FACTORIZATION OF TROPICAL POLYNOMIALS
IN ONE AND SEVERAL VARIABLES

by

Nathan B. Grigg

Submitted to Brigham Young University in partial fulfillment of graduation requirements for University Honors

Department of Mathematics
Brigham Young University
June 2007

Advisor: Tyler Jarvis  Honors Representative: Steve Turley
Signature: __________________________  Signature: __________________________
This paper discusses the theory of factoring tropical polynomials as well as some factoring techniques. I give an elementary proof of the Fundamental Theorem of Tropical Algebra and describe an algorithm to factor single-variable polynomials with rational coefficients. For multi-variable tropical polynomials I discuss the corner locus and a correspondence between polynomials and weighted balanced complexes. I prove that tropical factorization is NP-complete, and I give an algorithm that is efficient for many classes of polynomials.
I would like to thank all of those who have helped with this project. Thanks to Gretchen Rimmasch for being a good friend and mentor, to Nathan Manwaring, Natalie Wilde, and Julian Tay for their help, ideas, and support.

Thanks to my advisor, Dr. Jarvis, for making time in his very busy schedule to help me when I needed it. Also, thanks to the BYU Math Department and the Office of Research and Creative Activities for providing me with monetary support.

Most of all, thanks to my Amy for her love, support, and motivation. She has helped me more (and read this thesis more times) than anyone else that I know.
Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title and signature page</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>vii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 The Rational Tropical Semi-ring</td>
<td>3</td>
</tr>
<tr>
<td>2 Tropical Polynomials of One Variable</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Polynomials and Functions</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Least-coefficient Polynomials</td>
<td>8</td>
</tr>
<tr>
<td>2.3 The Fundamental Theorem of Tropical Algebra</td>
<td>13</td>
</tr>
<tr>
<td>2.4 The Corner Locus of a Polynomial</td>
<td>15</td>
</tr>
<tr>
<td>3 Tropical Polynomials of Several Variables</td>
<td>19</td>
</tr>
<tr>
<td>3.1 The Corner Locus and Weighted Balanced Complex</td>
<td>20</td>
</tr>
<tr>
<td>3.1.1 The Corner Locus</td>
<td>20</td>
</tr>
<tr>
<td>3.1.2 The Weighted Balanced Complex</td>
<td>21</td>
</tr>
<tr>
<td>3.1.3 Polynomial-Complex Correspondence</td>
<td>24</td>
</tr>
<tr>
<td>3.2 Factors and Balanced Subcomplexes</td>
<td>26</td>
</tr>
<tr>
<td>3.3 A Factoring Algorithm</td>
<td>29</td>
</tr>
<tr>
<td>3.3.1 NP-completeness</td>
<td>29</td>
</tr>
<tr>
<td>3.3.2 A factoring algorithm</td>
<td>30</td>
</tr>
<tr>
<td>3.3.3 Analysis</td>
<td>32</td>
</tr>
<tr>
<td>4 Conclusion</td>
<td>35</td>
</tr>
<tr>
<td>Appendix: The Tropical Maple Package</td>
<td>37</td>
</tr>
<tr>
<td>Bibliography</td>
<td>59</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Functional equivalence, equality, and least coefficients . . . . . . . . . . 7
2.2 A factoring example . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.1 The corner locus of a polynomial in two variables . . . . . . . . . . . 21
3.2 A weighted balanced 2-complex . . . . . . . . . . . . . . . . . . . . . 24
3.3 Weighted balanced graph of a product . . . . . . . . . . . . . . . . . 28
3.4 Decomposing a Newton polyhedron by factoring a tropical polynomial 30
3.5 A two-variable factoring example . . . . . . . . . . . . . . . . . . . . 33
Chapter 1

Introduction

Tropical algebra, also called “min-plus” or “max-plus” algebra, is a relatively new topic in mathematics that has recently caught the interest of algebraic geometers, computer scientists, combinatorists, and other mathematicians. According to Andreas Gathmann [5], tropical algebra was pioneered by mathematician and computer scientist Imre Simon in the 1980s but did not receive widespread attention until a few years ago. Interestingly, the title “tropical algebra” is nothing more than a reference to Simon’s home country—Brazil.

