comput. complex. (2022) 31:5
© The Author(s) 2022
1016-3328/22/010001-33
published online May 20, 2022
https://doi.org/10.1007/s00037-022-00222-9

COMPUTING ZERO-DIMENSIONAL TROPICAL VARIETIES VIA PROJECTIONS

Paul Görlach, Yue Ren, and Leon Zhang

Abstract. We present an algorithm for computing zero-dimensional tropical varieties using projections. Our main tools are fast monomial transforms of triangular sets. Given a Gröbner basis, we prove that our algorithm requires only a polynomial number of arithmetic operations, and, for ideals in shape position, we show that its timings compare well against univariate factorization and backsubstitution. We conclude that the complexity of computing positive-dimensional tropical varieties via a traversal of the Gröbner complex is dominated by the complexity of the Gröbner walk.

Keywords. tropical geometry, tropical varieties, computer algebra

Subject classification. 14T10, 13P10, 13P15, 68W30

1. Introduction

Tropical varieties are piecewise linear structures which arise from polynomial equations. They appear naturally in many areas of mathematics and beyond, such as geometry (Mikhalkin 2005), combinatorics (Ardila & Klivans 2006; Feichtner & Sturmfels 2005), and optimisation (Allamigeon *et al.* 2018), as well as phylogenetics (Lin *et al.* 2018; Speyer & Sturmfels 2004), celestial mechanics (Hampton & Jensen 2011; Hampton & Moeckel 2006), and auction theory (Baldwin & Klemperer 2019; Tran & Yu 2019). Wherever they emerge, tropical varieties often provide a fresh insight into existing computational problems, which is why efficient algorithms and optimised implementations are of great importance.

Computing tropical varieties of polynomial ideals is a fundamentally important yet algorithmically challenging task, requiring sophisticated techniques from computational algebra and convex geometry. Currently, Gfan (Jensen 2017) and Singular (Decker *et al.* 2019) are the only two programs capable of computing general tropical varieties. Both programs rely on a traversal of the Gröbner complex as initially suggested by Bogart, Jensen, Speyer, Sturmfels, and Thomas (Bogart *et al.* 2007), and for both programs the initial bottleneck had been the computation of the so-called tropical links. Experiments suggest that this bottleneck was resolved with the recent development of new algorithms (Chan 2013; Hofmann & Ren 2018). However, the new approaches still rely on computations that are known to be very hard, Chan (2013) on elimination and Hofmann & Ren (2018) on root approximation to an unknown precision.

In this paper, we study the computation of zero-dimensional tropical varieties, which is the key computational ingredient in Hofmann & Ren (2018). The computation uses triangular decomposition, which was also used in Hofmann & Ren (2018), and skew projections, which is the key conceptual idea behind Chan (2013). The triangular decomposition splits the ideal into parts on which transformations can be efficiently applied. We show that the algorithm requires a polynomial amount of field operations if we start with a Gröbner basis. In particular, we argue that in the computation of general tropical varieties, the calculation of the so-called tropical links becomes computationally insignificant compared to the Gröbner walk required to traverse the tropical variety.

Note that projections are a well-studied approach in polynomial systems solving; see Sturmfels (2002), Dickenstein & Emiris (2005) for an overview on various techniques. Our approach can be regarded as a non-Archimedean analogue of that strategy, since tropical varieties can be regarded as zeroth-order approximation of the solutions in the topology induced by the valuation.

Our paper is organised as follows: In Section 3, we introduce a special class of monomial transformations and study how they act on triangular sets. In Section 4, we explain our main algorithm for reconstructing zero-dimensional tropical varieties of triangular sets from their projections, while Section 5 analyses the complexity of our algorithm. Section 6 touches upon some technical details of the implementation for the special case that the ideal is in shape position, and Section 7 compares the performance of our algorithm against a MAGMA implementation using univariate factorization and backsubstitution.

Implementations of all our algorithms can be found in the SINGULAR library `tropicalProjection.lib`. Together with the data for the timings, it is available at https://software.mis.mpg.de, and will also be made publicly available as part of the official SINGULAR distribution.

2. Background

For the sake of notation, we briefly recall some basic notions of tropical algebraic geometry and computational algebra that are of immediate relevance to us. In tropical geometry, our notation closely follows that of Maclagan & Sturmfels (2015).

CONVENTION 2.1. *For the remainder of the article, let K be a field with non-trivial valuation $\nu\colon K^* \to \mathbb{R}$ and fix a multivariate polynomial ring $K[\mathbf{x}] := K[x_1, \dots, x_n]$ as well as a multivariate Laurent polynomial ring $K[\mathbf{x}^\pm] := K[x_1^\pm, \dots, x_n^\pm]$.*

Moreover, given a Laurent polynomial ideal $I \subseteq K[\mathbf{x}^\pm]$, we call a finite subset $G \subseteq I$ a ***Gröbner basis*** *with respect to a monomial ordering $\prec$ on $K[\mathbf{x}]$ if G consists of polynomials and forms a Gröbner basis of the polynomial ideal $I \cap K[\mathbf{x}]$ with respect to $\prec$ in the conventional sense, see for example Greuel & Pfister (2002, §1.6). Finally, a* ***lexicographical Gröbner basis*** *will be a Gröbner basis with respect to the lexicographical ordering $\prec_{\text{lex}}$ with $x_n \prec_{\text{lex}} \cdots \prec_{\text{lex}} x_1$.*

While there are many equivalent definitions for tropical varieties, the following definition in terms of coordinate-wise valuations suffices for the purposes of this article. Also note that while tropical varieties can be defined with multiplicities, we are only interested in them set-theoretically.

Definition 2.2. *Let $I \subseteq K[\mathbf{x}^{\pm}]$ be a Laurent polynomial ideal. The **tropical variety** $\operatorname{Trop}(I) \subseteq \mathbb{R}^n$ is given by*

$$\begin{aligned}&\operatorname{Trop}(I)\\ &:= cl\Big(\big\{(\widehat{\nu}^{al}(p_1), \dots, \widehat{\nu}^{al}(p_n)) \in \mathbb{R}^n \mid (p_1, \dots, p_n) \in V_{\widehat{K}^{al*}}(I)\big\}\Big),\end{aligned}$$

where $\widehat{K}^{al}$ denotes the algebraic closure of the completion of K, so that ν extends uniquely to a valuation $\widehat{\nu}^{al}$ on $\widehat{K}^{al}$, $V_{\widehat{K}^{al*}}(I)$ denotes the very affine variety of I in the algebraic torus $(\widehat{K}^{al*})^n$, and $cl(\cdot)$ is the closure in the Euclidean topology.

In the univariate case, the tropical variety of an ideal $I = \langle f \rangle \subseteq K[x_1^{\pm}]$ simply consists of the negated slopes of the Newton polygon of f (Neukirch 1999, Proposition II.6.3). Our approach for computing zero-dimensional tropical varieties of multivariate ideals is based on computing sufficiently many projections to the univariate case.

For it, we will require the notion of triangular decomposition, a common concept for decomposing ideal into easier parts. If given a Gröbner basis, its complexity is polynomial time in the number of variables and the degree of the ideal (Lazard 1992). In practice, it is fast enough to be a standard tool in some polynomial solvers such as in Maple (Bernardin *et al.* 1996-2020).

Definition 2.3. *A **triangular set** $T \subseteq K[\mathbf{x}]$ is a finite set of polynomials, say $T = \{g_n, \dots, g_1\}$, where each $g_i \in K[x_i, \dots, x_n]$ is of the form $g_i = x_i^{d_i} - f_i$ for some $d_i \in \mathbb{N}_{>0}$ and $f_i \in K[x_i, \dots, x_n]$ with x_j-degree less than d_j for $j \geq i$. Note that this makes any triangular set a reduced lexicographical Gröbner basis.*

Proposition 2.4 (Greuel & Pfister 2002, Corollary 4.7.4). *Let $J \subseteq K[\mathbf{x}]$ be a zero-dimensional polynomial ideal. Then, there are triangular sets $T_1, \dots, T_k \subseteq K[\mathbf{x}]$ such that*

$$\sqrt{J} = \bigcap_{T \in \mathcal{T}} \sqrt{\langle T \rangle} \quad \text{and} \quad \langle T_i \rangle + \langle T_j \rangle = K[\mathbf{x}] \text{ for } i \neq j.$$

Since each zero-dimensional ideal can be efficiently decomposed into triangular sets, we will focus on ideals that are generated by triangular sets from now onwards. Moreover, we will put special emphasis on ideals in shape position.

Definition 2.5. *A zero-dimensional ideal $I \subseteq K[\mathbf{x}^{\pm}]$ is in **shape position** if it is generated by a triangular set $T = \{g_n, \dots, g_1\}$ with $d_i = 1$ for $i < n$, i.e. $g_i = x_i - f_i$ for a univariate polynomial $f_i \in K[x_n]$ for $i < n$.*

3. Unitriangular transformations on triangular sets

In this section, we consider special transformations on $K[\mathbf{x}^{\pm}]$ which arise from unitriangular transformations on the lattice of Laurent monomials and describe how they operate on triangular sets.

