leaf by Corollary 3.9, contradicting the fact that it is a tree. $\square$

The converse of Proposition 3.11 is not true.

Example 3.12. The facet complex Δ is not a cycle, as its proper subset Δ' (which is indeed a cycle) has no leaf. However, we have $F_1 \sim_\Delta F_2 \sim_\Delta G \sim_\Delta F_3 \sim_\Delta F_4 \sim_\Delta F_1$, and these are the only pairs of strong neighbors in Δ.



Lemma 3.13. *If Δ is a cycle, written as $F_1 \sim_\Delta F_2 \sim_\Delta \dots \sim_\Delta F_n \sim_\Delta F_1$, then for each i, $\Delta_i = \Delta \setminus \{F_i\}$ is a tree with exactly two leaves F_{i-1} and F_{i+1}, with joints F_{i-2} and F_{i+2}, respectively.*

Proof. We know that Δ_i is a tree, so it has at least two leaves. By Lemma 3.8 F_{i-1} and F_{i+1} are the only choices. By Remark 3.7 F_{i-2} is the only possible joint for F_{i-1}, and F_{i+2} is the only possible joint for F_{i+1}. $\square$

The following lemma will be fundamental for the classification of cycles.

Lemma 3.14. *Let Δ be a cycle with facets $F \neq G \in \Delta$. If $F \not\sim_\Delta G$, then $F \cap G \subseteq H$ for all $H \in \Delta$.*

Proof. We first prove the claim in the special case where $F \sim_\Delta H$. Indeed, since F is a strong neighbor of exactly two facets, there must be some $L \neq G, H$ such that $L \sim_\Delta F \sim_\Delta H$. Then Lemma 3.13 implies that H is a joint of F in the tree $\Delta \setminus \{L\}$, and therefore $F \cap G \subseteq H$, or equivalently, $F \leqslant_G H$.

Now consider the general case. By Proposition 3.11, the facets of Δ can be enumerated as $F_1 \sim_\Delta F_2 \sim_\Delta \ldots \sim_\Delta F_n \sim_\Delta F_1$. Assume, without loss of generality, that $F = F_1$ and $G = F_i$, where $2 < i < n$. By repeated applications of the special case above, we have

$$F \leqslant_G F_2 \leqslant_G \ldots \leqslant_G F_{i-1}.$$

In the other direction, we similarly have

$$F \leqslant_G F_n \leqslant_G F_{n-1} \leqslant_G \ldots \leqslant_G F_{i+1}.$$

Therefore, $F \cap G \subseteq F_j$ for $j = 1, \ldots, n$. □

Lemma 3.15. *Let Δ be a facet complex, and let*

$$A = \bigcap_{F \in \Delta} F \text{ and } \Delta' = \{F \setminus A \mid F \in \Delta\}.$$

Then Δ' is a facet complex. Moreover, Δ is a cycle if and only if Δ' is a cycle.

Proof. For each $F \in \Delta$, let $F' = F \setminus A$. Since Δ is a facet complex, we have $F \not\subseteq G$ for any two distinct facets $F, G \in \Delta$, which clearly implies $F' \not\subseteq G'$. So Δ' is a facet complex. Let Γ be any subset of Δ, and let $\Gamma' = \{F' \mid F \in \Delta\}$ be the corresponding subset of Δ'. Then for any triple of facets $F, G, H \in \Gamma$, we have $F \leqslant_H G \iff F' \leqslant_{H'} G'$. Therefore, Γ has a leaf if and only if Γ' has a leaf. It follows that Δ is a cycle if and only if Δ' is a cycle. □

Theorem 3.16 (Structure of a cycle). *Let Δ be a facet complex. Then Δ is a cycle if and only if Δ can be written as a sequence of strong neighbors $F_1 \sim_\Delta F_2 \sim_\Delta \ldots \sim_\Delta F_n \sim_\Delta F_1$ such that $n \geqslant 3$, and for all i, j*

$$F_i \cap F_j = \bigcap_{k=1}^{n} F_k \quad \text{if } j \neq i-1, i, i+1 \pmod{n}.$$

Proof. Let Δ be a cycle. Then by Proposition 3.11 and Lemma 3.14, Δ can be written as a sequence of strong neighbors with the desired properties.

Conversely, suppose that Δ is written as a sequence of strong neighbors $F_1 \sim_\Delta F_2 \sim_\Delta \ldots \sim_\Delta F_n \sim_\Delta F_1$ such that $F_i \cap F_j = \bigcap_{k=1}^n F_k$ if $j \neq i - 1, i, i+1 \pmod{n}$. By Lemma 3.15 we can without loss of generality assume that $\bigcap_{k=1}^n F_k = \emptyset$.