Tropical algebra is a useful tool in solving a wide variety of problems. For example, it can be used to solve complicated phylogenetic problems [2, 14], to simplify mathematical physics problems [8], and to study implications of statistical models [11]. In addition, tropical algebra often provides an easier way to solve complex enumerative geometry problems [6, 10]. It can even be used to find and assess algorithms for solving the classic traveling salesman problem [1].

The tropical semi-ring, which is the basis of this problem-solving tool and our major object of study, is formed by replacing the standard addition operation with the minimum function and replacing multiplication with addition (see Definition 1).
Although many difficult problems can be translated into simpler tropical problems, even these simpler problems can be difficult if we do not understand the tropical algebraic system. Since tropical algebra is so new, several things about it are unknown. Our purpose in studying it, then, is to understand it better, since the more we understand about the mechanics of the tropical algebraic system, the more effective this tool becomes.

To provide a better understanding of this system, this thesis concentrates on factoring tropical polynomials. There are two reasons we want to factor polynomials. First, it helps us to better study other areas of tropical algebra, which makes it easier to solve some of the problems mentioned above. Second, factoring tropical polynomials helps us to factor classical polynomials [4, 5].

In this paper I discuss the factorization of polynomials over the rational tropical semi-ring. I discuss and present an elementary proof of the Fundamental Theorem of Tropical Algebra. I show that every tropical polynomial of a single variable can be factored completely into linear factors by a simple and straightforward process. I also discuss the ideas of functional equivalence of tropical polynomials and least-coefficient polynomials, two very important themes in tropical algebra.

I also discuss the factorization of polynomials of several variables. I prove that factorization is NP-complete, and give an algorithm that runs in polynomial time for some classes of polynomials. Since this algorithm depends on the properties of a polynomial’s zero locus, I also discuss the theory of tropical zero loci. I discuss weighted balanced complexes, as defined by Mikhalkin in [9].
1.1 The Rational Tropical Semi-ring

Definition 1. The rational tropical semi-ring is \( \mathbb{Q} = (\mathbb{Q} \cup \{\infty\}, \oplus, \odot) \), where

\[
a \oplus b := \min(a, b), \quad \text{and} \\
da \odot b := a + b.
\]

We note that the additive identity of \( \mathbb{Q} \) is \( \infty \), in the sense that \( a \oplus \infty = a \) for all \( a \in \mathbb{Q} \). Similarly, the multiplicative identity is 0, since \( a \odot 0 = a \) for all \( a \in \mathbb{Q} \). Elements of \( \mathbb{Q} \) do not have additive inverses, but the multiplicative inverse of \( a \) is the classical negative \( a \). The commutative, associative, and distributive properties hold.

Notation. To simplify our notation, we will generally write tropical multiplication \( a \odot b \) as just \( ab \), and we will write repeated multiplication \( a \odot a \) as \( a^2 \). Since the multiplicative inverse of an element \( a \) is negative \( a \), we will write it as either \( a^{-1} \) or \( -a \). We will write classical addition, subtraction, multiplication, and division as \( a + b \), \( a - b \), \( a \cdot b \), and \( \frac{a}{b} \), respectively.

For example, in a tropical polynomial a monomial \( ax^b \) corresponds to the classical function \( a + b \cdot x \). A general polynomial \( a_nx^n \oplus a_{n-1}x^{n-1} \oplus \cdots \oplus a_1x \oplus a_0 \) corresponds to the classical function \( \min \{a_n + n \cdot x, a_{n-1} + (n - 1) \cdot x, \ldots, a_1 + x, a_0\} \). If we write \( x^n \) without a coefficient, we mean \( 0x^n \), since 0 is the multiplicative identity. Also note that \( 1x^n \neq x^n \), so if we mean \( 1x^n \), we must write \( 1x^n \).
Chapter 2

Tropical Polynomials of One Variable

We begin by discussing the semi-ring $\mathbb{Q}[x]$—tropical polynomials of one variable. The goal of this chapter is to present an elementary proof of the Fundamental Theorem of Tropical Algebra\footnote{This theorem is referred to as the Fundamental Theorem of Tropical Algebra because it is the tropical analogue of the Fundamental Theorem of Algebra. Clearly, however, this theorem is separate from (and not a consequence of) the classical Fundamental Theorem.}, which states