Definition 3.1. *For any $u = (u_2, \dots, u_n) \in \mathbb{Z}_{\geq 0}^{n-1}$, we define a ring automorphism*

$$\varphi_u \colon K[\mathbf{x}^{\pm}] \to K[\mathbf{x}^{\pm}], \qquad x_i \mapsto \begin{cases} x_1 \cdot x_2^{-u_2} \cdots x_n^{-u_n} & \textit{if} \quad i = 1, \\ x_i & \textit{if} \quad i \neq 1, \end{cases}$$

and a linear projection

$$\pi_u \colon \mathbb{R}^n \twoheadrightarrow \mathbb{R}, \qquad (w_1, \dots, w_n) \mapsto w_1 + \sum_{i=2}^{n} u_i w_i.$$

We call such a φ_u a slim (unitriangular) transformation.

The reason we restrict ourselves to these simple transformations is because they allow us to compute a wide range of projections while being easy to use.

Lemma 3.2. *Let φ_u be a slim transformation. Then,*

$$\pi_u(\mathrm{Trop}(I)) = \mathrm{Trop}(\varphi_u(I) \cap K[x_\ell^{\pm}]).$$

Proof. We may assume that K is algebraically closed. Note that the ring automorphism φ_u induces a torus automorphism $f_u\colon (K^*)^n \xrightarrow{\sim} (K^*)^n$ with $f_u^{-1}(V(I)) = V(\varphi_u(I))$, which in turn induces a linear transformation $h_u\colon \mathbb{R}^n \xrightarrow{\sim} \mathbb{R}^n$ mapping $\operatorname{Trop}(\varphi_u(I))$ to $\operatorname{Trop}(I)$:

$$\begin{tikzcd}
x_1 \prod_{i>1} x_i^{-u_i} & & x_1 \arrow[ll, mapsto] \\
K[\mathbf{x}^{\pm}] & & K[\mathbf{x}^{\pm}] \arrow[ll, "\varphi_u"'] \\
& \text{induces} & \\
(z_1,\dots,z_n) \arrow[rr, mapsto] & & (z_1 \cdot \prod_{i>1} z_i^{-u_i}, z_2,\dots,z_n) \\
(K^*)^n \arrow[rr, "f_u"'] \arrow[d, "\nu"'] & & (K^*)^n \arrow[d, "\nu"] \\
\mathbb{R}^n \arrow[rr, "h_u"] & & \mathbb{R}^n \\
(w_1,\dots,w_n) \arrow[rr, mapsto] & & (w_1 - \sum_{i>1} u_i w_i, w_2,\dots,w_n)
\end{tikzcd}$$

Hence, with $p_1\colon \mathbb{R}^n \twoheadrightarrow \mathbb{R}$ denoting the projection onto the first coordinate:

$$\begin{aligned}\operatorname{Trop}(\varphi_u(I) \cap K[x_1^{\pm}]) &= p_1(\operatorname{Trop}(\varphi_u(I)))\\ &= (p_1 \circ h_u^{-1})(\operatorname{Trop}(I)) = \pi_u(\operatorname{Trop}(I)). \quad \square\end{aligned}$$

Algorithm 3.3. *(Unitriangular transformations of triangular sets)*

Input: (T, u), where

- ○ T is a triangular generating set of a zero-dimensional ideal $I \subseteq K[\mathbf{x}^{\pm}]$,
- ○ φ_u is a slim transformation.

Output: T', a triangular set generating $\varphi_u(I)$.

```
1: Suppose $g_1 := x_1^{d_1} - \sum_{i=1}^{d_1} p_i x_1^{d_1-i} \in T$ with $p_i \in K[x_2,\dots,x_n]$.
2: for $i = 1,\dots,d_1$ do
3:     $\hat{p}_i := \operatorname{reduce}\big((x_2^{u_2}\cdots x_n^{u_n})^i p_i, T \setminus \{g_1\}\big)$.
4: return $T' := T \setminus \{g_1\} \cup \{x_1^{d_1} - \sum_{i=1}^{d_1} \hat{p}_i x_1^{d_1-i}\}$.
```

Correctness of Algorithm 3.3. Only the last element in the triangular set

$$T = \Big\{ x_n^{d_n} - f_n, \dots, x_2^{d_2} - f_2, \underbrace{x_1^{d_1} - \sum_{i=1}^{d_1} p_i x_1^{d_1 - i}}_{=g_1} \Big\}$$

depends on x_1. Therefore, by replacing it with

$$\varphi_u(g_1) = (q^{-1}x_1)^{d_1} - \sum_{i=1}^{d_1} p_i (q^{-1}x_1)^{d_1-i},$$

where $q := x_2^{u_2} \dots x_n^{u_n} \in K[\mathbf{x}]$, we get generators of the transformed ideal $\varphi_u(I) \subseteq K[\mathbf{x}^{\pm}]$. Note that multiplying this element with the monomial q^{d_1} (which is invertible in $K[\mathbf{x}^{\pm}]$), we pass to the polynomial generating set of $\varphi_u(I)$ given by

$$T \setminus \{g_1\} \cup \{x_1^{d_1} - \sum_{i=1}^{d_1} p_i q^i x_1^{d_1-i}\}.$$

This is a non-reduced Grøbner basis with respect to the lexicographical ordering $\prec_{\text{lex}}$, and replacing $p_i q^i$ by $\hat{p}_i$, we reduce it modulo those generators not depending on x_1 to obtain the triangular set T'. $\square$

One special case that we would like to highlight separately is when an ideal I is in *shape position*. For ideals in shape position, Algorithm 3.3 simplifies drastically and performs quite well in practice, see the timings in Section 7. However, the complexity in Section 5 remains unchanged.

Remark 3.4. *(Unitriangular transformations of ideals in shape position) If I is in shape position, it is generated by a triangular set T of the following form for $f_n, \dots, f_1 \in K[x_n]$:*

$$T = \{x_n^d - f_n,\ x_{n-1} - f_{n-1},\ \dots,\ x_2 - f_2,\ x_1 - f_1\}.$$

This has two main implications:

(i) *It simplifies Algorithm 3.3. The triangular set generating $\varphi_u(I)$ is of the form $T' = \{x_n^d - f_n, \ldots, x_2 - f_2, x_1 - f_1'\}$, where $f_1' \in K[x_n]$ is the univariate polynomial with $\deg(f_1') < d$ and*

$$\begin{aligned} f_1' &\equiv \Big(x_n^{-u_n} \cdot \prod_{i=2}^{n-1} f_i^{-u_i}\Big)^{-1} \cdot f_1 \\ &\equiv \Big(x_n^{u_n} \cdot \prod_{i=2}^{n-1} f_i^{u_i}\Big) \cdot f_1 \pmod{x_n^d - f_n}. \end{aligned} \tag{3.5}$$

In particular, $\varphi_u(I)$ will be in shape position.

(ii) *It allows us to use a wider range of transformations beyond those considered in Definition 3.1. To be precise, replacing f_1 with f_ℓ in Equation (3.5) we may use any transformation of the form*

$$\varphi_u\colon K[\mathbf{x}^\pm] \to K[\mathbf{x}^\pm], \quad x_i \mapsto \begin{cases} x_1^{-u_1} \cdots x_\ell^{1} \cdots x_n^{-u_n} & \textit{if} \quad i = \ell, \\ x_i & \textit{if} \quad i \neq \ell, \end{cases}$$

with $u_i \in \mathbb{N}$.

4. Computing zero-dimensional tropical varieties via projections

In this section, we assemble our algorithm for computing $\mathrm{Trop}(I)$ for a zero-dimensional ideal $I \subseteq K[\mathbf{x}^\pm]$ generated by a triangular set. This is done in two stages, see Figure 4.1: In the first stage, we project $\mathrm{Trop}(I)$ onto all coordinate axes of $\mathbb{R}^n$. In the second stage, we iteratively glue the coordinate projections together by projecting $\mathrm{Trop}(I)$ onto more lines.

For the sake of simplicity, all algorithms contain some elements of ambiguity to minimise the level of technical detail. To see how these ambiguities are resolved in the actual implementation, see Section 6.

The following algorithm merges several small projections into a single large projection. For clarity, when given a finite subset $A \subseteq \{1, \ldots, n\}$, we use $\mathbb{R}^A$ to denote the linear subspace of $\mathbb{R}^n$

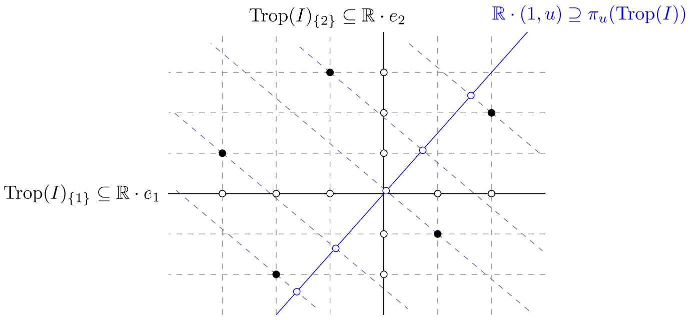

$\mathrm{Trop}(I)_{\{2\}} \subseteq \mathbb{R}\cdot e_2$
$\mathbb{R}\cdot(1,u) \supseteq \pi_u(\mathrm{Trop}(I))$
$\mathrm{Trop}(I)_{\{1\}} \subseteq \mathbb{R}\cdot e_1$

Figure 4.1: Zero-dimensional tropical varieties via projections.

spanned by the unit vectors indexed by A and p_A to denote the projection $\mathbb{R}^n \twoheadrightarrow \mathbb{R}^A$. For $w \in \mathbb{R}^n$ and $I \subseteq K[\mathbf{x}^{\pm}]$, we denote $w_A := p_A(w) \in \mathbb{R}^A$ and $\mathrm{Trop}(I)_A := p_A(\mathrm{Trop}(I)) \subseteq \mathbb{R}^A$.