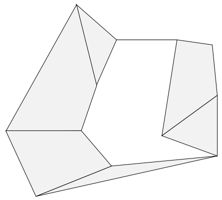
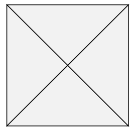
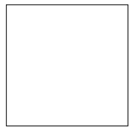
By Corollary 3.9, Δ has no leaf. Suppose Δ' is any nonempty proper subset of Δ. We need to show that Δ' has a leaf. Suppose $F_i \in \Delta'$ and $F_{i+1} \notin \Delta'$. There are two cases:

1. $F_{i-1} \notin \Delta'$. In this case, since $F_i \cap F_k = \emptyset$ for all $F_k \in \Delta' \setminus \{F_i\}$, F_i is a leaf.
2. $F_{i-1} \in \Delta'$. In this case, $F_i \cap F_k \subseteq F_{i-1}$ for all $F_k \in \Delta' \setminus \{F_i\}$, and so F_i is again a leaf, this time with F_{i-1} as a joint.

So Δ is a cycle and we are done. □

The implication of Theorem 3.16 is that a simplicial cycle has a very intuitive structure: it is either a sequence of facets joined together to form a circle in such a way that all intersections are pairwise disjoint (this is the case where the intersection of all the facets is the empty set in Theorem 3.16), or it is a cone over such a structure (Lemma 3.15).

Example 3.17. The facet complex Δ is a cycle. The facet complex Γ is a cycle and is also a cone over the cycle Γ'.

$\Delta =$  $\Gamma =$  $\Gamma' =$ 

4 Characterization of trees

We now consider the problem of deciding whether or not a given facet complex is a tree. We refer to this problem as the *decision problem for simplicial trees.*

Note that the naïve algorithm (namely, checking whether every non-empty subset has a leaf) is extremely inefficient: for a facet complex of n facets, there are $2^n - 1$ subsets to check. Also note that the definition of a tree is not inductive in any obvious way: for instance, attaching a single leaf to a tree need not yield a tree, as Example 2.10 shows. This seems to rule out an easy recursive algorithm.

Nevertheless, we demonstrate that the decision problem for simplicial trees can be solved efficiently. This is done via a characterization of trees given in this section.

Definition 4.1 (Paths and connectedness outside V). Let Δ be a facet complex, and let V be a set of vertices. We say that a sequence of facets $H_1, \ldots, H_n \in \Delta$ is a *path outside* V in Δ if for all $i = 1, \ldots, n-1$, $(H_i \cap H_{i+1}) \setminus V \neq \emptyset$. We say that two facets $F, G \in \Delta$ are *connected outside* V in Δ if there exists a path $H_1, \ldots, H_n$ outside V in Δ such that $H_1 = F$ and $H_n = G$.

Note that in case $V = \emptyset$, this coincides with the definition of connectedness from Definition 2.5.

Notation 4.2. If F, G_1, G_2 are three distinct facets of Δ, then we define $\Delta_F^{G_1,G_2}$ to be the following subset of Δ:

$$\Delta_F^{G_1,G_2} = \{H \in \Delta \mid H \cap F = G_1 \cap G_2\} \cup \{G_1, G_2\}.$$

Definition 4.3 (Triple condition). Let Δ be a facet complex. We say a triple of facets $\langle F, G_1, G_2 \rangle$ satisfies the *triple condition* if $G_1 \not\leqslant_F G_2$ and $G_2 \not\leqslant_F G_1$, and if G_1 and G_2 are connected outside F in the facet complex $\Delta_F^{G_1,G_2}$.

We note that the definitions of $\Delta_F^{G_1,G_2}$ and the triple condition have changed from an earlier version of this article [CFS1]; they have been simplified.

Example 4.4. Consider Δ in Example 3.12. Then the triple $\langle F_1, F_2, F_4 \rangle$ satisfies the triple condition. This is because $F_4 \not\leqslant_{F_1} F_2$ and $F_2 \not\leqslant_{F_1} F_4$. Moreover $\Delta_{F_1}^{F_2,F_4} = \{F_2, F_3, F_4, G\}$, and a path connecting F_2 and F_4 outside F_1 is F_2, F_3, F_4.