**Fundamental Theorem of Tropical Algebra.** *Every tropical polynomial in one variable with rational coefficients can be factored uniquely as a product of linear tropical polynomials with rational coefficients, up to functional equivalence.*\footnote{This theorem is only true if we consider two polynomials to be equivalent if they define the same function, as we do in this thesis. See Section 2.1}

Although the authors of some papers refer to this theorem, they only mention it to confirm it as true or dismiss it as trivial. Nevertheless, our proof of this theorem is key to understanding vital components of the tropical algebraic structure. Note that one version of the proof has been published by Izhakian in \cite{Izhakian}, but Izhakian gives his proof over an “extended” tropical semi-ring that is substantially different.
from the standard tropical semi-ring that most others study. Hence, there is merit in discussing this elementary proof and the underlying ideas it addresses.

To prove this theorem, we will first discuss the idea of functional equivalence, in Section 2.1. Tropically, it is most useful to consider two polynomials as equivalent if the functions they define are equivalent. In Section 2.2, we will discuss a best representative for each functional equivalence class, called a least-coefficient polynomial. We will prove that every tropical polynomial is equivalent to a least-coefficient polynomial and that each least-coefficient polynomial can be easily factored using the formula given in Section 2.3.

### 2.1 Polynomials and Functions

A polynomial \( f(x) \in \mathbb{Q}[x] \) is defined to be a formal sum

\[
    f(x) = a_n x^n \oplus a_{n-1} x^{n-1} \oplus \cdots \oplus a_0.
\]

For two polynomials \( f \) and \( g \), we write \( f = g \) if each pair of corresponding coefficients of \( f \) and \( g \) are equal.

We can also think of a tropical polynomial as a function. Two polynomials \( f \) and \( g \) are functionally equivalent (written \( f \sim g \)) if for each \( x \in \mathbb{Q} \), \( f(x) = g(x) \). It is easy to check that functional equivalence is in fact an equivalence relation. Notice that functional equivalence does not imply equality. For example, the polynomials \( x^2 \oplus 1x \oplus 2 \) and \( x^2 \oplus 2x \oplus 2 \) are functionally equivalent\(^3\) but not equal as polynomials.

In general, functional equivalence is a more useful equivalence relation to use with tropical polynomials than equality of coefficients.

\(^3\)That is, the classical functions that these polynomials correspond to—\( \min(2 \cdot x, 1 + x, 2) \) and \( \min(2 \cdot x, 2 + x, 2) \) respectively—are equal as functions.
The function graph of a polynomial. For a polynomial of one variable, we can plot the input $x$ versus the output $f(x)$ to better visualize the difference between equality and functional equivalence. For example, the polynomial $x^2 \oplus 1x \oplus 2$ mentioned above can be described classically as $\min\{2 \cdot x, 1 + x, 2\}$. So each term in the polynomial can be represented by a classical line, and a polynomial’s function corresponds to the lowest of these lines at each point, as shown in Figure 2.1.

![Figure 2.1: Although $f(x)$ and $g(x)$ are not equal as polynomials, they are functionally equivalent. Also, the $1x$ in $f(x)$ is a least-coefficient term, while the $2x$ in $g(x)$ is not.](image)

When dealing with equivalence classes, it is useful to have a representative of each equivalence class. One useful representative of the equivalence class $[f(x)]$ is the polynomial formed by making each of the coefficients of $f(x)$ as small as possible without changing the function defined by $f(x)$.

**Definition 2.** A coefficient $a_i$ of a polynomial $f(x)$ is a least coefficient if for any $b \in \mathbb{Q}$ with $b < a_i$, the polynomial $g(x)$ formed by replacing $a_i$ with $b$ is not functionally equivalent to $f(x)$.

**Note.** If $f(x) = a_nx^n \oplus a_{n-1}x^{n-1} \oplus \cdots \oplus a_rx^r$, where $a_n, a_r \neq \infty$, then $a_n$ and $a_r$ are least coefficients. Additionally, if $r < i < n$ and $a_i = \infty$, then $a_i$ is not a least coefficient.
Example. Let \( f(x) = x^2 \oplus 1x \oplus 2 \) and let \( g(x) = x^2 \oplus 2x \oplus 2 \), as in Figure [2.1]. Then the \( 2x \) in \( g(x) \) is not a least-coefficient term, since we can replace the 2 with a 1 and still have the same function. The \( 1x \) in \( f(x) \), however, is a least-coefficient term, since replacing the 1 with any number less than 1 would form a polynomial with a different function than \( f(x) \).