Algorithm 4.1 (gluing projections).
Input: $(T, \mathrm{Trop}(I)_{A_1}, \dots, \mathrm{Trop}(I)_{A_k})$, where

- T is a triangular generating set of a zero-dimensional ideal $I \subseteq K[\mathbf{x}^{\pm}]$,
- $A_1, \dots, A_k \subseteq \{1, \dots, n\}$ are non-empty sets with $1 \in A := A_1 \cup \ldots \cup A_k$.

Output: $\mathrm{Trop}(I)_A \subseteq \mathbb{R}^A$.

```
1: Construct the candidate set
   $$C := \Big\{ w \in \mathbb{R}^A \;\Big|\; w_{A_i} \in \mathrm{Trop}(I)_{A_i} \text{ for } i = 1, \dots, k \Big\}.$$
2: Pick a $u \in \mathbb{Z}^{n-1}_{\geq 0}$ with $u_i = 0$ for $i \notin A$ such that the following map is injective:
   $$\pi_u|_C \colon C \to \mathbb{R}, \qquad (w_i)_{i\in A} \mapsto w_1 + \sum_{i \in A\setminus\{1\}} u_i w_i.$$
3: Using Algorithm 3.3, transform $T$ into a triangular set $T'$ generating $\varphi_u(I)$.
4: Compute the eliminant $\mu \in K[x_1]$, i.e. a generator of $\langle T' \rangle \cap K[x_1]$ and read off $\mathrm{Trop}(\mu) \subseteq \mathbb{R}$ from its Newton polygon.
```

```
5: return  $\{w \in C \mid \pi_u(w) \in \mathrm{Trop}(\mu)\}$.
```

Correctness of Algorithm 4.1. First, we show the existence of a slim transformation φ_u required for Line 2 such that π_u is injective on the candidate set C. Extending the definition of the linear projection π_v from $v \in \mathbb{N}^{n-1}$ in Definition 3.1 to arbitrary $v \in \mathbb{R}^{n-1}$, it suffices to show that the set

$$Z := \{v \in \mathbb{R}^{n-1}_{\geq 0} \mid \pi_v|_T \text{ is injective}\} \subseteq \mathbb{R}^{n-1}$$

contains an integer point. By the definition of π_v, we see that

$$Z = \mathbb{R}^{n-1}_{\geq 0} \setminus \bigcup_{w \neq w' \in T} H_{w-w'},$$
$$\text{where } H_{w-w'} := \left\{ v \in \mathbb{R}^{n-1} \;\middle|\; \sum_{i=2}^{n} (w_i - w_i')v_i = w_1' - w_1 \right\}.$$

This describes Z as the complement of an affine hyperplane arrangement in $\mathbb{R}^B$ inside the positive orthant. Therefore, Z must contain an integer point.

Next, note that the candidate set C contains $\mathrm{Trop}(I)_A$ by construction, so the injectivity of $\pi_u|_T$ implies

$$\mathrm{Trop}(I)_A = \{w \in T \mid \pi_u(w) \in \pi_u(\mathrm{Trop}(I))\}.$$

Hence, the correctness of the output follows from the equality $\pi_u(\mathrm{Trop}(I)) = \mathrm{Trop}(\mu)$, which holds by Lemma 3.2. □

The next algorithm computes $\mathrm{Trop}(I)$ by projecting it onto all coordinate axes and gluing the projections together via Algorithm 4.1.

Algorithm 4.2 (tropical variety via projections).

Input: T, a triangular generating set of a zero-dimensional ideal $I \subseteq K[\mathbf{x}^{\pm}]$.

Output: $\mathrm{Trop}(I) \subseteq \mathbb{R}^n$

```
1: for $k \in \{1, \dots, n\}$ do
2:     Compute the eliminant $\mu_k \in K[x_k]$, i.e. a generator of
       $\langle T \rangle \cap K[x_k]$, and read off the projection $\mathrm{Trop}(I)_{\{k\}} = \mathrm{Trop}(\mu_k)$.
```

```
3: Initialise a set of computed projections
       $$W := \{\mathrm{Trop}(I)_{\{1\}}, \dots, \mathrm{Trop}(I)_{\{n\}}\}.$$
4: while $W \not\ni \mathrm{Trop}(I)_{\{1,\dots,n\}}$ do
5:     Pick projections $\mathrm{Trop}(I)_{A_1}, \dots, \mathrm{Trop}(I)_{A_k} \in W$ to be merged
       together such that $1 \in A$ and $\mathrm{Trop}(I)_A \notin W$ for $A :=
       A_1 \cup \dots \cup A_k$.
6:     Using Algorithm 4.1, compute $\mathrm{Trop}(I)_A$.
7:     $W := W \cup \{\mathrm{Trop}(I)_A\}$.
8: return $\mathrm{Trop}(I)_{\{1,\dots,n\}}$.
```

Correctness and Termination of Algorithm 4.2. In every iteration of the `while` loop, the set W grows in size. As there are only finitely many sets $A \subseteq \{1, \dots, n\}$, we will compute $\mathrm{Trop}(I) = \mathrm{Trop}(I)_{\{1,\dots,n\}}$ after finitely many iterations. □

Example 4.3. Consider $K = \mathbb{Q}$ equipped with the 2-adic valuation and the ideal

$$I = \langle \underbrace{2x_3^4 + x_3^3 + x_3^2 + x_3 + 2}_{=:g_3}, x_2 - \underbrace{2x_3}_{=:f_2}, x_1 - \underbrace{4x_3}_{=:f_1} \rangle \subseteq K[x_1^\pm, x_2^\pm, x_3^\pm].$$

This ideal is in shape position by Lemma 2.5. From the Newton polygon of g_3, see Figure 4.2 (left), it is not hard to see that (for the sake of clarity, points with multiplicity 2 are highlighted in bold):

$$\begin{aligned}
\mathrm{Trop}(I)_{\{3\}} &= \mathrm{Trop}(g_3) = \{-1, \mathbf{0}, 1\},\\
\mathrm{Trop}(I)_{\{2\}} &= \{\lambda + 1 \mid \lambda \in \mathrm{Trop}(I)_{\{3\}}\} = \{0, \mathbf{1}, 2\},\\
\mathrm{Trop}(I)_{\{1\}} &= \{\lambda + 2 \mid \lambda \in \mathrm{Trop}(I)_{\{3\}}\} = \{1, \mathbf{2}, 3\}.
\end{aligned}$$

To merge $\mathrm{Trop}(I)_{\{1\}}$ and $\mathrm{Trop}(I)_{\{2\}}$, we consider the following projection that is injective on the candidate set $C := \mathrm{Trop}(I)_{\{1\}} \times \mathrm{Trop}(I)_{\{2\}}$:

$$\pi_{(3,0)} : \quad C \longrightarrow \mathbb{R}, \quad (w_1, w_2) \longmapsto w_1 + 3w_2.$$

The corresponding slim transformation $\varphi_{(3,0)}$ sends x_1 to $x_1x_2^{-3}$ and hence $\varphi_{(3,0)}(I)$ is generated by $\{g_3, x_2 - f_2, x_1x_2^{-3} - 4x_3\}$, which Algorithm 3.3 transforms into the following lexicographical Gröbner basis:

$$\varphi_{(3,0)}(I) = \Big\langle g_3,\ x_2 - f_2,\ x_1 - (\underbrace{-16x_3^3 - 16x_3^2 - 16x_3 - 32}_{=:f_1'})\Big\rangle.$$

The eliminant in $K[x_1]$ of $\varphi_{(3,0)}(I)$ can be computed as the resultant

$$\begin{aligned}&\mathrm{Res}_{x_3}(g_3, x_1 - f_1')\\ &\quad = 8x_1^4 + 752x_1^3 + 32256x_1^2 + 770048x_1 + 8388608.\end{aligned}$$

Figure 4.2 (middle) shows the Newton polygon of the resultant, from which we see:

$$\mathrm{Trop}(\mathrm{Res}_{x_3}(g_3, x_1 - f_1')) = \{9, \mathbf{5}, 1\}.$$

Thus,

$$\mathrm{Trop}(I)_{\{1,2\}} = \{(3,2), (\mathbf{2},\mathbf{1}), (1,0)\}.$$

To merge $\mathrm{Trop}(I)_{\{1,2\}}$ and $\mathrm{Trop}(I)_{\{3\}}$, we consider the following projection that is injective on the candidate set $C := \mathrm{Trop}(I)_{\{1,2\}} \times \mathrm{Trop}(I)_{\{3\}}$:

$$\pi_{(0,3)} : \quad C \longrightarrow \mathbb{R}, \quad (w_1, w_2, w_3) \longmapsto w_1 + 3w_3.$$

The corresponding slim transformation $\varphi_{(0,3)}$ sends x_1 to $x_1x_3^{-3}$ and hence $\varphi_{(0,3)}(I)$ is generated by $\{g_3, x_2 - f_2, x_1x_3^{-3} - 4x_3\}$, which Algorithm 3.3 transforms into the following lexicographical Gröbner basis:

$$\varphi_{(0,3)}(I) = \Big\langle g_3,\ x_2 - f_2,\ x_1 - (\underbrace{-2x_3^3 - 2x_3^2 - 2x_3 - 4}_{=:f_1''})\Big\rangle.$$