However, $\langle G, F_2, F_3 \rangle$ does not satisfy the triple condition, since $F_2 \leqslant_G F_3$ (and $F_3 \leqslant_G F_2$). Also $\Delta_G^{F_2,F_3} = \{F_2, F_3\}$, and F_2 and F_3 are not connected outside G.

Proposition 4.5 (A triple is part of a cycle). *Let Δ be a facet complex. A triple $\langle F, G_1, G_2 \rangle$ satisfies the triple condition if and only if there exists a cycle $\Delta' \subseteq \Delta$ such that $F, G_1, G_2 \in \Delta'$ and $G_1 \sim_{\Delta'} F \sim_{\Delta'} G_2$.*

Proof. Suppose $\langle F, G_1, G_2 \rangle$ satisfies the triple condition. Then by definition, $G_1 \not\leqslant_F G_2$ and $G_2 \not\leqslant_F G_1$. Choose a minimal (with respect to inclusion) path $H_1, \ldots, H_n$ outside F that connects $H_1 = G_1$ to $H_n = G_2$. Note that minimality implies that for $j > i+1$, $(H_i \cap H_j) \setminus F = \emptyset$. We claim that $\Delta' = \{F, H_1, \ldots, H_n\}$ is a cycle with

$$F \sim_{\Delta'} H_1 \sim_{\Delta'} \ldots \sim_{\Delta'} H_n \sim_{\Delta'} F. \tag{1}$$

(a) $F \sim_{\Delta'} G_1$ and $F \sim_{\Delta'} G_2$.

If $F \cap G_1 \subseteq H_i$ for some i, $1 < i < n$, then since $H_i \in \Delta_F^{G_1,G_2}$, we have $F \cap G_1 \subseteq H_i \cap F = G_1 \cap G_2 \subseteq G_2$. This implies that $G_1 \leqslant_F G_2$, a contradiction. So $F \sim_{\Delta'} G_1$, and similarly $F \sim_{\Delta'} G_2$

(b) $H_i \sim_{\Delta'} H_{i+1}$ for $i = 1, \ldots, n-1$.

Since $(H_i \cap H_{i+1}) \setminus F \neq \emptyset$, we have that $H_i \cap H_{i+1} \not\subseteq F$. By minimality of the path, if $H_i \cap H_{i+1} \subseteq H_j$ for some $j > i+1$, then $H_i \cap H_{i+1} \subseteq H_i \cap H_j \subseteq F$, a contradiction. The case $j < i$ is similar.

This shows (1). To finish the proof that Δ' is a cycle, we must show that it meets the remaining condition of Theorem 3.16. If $n = 2$, there is nothing to show; assume therefore that $n \geqslant 3$. By definition of $\Delta_F^{G_1,G_2}$, $F \cap H_j = G_1 \cap G_2$ for $j = 2, \ldots, n-1$, and so $\bigcap_{G \in \Delta'} G = G_1 \cap G_2$. Also, if $j > i+1$, then $H_i \cap H_j \subseteq F$ by minimality of the path, therefore

$$H_i \cap H_j = (H_i \cap F) \cap (H_j \cap F) = G_1 \cap G_2 = \textstyle\bigcap_{G \in \Delta'} G.$$

So Δ' is a cycle.

Conversely, suppose that Δ' is a cycle containing F, G_1 and G_2, written as $F \sim_{\Delta'} G_1 \sim_{\Delta'} H_1 \sim_{\Delta'} \ldots \sim_{\Delta'} H_n \sim_{\Delta'} G_2 \sim_{\Delta'} F$, where $n \geqslant 0$.

From the strong neighbor relations it follows that $G_1 \not\leqslant_F G_2$ and $G_2 \not\leqslant_F G_1$. It also follows that the above sequence of strong neighbors provides a path from G_1 to G_2 outside F. We only need to show that for $i = 1, \ldots, n$, $H_i \cap F = G_1 \cap G_2$.

If $\Delta' = \{F, G_1, G_2\}$ we are done. So assume that $n \geqslant 1$.

We know $H_i \not\sim_{\Delta'} F$, and so by Lemma 3.14, $H_i \cap F \subseteq G_1 \cap G_2$. On the other hand, since $G_1 \not\sim_{\Delta'} G_2$, Lemma 3.14 implies the opposite inclusion $H_i \cap F \supseteq G_1 \cap G_2$. It therefore follows that $H_i \cap F = G_1 \cap G_2$ and we are done. □

An immediate implication of Proposition 4.5 is an (algorithmically) efficient criterion to determine whether or not a facet complex is a tree.