Lemma 1 (Alternate definition of least coefficient). Let \( a_i x^i \) be a term of a polynomial \( f(x) \), with \( a_i \) not equal to infinity. Then \( a_i \) is a least coefficient of \( f(x) \) if and only if there is some \( x_0 \in \mathbb{Q} \) such that \( f(x_0) = a_i x_0^i \).

Proof. For all \( x \in \mathbb{Q} \), note that \( f(x) \leq a_i x^i \). Suppose that there is no \( x \) such that \( f(x) = a_i x^i \). Then \( f(x) < a_i x^i \) for all \( x \). Now, let \( \varphi(x) = f(x) - (i \cdot x + a_i) \). Note that \( \varphi \) is a piecewise-linear, continuous function that is linear over a finite number of intervals. Thus, there is an interval large enough to contain all the pieces of \( \varphi \). By applying the extreme value theorem to this interval, we see that \( \sup \varphi \in \varphi(\mathbb{R}) \), and hence \( \sup \varphi < 0 \). Let \( \epsilon = |\sup \varphi| \) and \( b \in \mathbb{Q} \) be such that \( a_i - \epsilon < b < a_i \). Then

\[
f(x) - (i \cdot x + b) < f(x) - (i \cdot x + a_i) + \epsilon \leq 0
\]

and therefore \( f(x) < i \cdot x + b \) for all \( x \in \mathbb{Q} \). Since the polynomial created by replacing \( a_i \) with \( b \) is functionally equivalent to \( f(x) \), \( a_i \) is not a least coefficient.

For the other direction, suppose that there is an \( x_0 \in \mathbb{Q} \) such that \( f(x_0) = a_i x_0^i \). Given \( b < a_i \), let \( g(x) \) be \( f(x) \) with \( a_i \) replaced by \( b \). Then \( g(x_0) \leq bx_0^i < a_i x_0^i = f(x_0) \), so \( g \) is not functionally equivalent to \( f \). Therefore \( a_i \) is a least coefficient. \( \Box \)

2.2 Least-coefficient Polynomials

Definition 3. A polynomial is a least-coefficient polynomial if all its coefficients are least coefficients.
2.2 Least-coefficient Polynomials

Lemma 2 (Uniqueness of least-coefficient polynomials). Let $f$ and $g$ be tropical least-coefficient polynomials. Then $f$ is equal to $g$ if and only if $f$ is functionally equivalent to $g$.

Proof. It is clear that $f = g$ implies $f \sim g$, since this is true for all tropical polynomials. Now suppose that $f \neq g$. Then for some term $a_i x^i$ of $f(x)$ and the corresponding term $b_i x^i$ of $g(x)$, we have $a_i \neq b_i$. Without loss of generality, suppose $a_i < b_i$. Since $g$ is a least-coefficient polynomial, $g(x_0) = b_i x_0^i$ for some $x_0$, by Lemma 1. Now,

$$f(x_0) \leq a_i x_0^i < b_i x_0^i = g(x_0),$$

so $f$ is not functionally equivalent to $g$. \qed

The least-coefficient representative is often the most useful way to represent a functional equivalence class of tropical polynomials.

Lemma 3. Let $f(x) = a_n x^n \oplus a_{n-1} x^{n-1} \oplus \cdots \oplus a_r x^r$. There is a unique least-coefficient polynomial $g(x) = b_n x^n \oplus b_{n-1} x^{n-1} \oplus \cdots \oplus b_r x^r$ such that $f \sim g$. Furthermore, each coefficient $b_j$ of $g(x)$ is given by

$$b_j = \min \left( \left\{ a_j \right\} \cup \left\{ \frac{a_i \cdot (k - j) + a_k \cdot (j - i)}{k - i} \mid r \leq i < j < k \leq n \right\} \right).$$

(2.1)