Another resultant computation yields the eliminant in $K[x_1]$ of $\varphi_{(0,3)}(I)$:

$$\mathrm{Res}_{x_3}(g_3, x_1 - f_1'') = 8x_1^4 + 94x_1^3 + 504x_1^2 + 1504x_1 + 2048.$$

Figure 4.2 (right) shows the Newton polygon of the resultant, from which we see:

$$\mathrm{Trop}(\mathrm{Res}_{x_3}(g_3, x_1 - f_1'')) = \{6, \mathbf{2}, -2\},$$

and thus

$$\mathrm{Trop}(I) = \mathrm{Trop}(I)_{\{1,2,3\}} = \{(3,2,1), (\mathbf{2},\mathbf{1},\mathbf{0}), (1,0,-1)\}.$$

$\diamondsuit$

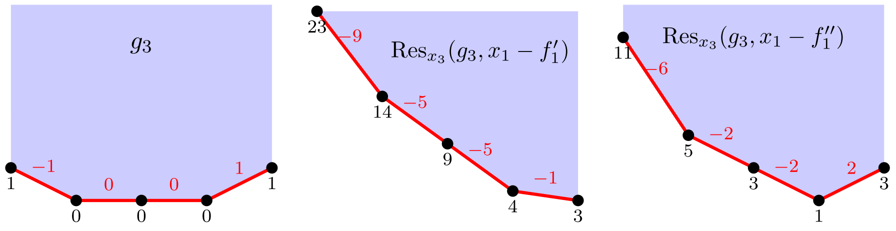

g_3
1
0
0
0
1
-1
0
0
1
$\mathrm{Res}_{x_3}(g_3, x_1 - f_1')$
23
14
9
4
3
-9
-5
-5
-1
$\mathrm{Res}_{x_3}(g_3, x_1 - f_1'')$
11
5
3
1
3
-6
-2
-2
2

Figure 4.2: Newton polygons of g_3 and the resultants in Example 4.3. Below each vertex is its height, above each edge is its slope.

Remark 4.4 (Eliminants for ideals in shape position). *If an ideal I is in shape position, say generated by $\{x_n^d - f_n, \ldots, x_1 - f_1\}$ with $f_i \in K[x_n]$, then computing eliminants such as in Line 4 of Algorithm 4.1 and Line 2 of Algorithm 4.2 becomes much simpler. To compute the eliminant of I in $K[x_\ell]$, it suffices to consider the two polynomials $x_n^d - f_n, x_\ell - f_\ell \in K[x_\ell, x_n]$ instead of the entire generating set.*

5. Complexity

In this section, we bound the complexity for computing a zero-dimensional tropical variety $\mathrm{Trop}(I) \subseteq \mathbb{R}^n$ of an ideal generated by a given triangular generating set using Algorithm 4.2 with the `sequential` strategy. The `sequential` strategy sequentially computes the projections $\mathrm{Trop}(I)_{\{1\}}$, $\mathrm{Trop}(I)_{\{1,2\}}$, ... until arriving at $\mathrm{Trop}_{\{1,\ldots,n\}} = \mathrm{Trop}(I)$. For more details on strategies, see Section 6.4. We show that the number of required arithmetic operations is polynomial in the degree of the ideal and the number of variables.

Combined with the `FGLM` algorithm (Faugère *et al.* 1993) and Lazard's lextriangular decomposition (Lazard 1992), this shows that the tropical variety of any zero-dimensional ideal can be computed from its reduced Gröbner basis using polynomially many arithmetic operations, see Corollary 5.9.

Convention 5.1. *For the remainder of the section, we assume that $\nu(K^*) \subseteq \mathbb{Q}$, so that $\mathrm{Trop}(I) \subseteq \mathbb{Q}^n$.*

For the sake of convenience, we recall some well-known results on the complexity of arithmetic operations over integral extensions.

Proposition 5.2. *(von zur Gathen & Gerhard (2013, Corollary 4.6 + Section 4.3)) Let R be a ring and let $f \in R[z]$ be a monic univariate polynomial of degree d. Then:*

(i) Addition and multiplication in $R[z]/\langle f\rangle$ require at most $\mathcal{O}(d^2)$ arithmetic operations in R.

(ii) Computing the qth power in $R[z]/\langle f\rangle$ requires at most $\mathcal{O}(d^2 \log q)$ arithmetic operations in R.

Corollary 5.3. *Given a triangular set $T \subseteq K[x_1,\ldots,x_n]$ generating a zero-dimensional ideal I of degree d:*

(i) Addition and multiplication in $K[x_1,\ldots,x_n]/I$ require at most $\mathcal{O}(d^2)$ arithmetic operations in K.

(ii) Computing the qth power in $K[x_1,\ldots,x_n]/I$ requires at most $\mathcal{O}(d^2 \log q)$ arithmetic operations in K.

Proof. Suppose $T = \{g_n,\ldots,g_1\}$ with $g_i \in K[x_i,\ldots,x_n]$. Let $T_k := T \cap K[x_k,\ldots,x_n]$, $R_k := K[x_k,\ldots,x_n]/\langle T_k\rangle$ for $k = n,\ldots,1$, and $T_{n+1} := \emptyset$, $R_{n+1} := K$. Note that T_k is a triangular set in R_k, and that R_k is an integral extension of R_{k+1} given by

$$R_k = R_{k+1}[x_k]/\langle g_k\rangle.$$

For $k \leq n$, Remark 5.2 implies that addition and multiplication in R_k can be carried out using at most $\mathcal{O}(d_k^2)$ arithmetic operations in R_{k+1} or $\mathcal{O}(d_k^2 \cdots d_n^2)$ arithmetic operations in K, and that computing a qth power in R_{k+1} requires at most $\mathcal{O}(d_k^2 \log q)$ arithmetic operations in R_{k+1} or $\mathcal{O}(d_k^2 \cdots d_n^2 \log q)$ arithmetic operations in K. The claimed bounds follow from $d = d_1 \cdots d_n$. □

Proposition 5.4. *Given a triangular set $T \subseteq K[x_1, \ldots, x_n]$ generating a zero-dimensional ideal I of degree d and a slim transformation φ_u, Algorithm 3.3 computes the triangular generating set of $\varphi_u(I)$ using at most $\mathcal{O}\big(d^2 \sum_{u_i>0} \log(u_i)\big)$ arithmetic operations in K.*

Proof. Consider the reductions in Algorithm 3.3 Line 3. Reducing $(x_2^{u_2} \ldots x_n^{u_n})^i p_i$ by $T \setminus \{g_1\}$ is equivalent to expressing it in $K[x_2, \ldots, x_n]/\langle T \setminus \{g_1\}\rangle$ as a linear combination of the K-basis $B := \{x_2^{b_2} \cdots x_n^{b_n} \mid 0 \leq b_i < d_i\}$.

By Corollary 5.3 and since $i \leq d_1$, expressing $(x_2^{u_2} \ldots x_n^{u_n})^i$ in terms of B requires at most

$$\mathcal{O}\Big(\big(\sum_{u_i>0} d_i^2 \cdots d_n^2 \log(u_i)\big) + (|\{i \mid u_i > 0\}| - 1) \cdot d_2^2 \cdots d_n^2 \\ + d_2^2 \cdots d_n^2 \log(d_1)\Big) \leq \mathcal{O}\Big(\log(d_1) d_2^2 \cdots d_n^2 \sum_{u_i>0} \log(u_i)\Big)$$

operations in K. As $d^2 = d_1^2 d_2^2 \cdots d_n^2$, repeating the computation for $i = 1, \ldots, d_1$ requires at most $\mathcal{O}\big(d^2 \sum_{u_i>0} \log(u_i)\big)$ operations in K. The multiplications by p_i for $i = 1, \ldots, d_1$ do not change the complexity. □

Lemma 5.5. *Given a triangular generating set $T \subseteq K[x_1, \ldots, x_n]$ of a zero-dimensional ideal I of degree d, the computation of the eliminant $\mu \in K[x_k]$ of I, i.e. a generator of $I \cap K[x_k]$, requires at most $\mathcal{O}(d^3)$ arithmetic operations in K.*

Proof. Note that the eliminant $\mu \in K[x_k]$ is also the minimal polynomial of $x_k \in K[x_1, \ldots, x_n]/I$. Hence, it can be computed by finding a linear relation among the powers $1, x_i, x_i^2, \dots, x_i^{d-1}$. By Corollary 5.3, computing all powers requires at most $\mathcal{O}(d^3)$ arithmetic operations in K and, by Bürgisser *et al.* (1997, Chapter 16), computing a linear dependency requires $\mathcal{O}(d^{\omega+\varepsilon})$, where $\omega < 3$ is the exponent of the complexity of matrix multiplication and $\varepsilon > 0$. $\square$

LEMMA 5.6. *Let $X, Y \subseteq \mathbb{Q}$ be finite sets of cardinality $\leq d$. Then, there exists a non-negative integer $m \leq \binom{d^2}{2}$ such that $X \times Y \to \mathbb{Q}$, $(a, b) \mapsto a - mb$ is injective. The smallest such m can be found in $\mathcal{O}(d^4)$ arithmetic operations in $\mathbb{Q}$.*

PROOF. The map $(a, b) \mapsto a - mb$ will fail to be injective if and only if there exists a pair of points in $X \times Y$ lying on an affine line with slope m. Since there are at most $\binom{d^2}{2}$ pairs of points, the statement follows by the pigeonhole principle.