Theorem 4.6 (Main Theorem). *Let Δ be a connected facet complex. Then Δ is a tree if and only if no triple of facets in Δ satisfies the triple condition.*

5 A polynomial-time tree decision algorithm

By Theorem 4.6, to check if a facet complex $\Delta = \{G_1, \ldots, G_l\}$ is a tree, we only need to check the triple condition for all triples of elements of Δ. The checks themselves are straightforward. Since the triple condition for $\langle F, G, G' \rangle$ is clearly unchanged if one switches G and G', we can limit triple checking to the elements of the set $\{\langle F, G_i, G_j \rangle \in \Delta^3 \mid G_i \neq F \neq G_j, i < j\}$. The procedures for the basic steps follow immediately from the earlier definitions.

Algorithm 5.1 (Tree decision algorithm).
Input: a connected facet complex $\Delta = \{G_1, \ldots, G_l\}$ with n vertices.
Output: **True** if Δ is a tree, **False** otherwise.

1. For each triple $\langle F, G, G' \rangle \in \{\langle F, G_i, G_j \rangle \in \Delta^3 \mid G_i \neq F \neq G_j, i < j\}$
   (a) If $G \leqslant_F G'$ or $G' \leqslant_F G$, continue with the next triple.
   (b) Build $\Delta_F^{G,G'}$.
   (c) If G and G' are connected outside F in $\Delta_F^{G,G'}$, return **False**.
2. Return **True**.

The correctness of this algorithm is an immediate consequence of Theorem 4.6. The algorithm uses very little memory; the input Δ requires nl bits, and $\Delta_F^{G,G'} \subseteq \Delta$ requires l bits. The memory required to perform the connectedness check and to store the various counters is negligible. Thus, memory locality is good, and the computations can generally take place in the cache.

Remark 5.2. In the process of checking the triple condition for a triple $\langle F, G, G' \rangle$ that is part of a cycle, we build a connection path outside F. Clearly, any such path can be reduced to a *minimal* connection path $\{H_1, \ldots, H_n\}$ outside F for G, G', and therefore, by the proof of Proposition 4.5, $\{F, H_1, \ldots, H_n\}$ forms a cycle. Therefore, an easy modification of Algorithm 5.1 allow us to produce the set of all the facets $F \in \Delta$ that are part of some cycle, and a cycle $\Delta'_F \supseteq \{F\}$ for each of them.

5.1 Complexity

For each triple it is trivial to see that steps (a) and (b) can be performed with cost $O(n)$ and $O(nl)$ respectively. For step (c), the following holds.

Lemma 5.3. *Let Δ be a facet complex with l facets over n variables such that F, G, G' are distinct facets of Δ. The connectedness outside F of $G, G' \in \Delta$ can be determined with time cost $O(nl)$.*

Proof. First of all we substitute Δ with the set $\{H \setminus F \mid H \in \Delta\}$. We then define $n+1$ equivalence relations $P_0, \ldots, P_n$ on the set $\{1, \ldots, l\}$. P_0 is the identity relation, i.e., each equivalence class is a singleton. For each $j = 1, \ldots, n$, consider the vertex v_j and the set $X_j = \{i \mid v_j \in F_i\}$. Let P_j be the smallest equivalence relation such that $P_{j-1} \subseteq P_j$ and such that for all $i, i' \in X_j$, $(i, i') \in P_j$. Then facets F_i and $F_{i'}$ are connected if and only if $(i, i') \in P_n$. With a suitable data structure for representing equivalence relations, the complexity of the procedure above is $O(nl)$. $\square$

Consequently, step (c) of the tree decision algorithm can be performed at cost $O(nl)$. Thus, the total complexity of the tree decision algorithm is as follows: in the worst case we have to check $3 \cdot \binom{l}{3} = \frac{l(l-1)(l-2)}{2} = O(l^3)$ triples. The complexity of the steps (a)–(c) is $O(nl)$ and hence the total complexity of the algorithm is $O(nl^4)$.

Example 5.4. Consider the facet complex $\Delta = \{xy, xz, yz, yu, zt\}$. We have to check $3 \cdot \binom{5}{3} = 30$ triples. We start with the triple $\langle xy, xz, yz \rangle$.

- $xz \not\leqslant_{xy} yz$ since $xy \cap xz = x \not\subseteq y = xy \cap yz$. Similarly $yz \not\leqslant_{xy} xz$.
- xz and yz are connected outside xy in the complex $\Delta_{xy}^{xz,yz} = \{zt, xz, yz\}$.