Proof. First we will show that $f \sim g$. Given $x_0$, note that $f(x_0) = a_s x_0^s = a_s + s \cdot x_0$ for some $s$. Also,

$$g(x_0) = \min_{r \leq j \leq n} \left\{ b_j + j \cdot x_0 \right\}$$

$$= \min_{r \leq i < j \leq n} \left\{ a_j + j \cdot x_0, \frac{a_i \cdot (k - j) + a_k \cdot (j - i)}{k - i} + j \cdot x_0 \right\}.$$  (2.2)
So for any $i, j,$ and $k$ such that $r \leq i < j < k \leq n$, if $x_0 \geq \frac{a_i - a_k}{k - i}$ then

\[
a_s + s \cdot x_0 \leq a_i + i \cdot x_0
= \frac{a_i \cdot (k - j) + a_k \cdot (j - i)}{k - i} + (j - i) \cdot \left(\frac{a_i - a_k}{k - i}\right) + i \cdot x_0
\leq \frac{a_i \cdot (k - j) + a_k \cdot (j - i)}{k - i} + (j - i) \cdot x_0 + i \cdot x_0
= \frac{a_i \cdot (k - j) + a_k \cdot (j - i)}{k - i} + j \cdot x_0.
\]

A similar argument shows that if $x_0 \leq \frac{a_i - a_k}{k - i}$, then

\[
a_s + s \cdot x_0 \leq a_k + k \cdot x_0 \leq \frac{a_i \cdot (k - j) + a_k \cdot (j - i)}{k - i} + j \cdot x_0.
\]

Since this is true for all $i, j,$ and $k$, the equation in (2.2) evaluates to $g(x_0) = a_s x_0^s$, so $g(x_0) = f(x_0)$ and $f \sim g$, as desired.

Secondly, we must show $g$ is a least-coefficient polynomial. Given a coefficient $b_j$ in $g$, suppose that $a_j$ is a least coefficient of $f$. From Equation (2.1) we see that $b_j \leq a_j$. Since $a_j$ is a least coefficient, there is some $x_0$ such that $f(x_0) = a_j x_0^j$, so $b_j x_0^j \geq g(x_0) = f(x_0) = a_j x_0^j$. Therefore $b_j = a_j$ and $g(x_0) = b_j x_0^j$.

Now suppose that $a_j$ is not a least coefficient. Then since $a_r$ and $a_n$ are least-coefficient, we can choose $u < j$ and $v > j$ such that $a_u$ and $a_v$ are least coefficients and for any $t$ such that $u < t < v$, $a_t$ is not a least coefficient.

Let $x_0 = \frac{a_u - a_v}{v - u}$ and suppose, by way of contradiction, that $f(x_0) \neq a_u x_0^u$. Then $f(x_0) = a_w x_0^w < a_u x_0^u$ for some $w$. Note that $a_w$ is a least coefficient, so it cannot be that $u < w < v$ by our assumption on $u$ and $v$. If $w < u$ then for $x \geq x_0$,

\[
a_w + w \cdot x = a_w + u \cdot x - (u - w) \cdot x
\leq a_w + u \cdot x - (u - w) \cdot x_0
= a_w + w \cdot x_0 + u \cdot (x - x_0)
< a_u + u \cdot x_0 + u \cdot (x - x_0)
= a_u + u \cdot x.
\]
For $x < x_0$,

$$a_v + v \cdot x = a_v + u \cdot x + (v - u) \cdot x$$

$$< a_v + u \cdot x + (v - u) \cdot x_0$$

$$= a_v + v \cdot x_0 + u \cdot (x - x_0)$$

$$= a_u + u \cdot x_0 + u \cdot (x - x_0)$$

$$= a_u + u \cdot x$$

So there is no $x$ such that $f(x) = a_u x^u$ and thus $a_u$ is not a least coefficient, which contradicts our assumption. If $w > v$, a similar argument shows that $a_v$ is not a least coefficient, again contradicting our assumption. Therefore,

$$f(x_0) = a_u + u \cdot \left(\frac{a_u - a_v}{v - u}\right)$$

$$= \frac{a_u \cdot (v - j) + a_v \cdot (j - u)}{v - u} + j \cdot \left(\frac{a_u - a_v}{v - u}\right)$$

$$= c + j \cdot x_0$$

where $c = \frac{a_u \cdot (v - j) + a_v \cdot (j - u)}{v - u}$. (2.3)

Again, from (2.1) we see that $b_j \leq c$, and from (2.3) we see $cx_0^j = f(x_0) = g(x_0) \leq b_j x_0^j$. So $b_j = c$ and $g(x_0) = b_j x_0^j$.

Finally, $g$ is the only such polynomial by Lemma 2.