We can determine all integral slopes attained by a line between any two points of $X \times Y$ with $\mathcal{O}(\binom{d^2}{2}) = \mathcal{O}(d^4)$ arithmetic operations in $\mathbb{Q}$. Picking the smallest natural number not occurring among these slopes gives the desired m. $\square$

The following proposition deals with the kth call of Algorithm 4.1 in Line 6 of Algorithm 4.2 running the `sequential` strategy. Recall that the strategy sequentially computes the projections $\mathrm{Trop}(I)_{\{1\}}$, $\mathrm{Trop}(I)_{\{1,2\}}$, … until $\mathrm{Trop}_{\{1,\dots,n\}} = \mathrm{Trop}(I)$.

PROPOSITION 5.7. *Let $I \subseteq K[x_1, \dots, x_n]$ be any zero-dimensional ideal of degree d. Let $k \in \{2, \dots, n\}$ and suppose Algorithm 4.1 is called with input*

- *T, a triangular set generating a zero-dimensional ideal $I \subseteq K[\mathbf{x}^{\pm}]$,*
- *$\mathrm{Trop}(I)_{\{1,\dots,k-1\}}$ and $\mathrm{Trop}(I)_{\{k\}}$.*

Moreover, assume that the following are known from the previous call of Algorithm 4.1 in Line 6 of Algorithm 4.2 running the `sequential` strategy:

- *$\varphi_{u'}$, a slim transformation such that the restriction of $\pi_{u'}$ to* $\mathrm{Trop}(I)_{\{1,\dots,k-1\}}$ *is injective,*
- *T', the triangular generating set of $\varphi_{u'}(I)$.*

Then, Algorithm 4.1 for gluing the two projections into $\mathrm{Trop}(I)_{\{1,\dots,k\}}$ *requires at most $\mathcal{O}(d^2(n+d))$ and $\mathcal{O}(d^4)$ arithmetic operations in K and $\mathbb{Q}$, respectively.*

Proof. Applying Lemma 5.6 to $X := \pi_{u'}(\mathrm{Trop}(I)_{\{1,\dots,k-1\}})$ and $Y := \mathrm{Trop}(I)_{\{k\}}$, we can compute a minimal $m \leq \binom{d^2}{2}$ such that $(a,b) \mapsto a - mb$ is injective on $X \times Y$ in $\mathcal{O}(d^4)$ arithmetic operations in $\mathbb{Q}$. Setting $u := u' + me_k$, this means that π_u is injective on $\mathrm{Trop}(I)_{\{1,\dots,k-1\}} \times \mathrm{Trop}(I)_{\{k\}}$.

Since $\varphi_u(I) = \varphi_v(\varphi_{u'}(I))$ for $v := me_k - e_\ell$ and the triangular generating set of $\varphi_{u'}(I)$ is already known, we may compute the triangular generating set T' of $\varphi_w(I)$ by applying Algorithm 3.3 to the input T' and φ_v. By Proposition 5.4, this requires $\mathcal{O}(nd^2 \log m) = \mathcal{O}(nd^2 \log d)$ arithmetic operations in K.

By Lemma 5.5, computing the eliminant $\mu \in K[x_1]$ requires $\mathcal{O}(d^3)$ arithmetic operations in K, so the overall number of arithmetic operations in K required for Algorithm 4.1 is $\mathcal{O}(d^2(n+d))$. □

Theorem 5.8. *Let I be any zero-dimensional ideal in $K[x_1,\dots,x_n]$ of degree d. Algorithm 4.2, which computes the zero-dimensional tropical variety* $\mathrm{Trop}(I)$, *with the* `sequential` *strategy requires at most $\mathcal{O}(nd^2(n+d))$ and $\mathcal{O}(nd^4)$ arithmetic operations in K and $\mathbb{Q}$, respectively.*

Proof. Running Algorithm 4.2 with the `sequential` strategy consists of the following non-trivial operations:

- computing eliminants $\mu_k \in K[x_k]$ for $k = 1,\dots,n$ in Line 2,
- applying Algorithm 4.1 to $\mathrm{Trop}(I)_{\{1,\dots,k-1\}}$ and $\mathrm{Trop}(I)_{\{k\}}$ for $k = 2,\dots,n$ in Line 5.

Combining Lemma 5.5 and Proposition 5.7 then yields the stated bounds. □

Corollary 5.9. *Let $I \subseteq K[x_1, \ldots, x_n]$ be any zero-dimensional ideal in of degree d. Given any Gröbner basis of I, computing* $\operatorname{Trop}(I)$ *requires at most polynomially many (in terms of n and d) arithmetic operations in K and $\mathbb{Q}$, respectively.*

Proof. Using polynomially many arithmetic operations in K, any Gröbner basis may be transformed to a lexicographical Gröbner basis by Faugère *et al.* (1993), and any lexicographical Gröbner basis may be decomposed into triangular sets by Lazard (1992). The claim then follows from Theorem 5.8. □

Remark 5.10 (Comparison with Magma). *In Section 7, we compare timings of Algorithm 4.2 to the* Magma *script in Appendix A in the special case that the ideal is in shape position, i.e. generated by a set*

$$\{x_n^d - f_n,\ x_{n-1} - f_{n-1},\ \ldots,\ x_1 - f_1\} \quad \text{for some } f_n, \ldots, f_1 \in K[x_n].$$

Our algorithm was implemented with the practical optimizations outlined in Remark 3.4 and Remark 4.4. The Magma *script uses a p-adic approximation of the roots of the univariate polynomial $x_n^d - f_n$ and substitution into $f_{n-1}, \ldots, f_1$.*

It is difficult to compare the two implementations in terms of complexity due to their fundamentally different nature. As the Magma *script factorizes $x_n^d - f_n$, its complexity depends on the valuation of the discriminant of $x_n^d - f_n$ (Ford & Veres 2010). Moreover, the root approximations need to be of sufficiently high precision to determine the valuation of the substituted polynomials.*

In the best case, such as when generating set is a tropical basis (Hofmann & Ren 2018, Proposition 2.16), the Magma *script terminates instantaneously:*

- *the valuations of the roots of $x_n^d - f_n$ are distinct and thus may be read off the slopes of its Newton polygon,*
- *the valuations of $f_{n-1}(z), \ldots, f_1(z)$, z a root of f_n, are uniquely determined by the valuation of z.*

Note that in general the valuation of the discriminant of $x_n^d - f_n$ and hence the complexity for its factorization depend in particular on the coefficients of f_n.

6. Implementation

In this section, we reflect on design decisions that were made in the implementation of the algorithms available in the Singular library `tropicalProjection.lib`. The library contains an implementation of Algorithms 3.3, 4.1, and 4.2 for the special case that the ideal is in shape position, as discussed in Remarks 3.4 and 4.4. We chose to focus our implementation on the case of shape position. This is the generic case and arbitrary ideals may be transformed into shape position by applying a unimodular transformation on the lattice of Laurent monomials to the generators of the ideal.

While the reader who is only interested in the algorithms and their complexity may skip this section without impeding their understanding, we thought it important to include this section for the reader who is interested in the actual implementation.

6.1. Picking slim transformations in Algorithm 4.1 Line 2. As $\pi_u|_T$ is injective for generic $u \in \mathbb{Z}_{\geq 0}^{n-1}$, it seems reasonable to sample random $u \in \mathbb{Z}_{\geq 0}^{n-1}$ until the corresponding projection is injective on the candidate set. Our implementation, however, iterates over all $u \in \mathbb{Z}_{\geq 0}^{n-1}$ in increasing ℓ_1-norm until the smallest one with injective $\pi_u|_T$ is found. This is made in an effort to keep the slim transformation $\varphi_u(I)$ as simple as possible, since Lines 3–4 are the main bottlenecks of our algorithm.

6.2. Transforming Gröbner bases in Algorithm 4.1 Line 3. As mentioned, Lines 3–4 are the main bottlenecks of our algorithm. Two common reasons why polynomial computations may scale badly are explosions in degree or in coefficient size.

Note that the degree of the polynomials is unproblematic in our algorithm: By Remark 3.4.(i), using Algorithm 3.3 in Line 3 only incurs basic arithmetic operations in $K[x_n]/\langle x_n^d - f_n\rangle$ whose elements can be represented by polynomials of degree less than d. Also, the degree of the eliminant in Line 4 is naturally bounded by d. Coefficient explosion can be a problem in large examples, which is why we choose u as small as possible for the transformation.

6.3. Computing eliminants in Algorithm 4.1 Line 4 and Algorithm 4.2 Line 2. The computation of eliminants of an ideal in shape position, say generated by

$$\{x_n^d - f_n, x_{n-1} - f_{n-1}, \dots, x_1 - f_1\} \quad \text{with } f_i \in K[x_n],$$

can be carried out in many different ways. For example:

Resultants: We can compute the resultant of the two polynomials $x_n^d - f_n$ and $x_\ell - f_\ell \in K[x_\ell, x_n]$ with respect to the variable x_n by standard resultant algorithms. The eliminant $\mu_\ell \in K[x_\ell]$ lies somewhere between the resultant and its squarefree part. In particular, the tropical variety of the eliminant is the tropical variety of the resultant.