We have hence discovered that Δ is not a tree. A more unlucky choice of facets could have brought about the checking of 27 useless triples before the discovery that Δ is not a tree, the other two useful triples being $\langle yz, xy, xz \rangle$ and $\langle xz, xy, yz \rangle$.

Example 5.5. Some statistics for a bigger random example. Consider the facet complex $\Delta = \{lka$, qik, $tykj$, wuv, rjb, $eioab$, gdc, zv, rtj, $qrvm$, gzm, $tgzb$, $rgvm$, $qlav$, $qeocn$, $ikfaz$, bn, $ekjs$, $pfvn$, $wtodv\}$. We discover that it is not a tree after checking 4 facets; we performed the connectedness check only once. If one checks all $3 \cdot \binom{20}{3} = 3420$ triples, one finds that 445 of them require a connectedness check, and 403 of them reveal that Δ is not a tree.

Example 5.6. The facet complex $\{x_i x_{i+1} x_{i+2} \mid i = 1, \ldots, 400\}$ is trivially a tree. Checking this by a direct application of Algorithm 5.1 requires dealing with $3 \cdot \binom{400}{3} = 31,760,400$ triples, and takes about 12.6 seconds on an Athlon 2600+ machine for our C++ implementation. All the timings in the remainder of this paper refer to this machine.

5.2 Optimization

The runtime of Algorithm 5.1 can be improved by introducing some optimizations. First, note that if F is a facet such that no triple $\langle F, G, G' \rangle$ satisfies the triple condition, then by Proposition 4.5, F cannot be part of any cycle of Δ. Therefore, F can be removed from Δ, reducing the number of subsequent triple checks. We refer to this optimization as the *removal of useless facets*.

Example 5.7. We check the tree $\{x_i x_{i+1} x_{i+2} \mid i = 1, \ldots, 400\}$ of Example 5.6 with a version of Algorithm 5.1 with removal of useless facets. This requires checking 10,586,800 triples and takes about 3.46 seconds.

An important special case of a "useless facet" is a reducible leaf, as captured in the following definition:

Definition 5.8 (Reducible leaf). A facet F of a facet complex Δ is called a *reducible leaf* if for all $G, G' \in \Delta$, either $G \leqslant_F G'$ or $G' \leqslant_F G$.

A reducible leaf is called a "good leaf" by Zheng [Z].

Remark 5.9. The facet F is a reducible leaf of Δ if and only if F is a leaf of every $\Delta' \subseteq \Delta$ with $F \in \Delta'$.

The remark immediately implies that a reducible leaf cannot be part of a cycle. Thus, it can be removed from Δ, and the algorithm can then be recursively applied to $\Delta \setminus \{F\}$. We were not able to find a tree without a reducible leaf; in fact, Zheng [Z] conjectured that this is always the case. Checking whether a given facet F is a reducible leaf requires ordering all facets with respect to $\leqslant_F$, which takes $O(nl \log l)$ steps. A reducible leaf can thus be found in time $O(nl^2 \log l)$. Therefore, if Zheng's conjecture is true, the tree problem can be decided in time $O(nl^3 \log l)$. But even if the conjecture is not true, removing all reducible leaves at the beginning of Algorithm 5.1 is still a worthwhile optimization.

5.3 Optimization for sparse complexes

Let Δ be a facet complex with l facets. If every $F \in \Delta$ intersects a substantial ($\approx l$) number of facets, then the number of cycles is probably high and our algorithm is usually able to detect one of them easily. If this does not happen, we can exploit the "sparseness" of the facet complex in our algorithm.

For the remainder of this subsection, Δ will be a facet complex with l facets over n vertices such that the maximum number of neighbors of a facet $F \in \Delta$

is d and the maximum number of vertices of a facet $F \in \Delta$ is v. Note that trees are the hard cases for our algorithm, since all the triples have to be checked. Also note that, if Δ is a tree, then $l \leqslant n$. This follows by induction on l, from the fact that every leaf contains at least one free vertex.