**Note.** The use of a least-coefficient polynomial as a best representative for a functional equivalence class is one of the key ideas of this chapter. We cannot develop well-defined algebraic transformations of tropical polynomials without unique representatives for functional equivalence classes. While Izhakian discusses in [7] what he calls an “effective” coefficient (similar to a least coefficient), the idea of using least-coefficient polynomials to represent functional equivalence classes has not been discussed.

When dealing with polynomials of a single variable, we have a simple way to determine whether or not a polynomial is a least coefficient:
Lemma 4. Let \( f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_r x^r \), where each \( a_i \) is not infinity. Let \( d_i = a_{i-1} - a_i \) be the difference between two consecutive coefficients. Then \( f(x) \) is a least-coefficient polynomial if and only if the difference between consecutive coefficients is non-decreasing, that is, if \( d_n \leq d_{n-1} \leq \cdots \leq d_{r+1} \).

Proof. Suppose that \( f \) has a set of consecutive coefficients whose differences are decreasing, that is, \( ax^{i+1}, bx^i, \) and \( cx^{i-1} \) are consecutive terms of \( f(x) \) such that \( b - a > c - b \). Then \( b > \frac{1}{2} \cdot (a + c) \). We will show that \( f(x_0) < bx_0^i \) for all \( x_0 \), meaning that \( b \) is not a least coefficient.

Given \( x_0 \), if \( x_0 \leq \frac{1}{2} \cdot (c - a) \) then

\[
ax_0^{i+1} = (i + 1) \cdot x_0 + a \\
\leq i \cdot x_0 + \frac{1}{2} \cdot (c - a) + a \\
= i \cdot x_0 + \frac{1}{2} \cdot (c + a) \\
< i \cdot x_0 + b = bx_0^i,
\]

so \( f(x_0) \leq ax_0^{i+1} < bx_0^i \). Similarly, if \( x_0 \geq \frac{1}{2} \cdot (c - a) \),

\[
ax_0^{i-1} = (i - 1) \cdot x_0 + c \\
\leq i \cdot x_0 - \frac{1}{2} \cdot (c - a) + c \\
= i \cdot x_0 + \frac{1}{2} \cdot (c + a) \\
< i \cdot x_0 + b = bx_0^i,
\]

so \( f(x_0) \leq cx_0^{i-1} < bx_0^i \). Therefore \( b \) is not a least coefficient, and \( f \) is not a least-coefficient polynomial.

For the other direction, suppose that the differences between the coefficients of \( f(x) \) are non-decreasing. Since \( a_n, a_r \neq \infty \), \( a_n \) and \( a_r \) are least coefficients. Let \( a_i \) be a coefficient of \( f \), with \( r < i < n \), and let \( x_0 = \frac{a_{i-1} - a_{i+1}}{2} \). We will show that \( f(x_0) = a_i x_0^i \), so \( a_i \) is a least coefficient. We must show for all \( k \) that \( i \cdot x_0 + a_i \leq k \cdot x_0 + a_k \).
This is certainly true for \( i = k \). Suppose \( k > i \). Since \((a_t - a_{t+1}) \leq (a_s - a_{s+1})\) for \( t \geq s \), we have

\[
(a_i - a_{i+2}) = (a_i - a_{i+1}) + (a_{i+1} - a_{i+2}) \leq 2 \cdot (a_i - a_{i+1})
\]

\[
(a_i - a_{i+3}) = (a_i - a_{i+2}) + (a_{i+2} - a_{i+3}) \leq 3 \cdot (a_i - a_{i+1})
\]

And in general we get

\[
(a_i - a_k) \leq (a_i - a_{i+1}) \cdot (k - i) = \frac{1}{2} \cdot (2 \cdot (a_i - a_{i+1})) \cdot (k - i)
\]

\[
\leq \frac{1}{2} \cdot ((a_{i-1} - a_i) + (a_i - a_{i+1})) \cdot (k - i) = x_0 \cdot (k - i).
\]

Thus, \( i \cdot x_0 + a_i \leq k \cdot x_0 + a_k \). A similar argument holds for \( k < i \). So (tropically) \( a_i x_0^i \leq a_s x_0^n \) for all \( s \). This means that \( f(x_0) = a_i x_0^i \), so \( a_i \) is a least coefficient. Therefore, \( f \) is a least-coefficient polynomial. \( \square \)

**Note.** If \( f(x) \) has a coefficient \( a_i \) such that \( a_i = \infty \) for \( r < i < n \), then \( f \) is not a least-coefficient polynomial; but of course, \( a_j = \infty \) for all \( j > n \) and all \( j < r \), even in a least-coefficient polynomial.