Minimal polynomial: Note that the eliminant $\mu_\ell \in K[x_\ell]$ is also the minimal polynomial of $\overline{f}_\ell \in K[x_n]/\langle x_n^d - f_n\rangle$. Hence, it can be computed using by standard minimal polynomial algorithms.

Gröbner bases: Note that $\{x_n^d - f_n, x_\ell - f_\ell\} \subseteq K[x_\ell, x_n]$ is a Gröbner basis with respect to the lexicographical ordering with $x_n \prec x_\ell$. We can transform this to a Gröbner basis with respect to the lexicographical ordering with $x_\ell \prec x_n$ and read off the eliminant μ from it.

For polynomials with small coefficients, the implementation using Singular's resultants seemed the fastest, but Singular's `FGLM` (Faugère *et al.* 1993) seems to be best when dealing with very large coefficients, though that may be due to implementation.

For $K = \mathbb{Q}$, however, we can use a modular approach thanks to the Singular library `modular.lib` (Steenpass 2019): It computes the eliminants over $\mathbb{F}_p$ for several primes p using any of the above methods, then lifts the results to $\mathbb{Q}$. This modular approach avoids problems caused by very large coefficients and works particularly well using the method based on minimal polynomials from above. Note that `modular.lib` only checks correctness of the lifted μ_ℓ probabilistically by picking additional primes p and comparing the reduction in μ_ℓ modulo p with the eliminant over $\mathbb{F}_p$. However, if we are in the case that the resulting μ_ℓ is of maximal degree d, one can verify correctness by testing whether $\mu_\ell(\overline{f}_\ell) = 0$ in $K[x_n]/\langle x_n^d - f_n\rangle$.

6.4. Picking gluing strategies in Algorithm 4.2 Line 5. Algorithm 4.2 is formulated in a flexible way. Different strategies of realising the choice of coordinate sets $A_1, \dots, A_k$ in Line 5 can adapt to the needs of a specific tropicalization problem. The four gluing strategies that are implemented in our Singular library are (see Figure 6.1 for an illustration in the case $n = 5$):

`oneProjection`: Only a single iteration of the `while` loop, in which we pick $k = n$ and $A_i = \{i\}$ for $i = 1, \dots, n$.

`sequential`: $n-1$ iterations of the `while` loop, during which we pick $k = 2$ and $A_1 = \{1, \dots, i\}$ and $A_2 = \{i+1\}$ in the ith iteration.

`regularTree(`k`)`: $n-1$ iterations of the `while` loop, which can be partially run in parallel in $\lceil \log_k n \rceil$ batches. In each batch, we merge k of the previous projections. Note that, by Remark 3.4.(ii), the condition that $1 \in A_1 \cup \dots A_k$ is unnecessary if the ideal is in shape position.

`overlap`: $(n-1)n/2$ iterations of the `while` loop, which can be partially run in parallel in $n-1$ batches. During batch i, we pick $k = 2$ and $A_1 = \{1, \dots, i\}$, $A_2 = \{1, \dots, i-1, j\}$ for $j > i$.

`oneProjection` is the simplest strategy, requiring only a single slim transformation. For examples of very low degree, it is the best strategy due to its minimal overhead. For examples of higher degree d, the candidate set C in Algorithm 4.1 can become quite large, at worst $|C| = d^n$. This generally leads to larger $u \in \mathbb{Z}_{\geq 0}^{n-1}$ in Line 2 and causes problems due to coefficient growth.

`sequential` avoids the problem of a large candidate set C by only gluing two projections at a time, guaranteeing $|C| \leq d^2$. This comes at the expense of computing $n-1$ slim transformations, but even for medium-sized instances we observe considerable improvements compared to `oneProjection`. In Section 5, we have proved that `sequential` guarantees good complexity bounds on Algorithm 4.2.

`regularTree(`k`)` can achieve a considerable speed-up by parallelisation. Whereas every `while`-iteration in `sequential` depends

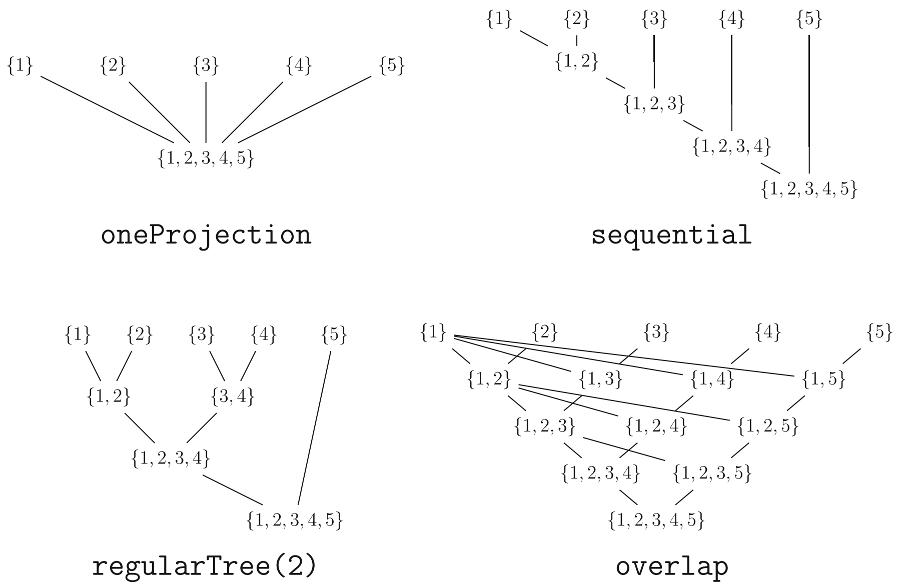


Figure 6.1: Visualisation of different gluing strategies.

on the output of the previous iteration, `regularTree(`k`)` allows us to compute all gluings in parallel in $\lceil \log_k n \rceil$ batches. The total number of gluings remains the same.

`overlap` further reduces the size of the candidate set C compared to `sequential`, while exploiting parallel computation like `regularTree(`k`)`. It glues projections two at a time, but only those A_1 and A_2 which overlap significantly. This can lead to much smaller candidate sets T, at best $|T| = d$, which makes a slim transformation obsolete. The strategy `overlap` seems particularly successful in practice and is the one used for the timings in Section 7.

Our implementation in Singular also allows for custom gluing strategies by means of specifying a graph as in Figure 6.1.

7. Timings

In this section, we present timings of our Singular implementation of Algorithm 4.2 for $K = \mathbb{Q}$ and the 2-adic valuation. We compare it to a Magma (Bosma *et al.* 1997) implementation by

Avi Kulkarni which uses univariate factorization and backsubstitution, see Appendix A. Note that MAGMA operates under finite absolute precision, which was chosen to be 2^{1000} by default and was increased if needed. While SINGULAR is also capable of the same task, we chose to compare to MAGMA instead as it is significantly faster due to its finite precision arithmetic over p-adic numbers. Our SINGULAR timings use the `overlap` strategy, a modular approach and parallelisation with up to four threads. The SINGULAR times we report on are total CPU times across all threads. (For reference, the longest example in SINGULAR required 118 seconds total CPU time, but only 32 seconds real time.) All computations were run on a server with 2 Intel Xeon Gold 6144 CPUs, 384GB RAM and Debian GNU/Linux 9.9 OS. All examples and scripts are available at https://software.mis.mpg.de.

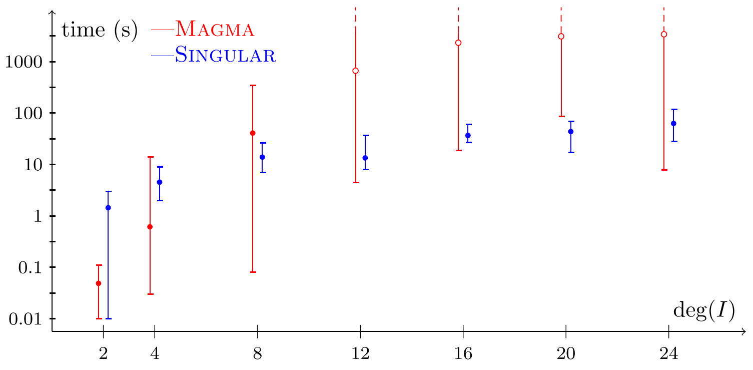
7.1. Random lexicographical Gröbner bases in shape position. Given $d, n \in \mathbb{Z}_{>0}$, we construct random lexicographical Gröbner bases $G \subseteq \mathbb{Q}[x_1, \dots, x_n]$ of degree d in shape position of the form

$$G = \{x_n^d - f_n,\ x_{n-1} - f_{n-1},\ x_{n-2} - f_{n-2},\ \dots,\ x_2 - f_2,\ x_1 - f_1\},$$

where $f_n, \dots, f_1$ are univariate polynomials in x_n of degree $d-1$ with coefficients of the form $2^\lambda \cdot (2k+1)$ for random $\lambda \in \{0, \dots, 99\}$ and random $k \in \{0, \dots, 4999\}$.

Figure 7.1 shows timings for $n = 5$ and varying d. Each computation was aborted if it failed to terminate within one hour. We see that MAGMA is significantly faster for small examples, while SINGULAR scales better with increasing degree.