5.3.1 Connection set algorithm

To check if Δ is a tree it is sufficient to check the connected triples only. For each facet F (l facets): first construct the set of all facets G connected to F (called the *connection set*, at cost $O(lv)$), then for all G, G' in the set ($O(d^2)$ pairs) perform the triple check on $\langle F, G, G' \rangle$ (cost $O(nl)$ per triple). We call this optimization of Algorithm 5.1 the *connection set algorithm.* The total cost is $O(nl^2d^2)$. The space required to construct the connection sets is $O(d)$, hence negligible. If the complex is not sparse ($d \approx l$, $v \approx n$), the complexity is the same as Algorithm 5.1. However, for sparse examples, this optimization is clearly worthwhile:

Example 5.10. We check the tree $\{x_i x_{i+1} x_{i+2} \mid i = 1, \ldots, 400\}$ of Example 5.6 with the algorithm detailed above. We deal with 398 triples and spend 0.2 seconds.

Example 5.11. The facet complex $\{x_i x_{i+1} \cdots x_{i+200} \mid i = 1, \ldots, 3200\}$ is a tree but not sparse. Tree checking with the connection set algorithm is still quite efficient; it requires dealing with $61,013,400$ triples, and takes about 140 seconds. Without any optimization, the number of triples to check is $16,368,643,200$ and the time spent by the algorithm is > 2 days.

5.3.2 Incidence matrix algorithm

The connectedness relation for a facet complex Δ can be represented by a graph through an incidence matrix. This matrix can be built and used during the tree checking algorithm. Since creating incidence matrices from a complex is a relatively expensive operation, we build them in steps, exploiting at every step the relations already computed.

We compute the connectedness relation for Δ at cost $O(l^2d)$. Then for every facet $F \in \Delta$ we compute the "connectedness outside F" relation for Δ, at cost $O(nld)$. Then for every triple $\langle F, G, G' \rangle$ (there are $O(d^2)$ of them) we compute the "connectedness outside F" relation for $\Delta_F^{G,G'}$ at cost $O(dv + ld)$. Using this additional structure, we do not actually need to build $\Delta_F^{G,G'}$, and we can check connectedness outside F in $\Delta_F^{G,G'}$ using the connectedness relations at cost $O(ld)$. We call this optimization of Algorithm 5.1 the *incidence matrix algorithm.*

The total complexity for this algorithm is hence $O(nl^2d + ld^3v + l^2d^3)$. If Δ is not sparse ($v \approx n$, $d \approx l$), then this algorithm has roughly the same complexity as Algorithm 5.1.

On the other hand, if $d \approx v \approx \sqrt{l} \approx \sqrt{n}$, which is a reasonable assumption for sparseness, then the complexity of the incidence matrix algorithm is

$O(l^3\sqrt{l})$, while the complexity of the connection set algorithm is $O(l^4)$ and that of Algorithm 5.1 is $O(l^5)$.

6 Algebraic properties of facet ideals

We now study facet ideals from a more algebraic point of view. In particular, we are interested in ways to determine whether a given facet complex Δ is Cohen-Macaulay, meaning whether $R/\mathcal{F}(\Delta)$ is a Cohen-Macaulay ring. We first need to introduce some new terminology.

Definition 6.1 (Vertex covering number, unmixed facet complex). Let Δ be a facet complex. A *vertex cover* for Δ is a set A of vertices of Δ, such that $A \cap F \neq \emptyset$ for every facet F. The smallest cardinality of a vertex cover of Δ is called the *vertex covering number* of Δ and is denoted by $\alpha(\Delta)$. A vertex cover A is *minimal* if no proper subset of A is a vertex cover. A facet complex Δ is *unmixed* if all of its minimal vertex covers have the same cardinality.

Example 6.2. Consider the two facet complexes in Example 2.8. We have $\alpha(\Delta) = 2$. Also, Δ is unmixed as its minimal vertex covers $\{x, u\}$, $\{y, u\}$, $\{y, v\}$, $\{z, u\}$ and $\{z, v\}$ all have cardinality equal to two. We further have $\alpha(\Delta') = 1$, but Δ' is not unmixed, because $\{x\}$ and $\{y, z\}$ are minimal vertex covers of different cardinalities.

The following observations are basic but useful.

Proposition 6.3 (Cohen-Macaulay facet complexes [F1, F3]). *Let Δ be a facet complex with vertices in $x_1, \ldots, x_n$, and consider its facet ideal $I = \mathcal{F}(\Delta)$ in the polynomial ring $R = k[x_1, \ldots, x_n]$. Then the following hold:*

(a) height $I = \alpha(\Delta)$ *and* dim $R/I = n - \alpha(\Delta)$.

(b) An ideal $p = (x_{i_1}, \ldots, x_{i_s})$ of R is a minimal prime of I if and only if the set $\{x_{i_1}, \ldots, x_{i_s}\}$ is a minimal vertex cover for Δ.