### 2.3 The Fundamental Theorem of Tropical Algebra

We are now ready to state and prove the Fundamental Theorem of Tropical Algebra.

**Theorem 5** (Fundamental Theorem). Let \( f(x) = a_n x^n \oplus a_{n-1} x^{n-1} \oplus \cdots \oplus a_r x^r \) be a least-coefficient polynomial. Then \( f(x) \) can be written uniquely as the product of linear factors

\[
a_n x^r (x \oplus d_n) (x \oplus d_{n-1}) \cdots (x \oplus d_{r+1}) ,
\]
where \( d_i = a_{i-1} - a_i \). In other words, the roots of \( f(x) \) are the differences between consecutive coefficients. Since every polynomial is equivalent to a least-coefficient polynomial, this proves that every tropical polynomial can be factored into linear terms.

**Proof.** Since \( f(x) \) is a least-coefficient polynomial, the differences between consecutive coefficients is non-decreasing, i.e., \( d_n \leq d_{n-1} \leq \cdots \leq d_{r+1} \). Knowing these inequalities, we can expand (2.4) to get

\[
a_n x^n \oplus a_n d_n x^{n-1} \oplus a_n d_n d_{n-1} x^{n-2} \oplus \cdots \oplus a_n d_n d_{n-1} \cdots d_{r+1}.
\]  

But the coefficient of the \( x^i \) term in this polynomial is

\[a_n d_n d_{n-1} \cdots d_{i+1} = a_n + d_n + d_{n-1} + \cdots + d_{i+1} \cdot\]

A straightforward computation shows that this is equal to \( a_i \), so the polynomial in (2.5) is equal to \( f(x) \), as desired.

Now suppose that there is another way of writing \( f(x) \) as a product of linear factors. Call this product \( g \) and note that it must have the same degree as \( f \). Additionally, the smallest non-infinite term of \( g \) must have the same degree as the smallest non-infinite term of \( f \). Hence, we are able to write

\[g(x) = a'_n x^r (x \oplus d'_n) (x \oplus d'_{n-1}) \cdots (x \oplus d'_{r+1}),\]

with each \( d'_i \) chosen, after reindexing, if necessary, such that \( d'_n \leq d'_{n-1} \leq \cdots \leq d'_{r+1} \).

Expanding this product shows that the differences between consecutive coefficients of \( g \) are non-decreasing, so \( g \) is a least-coefficient polynomial by Lemma 4. Also, it is clear that \( f \neq g \), so by Lemma 2, \( f \) is not functionally equivalent to \( g \). Therefore, the factorization is unique.

**Factoring: an example.** To better understand how factoring one-variable polynomials works, consider the polynomial \( f(x) = x^3 \oplus 3x^2 \oplus 2x \oplus 4 \). First we use
Equation [2.1] to find a least coefficient polynomial that is functionally equivalent to $f$. This polynomial is $g(x) = x^3 + 1x^2 + 2x + 4$. Then the “roots” of $f(x)$ are the differences between consecutive coefficients of $g(x)$: 1, 1, 2. So $f(x) \sim (x \oplus 1)^2(x \oplus 2)$. Note that these “roots” correspond to the corners of the function graph of $f(x)$ (see Figure 2.3).

![Figure 2.3](image)

$f(x) = x^3 \oplus 3x^2 \oplus 2x \oplus 4$

Figure 2.2: The function graph of $f(x)$. Note that $f(x)$ has “roots” at 1, 1, 2, corresponding to the corners of the graph.

### 2.4 The Corner Locus of a Polynomial

Finally we note that tropical factoring gives us a slightly different result than classical factoring. Classically, the set of roots (or zero locus) of a polynomial is the set of points at which the polynomial evaluates to the additive identity. Unfortunately, tropical polynomials have either no roots or trivial roots in this sense. In fact, if $f(x) \neq \infty$, then $f(x_0)$ never evaluates to the additive identity $\infty$ when $x_0 \neq \infty$. However, as we have seen, polynomials in $\mathbb{Q}[x]$ can be factored and seem to have “roots,” although they do not evaluate to the additive identity at these points. Clearly, we must use a different, more meaningful definition. In [12] motivation is given for the following definition of zero locus.
2.4 The Corner Locus of a Polynomial

Definition 4. Let \( f(x) \in \mathbb{Q}[x] \). The zero locus (or corner locus) \( \mathcal{Z}(f) \) is defined to be the set of points \( x_0 \) in \( \mathbb{Q} \) for which at least two monomials of \( f \) attain the minimum value.