For many of the ideals I, however, $\mathrm{Trop}(I)$ has fewer than d distinct points. This puts our algorithm at an advantage, as it allows for easier projections in Algorithm 4.2 Line 2. Mathematically, it is not an easy task to generate non-trivial examples with distinct tropical points. Picking $x_n^d - f_n$ to have d roots with distinct valuation for example would make all roots live in $\mathbb{Q}_2$, in which case MAGMA terminates instantly. Our next special family of examples has criteria which guarantee distinct points.



deg(I)	2	4	8	12	16	20	24
#SINGULAR finished	100	100	100	100	100	100	100
#MAGMA finished	100	100	100	93	51	21	9
SINGULAR avg. (s)	1	5	14	19	37	44	63
MAGMA avg. (s)	0	1	41	>663	>2273	>3095	>3395

Figure 7.1: Timings for random ideals in shape position.

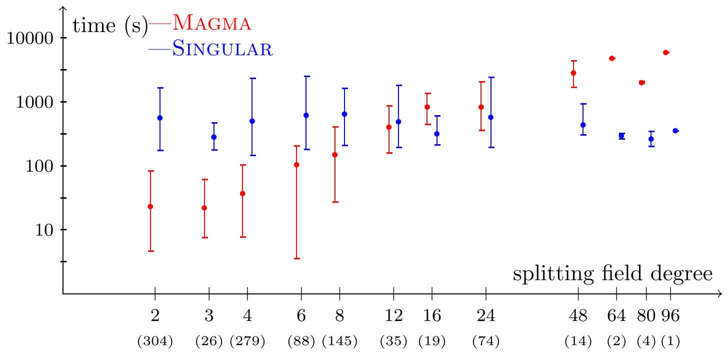
7.2. Tropical lines on a random honeycomb cubic. Let $V(f) \subseteq \mathbb{P}^3$ be a smooth cubic surface. In Panizzut & Vigeland (2019), it is shown that $\mathrm{Trop}(f) \subseteq \mathbb{R}^3$ may contain infinitely many tropical lines. However, for general f whose coefficient valuations induce a honeycomb subdivision of its Newton polytope, $\mathrm{Trop}(f)$ will always contain exactly 27 distinct tropical lines (Panizzut & Vigeland 2019, Theorem 27), which must therefore be the tropicalizations of the 27 lines on $V(f)$.

We used POLYMAKE (Gawrilow & Joswig 2000) to randomly generate 1000 cubic polynomials with honeycomb subdivisions whose coefficients are pure powers of 2. For each cubic polynomial f, we constructed the one-dimensional homogeneous ideal

$$\mathcal{L}_f \subseteq \mathbb{Q}[p_{12}, p_{13}, p_{14}, p_{23}, p_{24}, p_{34}]$$

of degree 27 whose solutions are the lines on $V(f)$ in Plücker coordinates. Figure 7.2 shows the timings for computing $\mathrm{Trop}(L_f)$, where $L_f := \mathcal{L}_f + \langle p_{34} - 1\rangle$ is a zero-dimensional ideal of degree 27. Out of our 1000 random cubics, 8 had to be discarded because L_f was of lower degree, i.e. $V(f)$ contained lines with $p_{34} = 0$.

Unsurprisingly, the Singular timings are relatively stable, while the Magma timings heavily depend on the degree of the splitting field of L_f over $\mathbb{Q}_2$. While the generic splitting field degree is 51840 over $\mathbb{Q}$ (Elsenhans & Jahnel 2012), the distinct tropical points of $\mathrm{Trop}(L_f)$ severely restrict the Galois group of the splitting field over $\mathbb{Q}_2$.



splitting deg.	2	3	4	6	8	12	16	24	48	64	80	96
Frequency	304	26	279	88	145	35	19	74	14	2	4	1
Singular avg.	556	281	505	610	651	490	313	580	440	294	261	352
Magma avg.	23	22	37	104	149	403	831	830	2840	4791	1998	5935

Figure 7.2: Timings for the 27 tropical lines on a tropical honeycomb cubic.

8. Conclusion

If a zero-dimensional ideal $I \subseteq K[x_1^{\pm}, \dots, x_n^{\pm}]$ is generated by a given triangular set, we have shown that the tropical variety $\mathrm{Trop}(I) \subseteq \mathbb{R}^n$ can be computed using at most $\mathcal{O}(nd^2(n+d))$ and $\mathcal{O}(nd^4)$ arithmetic operations in K and $\mathbb{Q}$, respectively. Given a Gröbner basis of a general zero-dimensional ideal, it is thus possible to compute its tropical variety using polynomially many arithmetic operations.

For the special case that the ideal is in shape position, we have implemented our algorithms in Singular using parallelization and modular techniques, and we have compared its timings for $K = \mathbb{Q}_p$ to a Magma implementation using univariate factorization and backsubstitution. The timings of our algorithm are relatively constant, while the MAGMA timings depend strongly on the degree of the splitting field: For small degrees, it severely outperformed our algorithm, while for large degrees it is significantly slower.

We would like to conclude the paper with a remark on the complexity of computing tropical varieties $\mathrm{Trop}(I)$ for $I \subseteq K[x_1, \ldots, x_n]$ and $d := \deg(I) > 0$. Currently, there are two distinct methods for computing tropical varieties:

- reconstruction from sufficiently many projections of the tropical variety (Hept & Theobald 2009),
- traversing the Gröbner complex along the tropical variety (Bogart *et al.* 2007; Markwig & Ren 2019).

Not much is known about the complexity of either method. In fact not much is known about the *combinatorial complexity* of tropical varieties in general (Joswig & Schröter 2018). Both methods generally involve Gröbner basis computations, though the projections in the first method may also be computed using numerical techniques if $K = \mathbb{C}\{\{t\}\}$ and the ideal is generated over $\mathbb{C}$ (Brysiewicz 2020). However, Theobald has shown that even simple questions on the combinatorics of the intersection of degree 3 tropical hypersurfaces, such as deciding whether it is non-empty or connected, are NP-complete and co-NP-hard (Theobald 2006). And linear tropical hypersurfaces are known to be tied to mean payoff games; therefore, problems such as deciding the dimension of their intersection are NP-complete (Grigoriev & Podolskii 2015). Due to the Fundamental Theorem of Tropical Geometry (Maclagan & Sturmfels 2015, Theorem 3.2.3), deciding whether a tropical variety $\mathrm{Trop}(I)$ is non-empty is at least as hard as deciding whether I has solutions in the torus.

Currently, GFAN and SINGULAR are the only software systems capable of computing positive-dimensional tropical varieties, and both use the traversal of the Gröbner complex. The algorithm for the traversal method consists of two main parts:

1. a Gröbner walk algorithm to walk from one Gröbner polyhedra to another,

2. a tropical link algorithm to direct the Gröbner walk along the tropical variety.

While the computation of tropical links had been a major bottleneck of the original algorithm and in early implementations, experiments suggest that it has since been resolved by new approaches (Chan 2013; Hofmann & Ren 2018). However, the algorithm in Chan (2013, §4.2) relies heavily on projections, while (Hofmann & Ren 2018, Algorithm 2.10) relies on root approximations to a possibly exponential precision, so neither approach has good complexity bounds.

Algorithm 4.2 is designed with Hofmann & Ren (2018, Algorithm 2.10) in mind. Replacing Hofmann & Ren (2018, Algorithm 2.10) in Hofmann & Ren (2018, Algorithm 4.6) with our Algorithm 4.2 allows us to compute tropical links at the cost of $2n$ Gröbner basis computations of zero-dimensional ideals of degree d, which is at most single exponential in the number of variables n (Lakshman 1991; Lakshman & Lazard 1991). The Gröbner walk, however, requires Gröbner bases computations of initial ideals with respect to weight vectors $w \in \mathrm{Trop}(I)$ with $\dim C_w(I) = \dim \mathrm{Trop}(I) - 1$, where $C_w(I)$ denotes the Gröbner polyhedron of I around w. These initial ideals are of the same dimension as I and neither monomial, as $w \in \mathrm{Trop}(I)$, nor binomial, as $\dim C_w(I) < \dim \mathrm{Trop}(I)$. Moreover, due to how Gröbner bases of $\mathrm{in}_w(I)$ can be lifted to Gröbner bases of I, the degrees of their basis elements coincide. Hence, we expect that the complexity of computing the zero-dimensional Gröbner bases required by our algorithms is less than the complexity of computing the Gröbner bases required by the traversal of the Gröbner complex. Therefore, the complexity of tropical varieties via a traversal of the Gröbner complex is still dominated by the complexity of the necessary Gröbner walk, and not the complexity of computing tropical links.

Acknowledgements

The authors would like to thank Avi Kulkarni (Dartmouth College) for his Magma script for solving polynomial equations over p-adic numbers, Marta Panizzut (TU Berlin) and Bernd Sturmfels

(MPI MiS Leipzig + UC Berkeley) for the examples of tropical cubic surfaces with 27 distinct lines, as well as Andreas Steenpaß (TU Kaiserslautern) for technical support on the SINGULAR library `modular.lib` (Steenpass 2019). The authors would also like to thank the anonymous referees for their comments and for pointing out a mistake in the previous version.

Paul Görlach and Yue Ren were partially supported by the Max Planck Institute for Mathematics in the Sciences. Paul Görlach acknowledges partial support by the DFG grant SE 1114/5-2, and Yue Ren is supported by the UKRI Future Leaders Fellowship (MR/S034463/1).