(c) If $k[x_1, \ldots, x_n]/\mathcal{F}(\Delta)$ is Cohen-Macaulay, then Δ is unmixed.

6.1 Grafting

One of the most basic ways to build a Cohen-Macaulay facet complex is via grafting.

Definition 6.4 (Grafting [F3]). A facet complex Δ is a *grafting* of the facet complex $\Delta' = \{G_1, \ldots, G_s\}$ with the facets $F_1, \ldots, F_r$ (or we say that Δ is *grafted*) if

$$\Delta = \{F_1, \ldots, F_r\} \cup \{G_1, \ldots, G_s\}$$

with the following properties:

(i) $G_1 \cup \ldots \cup G_s \subseteq F_1 \cup \ldots \cup F_r$;

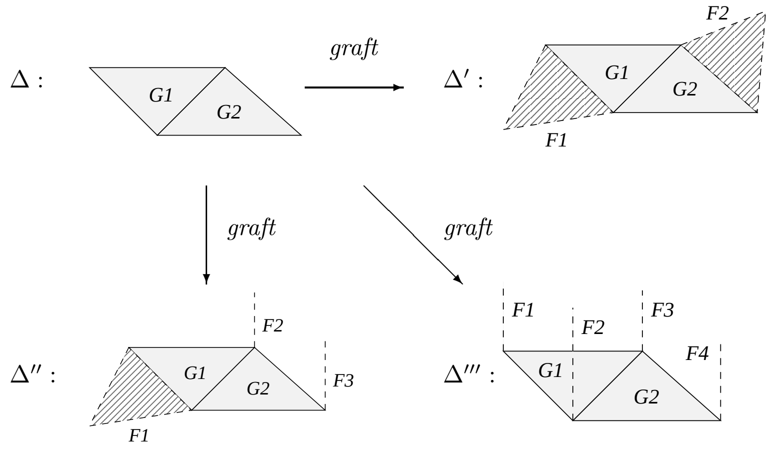


Figure 1: Three different ways of grafting the facet complex Δ.

(ii) $F_1, \ldots, F_r$ are all the leaves of Δ;

(iii) $\{G_1, \ldots, G_s\} \cap \{F_1, \ldots, F_r\} = \emptyset$;

(iv) For $i \neq j$, $F_i \cap F_j = \emptyset$;

(v) If G_i is a joint of Δ, then $\Delta \setminus \{G_i\}$ is also grafted.

Note that the definition is recursive, since graftedness of Δ is defined in terms of graftedness of $\Delta \setminus \{G_i\}$. Also note that a facet complex that consists of only one facet or several pairwise disjoint facets is grafted, as it can be considered as a grafting of the empty facet complex. It is easy to check that conditions (i) to (v) above are satisfied in this case. It is also clear that the union of two or more grafted facet complexes is itself grafted.

Example 6.5. There may be more than one way to graft a given facet complex. For example, some possible ways of grafting $\{G_1, G_2\}$ are shown in Figure 1.

The interest in grafted facet complexes, from an algebraic point of view, lies in the following facts.

Theorem 6.6 (Grafted facet complexes are Cohen-Macaulay [F3]). *Let Δ be a grafted facet complex. Then $\mathcal{F}(\Delta)$ is Cohen-Macaulay.*

Even more holds when Δ is a tree.

Theorem 6.7 ([F3, Corollaries 7.8, 8.3]). *If Δ is a simplicial tree, then the following are equivalent:*

(i) Δ is unmixed;

(ii) Δ is grafted;

(iii) $\mathcal{F}(\Delta)$ is Cohen-Macaulay.

6.2 Graftedness algorithm

A direct application of Definition 6.4 is not very convenient for checking whether a given facet complex Δ is grafted, since at each step of the recursion, one potentially needs to check condition (v) for several of the G_i, and this leads to a worst-case exponential algorithm. In order to arrive at a more efficient algorithm, we characterize graftedness as follows:

Lemma 6.8 (cf. [F3, Remarks 7.2, 7.3]). *A facet complex Δ is grafted if and only if (1) for each vertex v, there exists a unique leaf F such that $v \in F$, and (2) all leaves of Δ are reducible.*

Sketch of the proof. First, assume that Δ is grafted. Condition (1) follows from (i), (ii) and (iv). The fact that all leaves are reducible is shown by induction on the number of facets of Δ. The converse is also shown by induction. Suppose Δ satisfies (1) and (2), and let $\{F_1, \ldots, F_r\}$ and $\{G_1, \ldots, G_s\}$ be the sets of leaves and non-leaves, respectively. Conditions (i)–(iv) hold trivially. Further, if G_i is a joint, then $F_1, \ldots, F_r$ are still reducible leaves of $\Delta \setminus \{G_i\}$ by Remark 5.9. Also, there are no additional leaves in $\Delta \setminus \{G_i\}$, since none of the G_j have free vertices by Condition (1). Therefore, $\Delta \setminus \{G_i\}$ satisfies (1) and (2) and is therefore grafted by induction hypothesis, proving (v). □

The algorithm for checking if a facet complex is grafted follows immediately from Lemma 6.8.

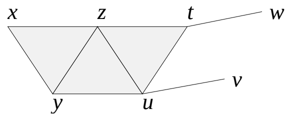
Algorithm 6.9 (Graftedness algorithm).
Input: A facet complex Δ with l facets and n vertices.
Output: **True** if Δ is grafted, **False** otherwise.

1. Build the lists $\mathcal{F} = \{F_1, \ldots, F_k\}$ (leaves of Δ) and $\mathcal{G} = \{G_1, \ldots, G_m\}$ (facets of Δ which are not leaves).
2. If $\bigcup_{G \in \mathcal{G}} G \not\subseteq \bigcup_{F \in \mathcal{F}} F$, return **False**.
3. If $\exists\ F, F' \in \mathcal{F}$ such that $F \cap F' \neq \emptyset$, return **False**.
4. If $\exists\ F \in \mathcal{F}$ that is not a reducible leaf, return **False**.
5. Return **True**.

6.3 Complexity

The leaf checking cost is $O(nl)$, hence the cost of step 1 is $O(nl^2)$. The cost of steps 2 and 3 is $O(nl)$. For step 4, there are k facets F to check. Checking whether F is reducible takes $O(nl \log l)$ steps as mentioned in Section 5.2. Therefore the total cost for step 4 is $O(nl^2 \log l)$, and this is the cost of the algorithm.

Example 6.10. Let $\Delta = \{xyz, yzu, ztu, uv, tw\}$, with $\mathcal{F} = \{xyz, uv, tw\}$ and $\mathcal{G} = \{yzu, ztu\}$. Then $\bigcup_{G\in\mathcal{G}} G \subseteq \bigcup_{F\in\mathcal{F}} F = \{x, y, z, t, u, v, w\}$ and $xyz \cap uv = xyz \cap tw = uv \cap tw = \emptyset$. Additionally, we check that each $F \in \mathcal{F}$ is a reducible leaf by showing that the set $\{F \cap G \mid G \in \mathcal{G}\}$ is a totally ordered set under inclusion. For example, if $F = xyz$, then this set is equal to $\{yz, z\}$ which is totally ordered. This holds for all $F \in \mathcal{F}$, and hence the facet complex is grafted.



References

[CFS1] M. Caboara, S. Faridi, P. Selinger. Tree computations. In: Proceedings of the MEGA05 Conference, Alghero, Italy, May 27th–June 1st 2005. Extended Abstract, 2005.

[CFS2] M. Caboara, S. Faridi, P. Selinger. Prototype implementation of tree algorithms, available at `http://www.dm.unipi.it/~caboara/Research/`, 2006.

[F1] S. Faridi. The facet ideal of a simplicial complex. Manuscripta Mathematica 109, 159–174, 2002.

[F2] S. Faridi. Simplicial trees are sequentially Cohen-Macaulay. J. Pure and Applied Algebra 190, 121–136, 2004.

[F3] S. Faridi. Cohen-Macaulay properties of square-free monomial ideals. Journal of Combinatorial Theory, Series A 109 (2), 299–329, 2005.

[F4] S. Faridi. Monomial ideals via square-free monomial ideals. In: Commutative Algebra: Geometric, Homological, Combinatorial and Computational Aspects. Lecture Notes in Pure and Applied Mathematics 244. pp. 85–114, 2005

[SVV] A. Simis, W. Vasconcelos, R. Villarreal. On the ideal theory of graphs. J. Algebra 167 (2), 389–416, 1994

[Vi1] R. Villarreal. Cohen-Macaulay graphs. Manuscripta Math. 66 (3), 277–293, 1990.

[Z] X. Zheng. Homological properties of monomial ideals associated to quasi-trees and lattices. Ph.D. thesis, Universität Duisburg-Essen, August 2004.