Open Access

Funding

Open Access funding enabled and organized by Projekt DEAL

A. Appendix: Magma implementation

The comparison of timings in Section 7 is based on the following Magma implementation by Avi Kulkarni for finite precision p-adic fields. The function assumes that the ideal is in shape position and uses univariate factorization and substitution.

```
function pAdicSolutionsOverSplittingField(I, Qp)
   R := Generic(I);
```

```
    gs := GroebnerBasis(I);   //assumed to be in shape position

    u := UnivariatePolynomial(gs[#gs]);
    up := ChangeRing(u,Qp);
    K := SplittingField(up);  //main bottleneck of the algorithm

   vars_padic := Variables(ChangeRing(R,K));
   padic_rts := Roots(ChangeRing(up,K));

   function backSolve(rt)
     rt_coords := [rt];
     for i in [#gs-1 .. 1 by -1]  do
      g := Evaluate(gs[i], vars_padic[1..i] cat rt_coords);
      rti := Roots(UnivariatePolynomial(g));
      assert #rti eq 1;
      Insert(~rt_coords, 1, rti[1][1]);
     end for;
     return rt_coords;
   end function;

   return [ backSolve(rt[1]) : rt in padic_rts], K;
 end function;
```

Publisher's Note

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

References

Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert & Michael Joswig (2018). Log-barrier interior point methods are not strongly polynomial. *SIAM J. Appl. Algebra Geom.* **2**(1), 140–178. ISSN 2470-6566.

Federico Ardila & Caroline J. Klivans (2006). The Bergman complex of a matroid and phylogenetic trees. *J. Combin. Theory Ser. B* **96**(1), 38–49. ISSN 0095-8956.

Elizabeth Baldwin & Paul Klemperer (2019). Understanding Preferences: Demand Types and the Existence of Equilibrium With Indivisibilities. *Econometrica* **87**(3), 867–932.

L. Bernardin, P. Chin, P. DeMarco, K. O. Geddes, D. E. G. Hare, K. M. Heal, G. Labahn, J. P. May, J. McCarron, M. B. Monagan, D. Ohashi & S. M. Vorkoetter. (1996-2020). *Maple Programming Guide.* Maplesoft, a division of Waterloo Maple Inc., Waterloo ON, Canada.

T. Bogart, A. N. Jensen, D. Speyer, B. Sturmfels & R. R. Thomas (2007). Computing tropical varieties. *J. Symbolic Comput.* **42**(1-2), 54–73. ISSN 0747-7171.

Wieb Bosma, John Cannon & Catherine Playoust (1997). The Magma algebra system. I. The user language. *J. Symb. Comput.* **24**(3-4), 235–265. ISSN 0747-7171. Computational algebra and number theory (London, 1993).

Taylor Brysiewicz (2020). Numerical Software to Compute Newton polytopes and Tropical Membership. *Math. Comput. Sci.* **14**, 577–589.

Peter Bürgisser, Michael Clausen & M. Amin Shokrollahi (1997). *Algebraic complexity theory. With the collaboration of Thomas Lickteig*, volume 315. Berlin: Springer. ISBN 3-540-60582-7/hbk, xxiii + 618 .

Andrew Chan (2013). *Gröbner bases over fields with valuation and tropical curves by coordinate projections.* Ph.D. Thesis, University of Warwick.

Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister & Hans Schönemann (2019). Singular 4-1-2 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de.

Alicia Dickenstein & Ioannis Z. Emiris (editors) (2005). *Solving polynomial equations*, volume 14 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin. ISBN 978-3-540-24326-7; 3-540-24326-7, xiv+425 . Foundations, algorithms, and applications.

Andreas-Stephan Elsenhans & Jörg Jahnel (2012). The discriminant of a cubic surface. *Geom. Dedicata* **159**, 29–40. ISSN 0046-5755.

J. C. Faugère, P. Gianni, D. Lazard & T. Mora (1993). Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.* **16**(4), 329–344. ISSN 0747-7171.

Eva Maria Feichtner & Bernd Sturmfels (2005). Matroid polytopes, nested sets and Bergman fans. *Port. Math. (N.S.)* **62**(4), 437–468. ISSN 0032-5155.

David Ford & Olga Veres (2010). On the Complexity of the Montes Ideal Factorization Algorithm. In *Algorithmic Number Theory*, Guillaume Hanrot, François Morain & Emmanuel Thomé, editors, 174–185. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-14518-6.

Joachim von zur Gathen & Jürgen Gerhard (2013). *Modern computer algebra*. Cambridge University Press, Cambridge, 3rd edition. ISBN 978-1-107-03903-2, xiv+795 .

Ewgenij Gawrilow & Michael Joswig (2000). `polymake`: a framework for analyzing convex polytopes. In *Polytopes—combinatorics and computation (Oberwolfach, 1997)*, volume 29 of *DMV Sem.*, 43–73. Birkhäuser, Basel.

Gert-Martin Greuel & Gerhard Pfister (2002). *A* Singular *introduction to commutative algebra*. Springer-Verlag, Berlin. ISBN 3-540-42897-6, xviii+588 . With contributions by Olaf Bachmann, Christoph Lossen and Hans Schönemann.

Dima Grigoriev & Vladimir V. Podolskii (2015). Complexity of Tropical and Min-plus Linear Prevarieties. *Comput. Complex.* **24**(1), 31–64. ISSN 1016-3328.

Marshall Hampton & Anders Jensen (2011). Finiteness of spatial central configurations in the five-body problem. *Celest. Mech. Dyn. Astron.* **109**(4), 321–332. ISSN 0923-2958.

Marshall Hampton & Richard Moeckel (2006). Finiteness of relative equilibria of the four-body problem. *Invent. Math.* **163**(2), 289–312. ISSN 0020-9910.

Kerstin Hept & Thorsten Theobald (2009). Tropical bases by regular projections. *Proc. Am. Math. Soc.* **137**(7), 2233–2241. ISSN 0002-9939.

Tommy Hofmann & Yue Ren (2018). Computing tropical points and tropical links. *Discrete Comput. Geom.* **60**(3), 627–645. ISSN 0179-5376.

Anders N. Jensen (2017). Gfan 0.6.2, a software system for Gröbner fans and tropical varieties. Available at `http://home.imf.au.dk/jensen/software/gfan/gfan.html`.

Michael Joswig & Benjamin Schröter (2018). The degree of a tropical basis. *Proc. Am. Math. Soc.* **146**(3), 961–970. ISSN 0002-9939.

Y. N. Lakshman (1991). A single exponential bound on the complexity of computing Gröbner bases of zero-dimensional ideals. In *Effective methods in algebraic geometry (Castiglioncello, 1990)*, volume 94 of *Progr. Math.*, 227–234. Birkhäuser Boston, Boston, MA.

Y. N. Lakshman & Daniel Lazard (1991). On the complexity of zero-dimensional algebraic systems. In *Effective methods in algebraic geometry (Castiglioncello, 1990)*, volume 94 of *Progr. Math.*, 217–225. Birkhäuser Boston, Boston, MA.

D. Lazard (1992). Solving zero-dimensional algebraic systems. *J. Symb. Comput.* **13**(2), 117–131. ISSN 0747-7171.

Bo Lin, Anthea Monod & Ruriko Yoshida (2018). Tropical Foundations for Probability & Statistics on Phylogenetic Tree Space. Eprint: `arXiv:1805.12400`.

Diane Maclagan & Bernd Sturmfels (2015). *Introduction to tropical geometry*, volume 161 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI. ISBN 978-0-8218-5198-2, xii+363 .

Thomas Markwig & Yue Ren (2019). Computing Tropical Varieties Over Fields with Valuation. *Found. Comput. Math.* ISSN 1615-3383.

Grigory Mikhalkin (2005). Enumerative tropical algebraic geometry in $\mathbb{R}^2$. *J. Am. Math. Soc.* **18**(2), 313–377. ISSN 0894-0347.

Jürgen Neukirch (1999). *Algebraic number theory*, volume 322 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, Berlin. ISBN 3-540-65399-6.

Marta Panizzut & Magnus Dehli Vigeland (2019). Tropical Lines on Cubic Surfaces. Eprint: `arXiv:0708.3847v2`.

David Speyer & Bernd Sturmfels (2004). The tropical Grassmannian. *Adv. Geom.* **4**(3), 389–411. ISSN 1615-715X.

Andreas Steenpass (2019). `modular.lib`. A Singular 4-1-2 library for modular techniques.

Bernd Sturmfels (2002). *Solving systems of polynomial equations*, volume 97 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI. ISBN 0-8218-3251-4, viii+152 .

Thorsten Theobald (2006). On the frontiers of polynomial computations in tropical geometry. *J. Symb. Comput.* **41**(12), 1360–1375. ISSN 0747-7171.

Ngoc Mai Tran & Josephine Yu (2019). Product-mix auctions and tropical geometry. *Math. Oper. Res.* **44**(4), 1396–1411. ISSN 0364-765X; 1526-5471/e.

Manuscript received 27 January 2020

Paul Görlach
Institute of Algebra and Geometry
Otto von Guericke University
39106 Magdeburg, Germany
paul.goerlach@ovgu.de
https://www.ovgu.de/goerlach

Yue Ren
Department of Mathematics
Durham University
Durham DH1 3LE, UK
yue.ren2@durham.ac.uk
https://yueren.de

Leon Zhang
University of California, Berkeley
Evans Hall
Berkeley, CA 94720-3840, USA
leonyz@berkeley.edu
https://math.berkeley.edu/~leonyz/