The \( d_i \) in (2.4) are precisely that points of \( \mathcal{Z}(f) \), as we now show.

Theorem 6. Given a point \( d \in \mathbb{Q} \) and a least-coefficient polynomial \( f(x) \), \( x \oplus d \) is a factor of \( f(x) \) if and only if \( f(d) \) attains its minimum on at least two monomials.

Proof. First, suppose that \( x \oplus d \) is a factor of \( f(x) \). If we write \( f \) as a product of linear factors as in (2.4), \( d_i = d \) for some \( d_i \). For all \( j < i \),

\[
a_i + i \cdot d_i = a_i + (i - j) \cdot d_i + j \cdot d_i \\
\leq a_i + d_i + d_{i-1} + \cdots + d_{j+1} + j \cdot d_i \\
= a_j + j \cdot d_i.
\]

A similar calculation shows that for \( j > i \), we have \( a_i + i \cdot d_i \leq a_j + j \cdot d_i \). So \( f(d_i) = a_i d_i^i \). Also, \( a_i d_i^i = a_n d_n d_{n-1} \cdots d_{i+1} d_i^i = a_n d_n d_{n-1} \cdots d_i d_{i-1} = a_{i-1} d_i^{i-1} \), so the minimum is attained by at least two monomials of \( f(x) \) at \( x = d \).

For the other direction, suppose that the minimum is attained by two monomials at \( f(d) \). By way of contradiction, suppose that these monomials are not consecutive. Then for some \( j < i < k \), we have \( a_j d^j = a_k d^k < a_i d^i \). If \( x \leq d \) then

\[
a_k + k \cdot x = a_k + i \cdot x + (k - i) \cdot x \\
\leq a_k + i \cdot x + (k - i) \cdot d \\
= a_k + k \cdot d + i \cdot (x - d) \\
< a_i + i \cdot d + i \cdot (x - d) \\
= a_i + i \cdot x.
\]
Similarly, if $x \geq d$, then $a_j x^j < a_i x^i$. Thus there is no $x$ such that $f(x) = a_i x^i$, so $a_i$ is not a least coefficient, which is a contradiction. Therefore there is some $i$ such that $a_i d^j = a_{i-1} d^{j-1}$. Thus we have

$$0 = a_{i-1} + (i - 1) \cdot d - (a_i + i \cdot d) = a_{i-1} - a_i - d.$$ 

So $d = a_{i-1} - a_i$, the difference between two consecutive coefficients. Since $f$ is a least-coefficient polynomial, $x \oplus d$ is a factor of $f$ by the Fundamental Theorem.

Thus, as in the classical case, the unique factorization of a polynomial in $\mathbb{Q}[x]$ gives us what could be considered the roots of the polynomial. It is clear that all of the arguments and results of this chapter hold if we replace the rationals $\mathbb{Q}$ with any classical ordered field. Thus any ordered field, together with $\infty$, can be said to be tropically algebraically closed.
Chapter 3

Tropical Polynomials of Several Variables

Although we can factor every single-variable polynomial into linear factors, we have no hope of being able to do the same for polynomials of two or more variables. In fact, tropical polynomials in more than one variable do not always have unique factorizations. Nevertheless, the structure of the tropical zero locus makes it possible for us to learn a lot more about tropical factorization than we would expect to be able to. To make use of this structure, we discuss the tropical zero locus in detail. We discuss polyhedral complexes and a correspondence between weighted balanced complexes and tropical polynomials. We prove that factorization of tropical polynomials in two or more variables is NP-complete. We then present a method in which we use the structure of the tropical zero locus to find factorizations of polynomials.

Notation. In Chapter 2, we were very careful to distinguish between formal equality and functional equivalence. In this chapter, we will not be so careful. We will now consider two polynomials the same polynomial if they define the same function. In

$1$For example, the cubic polynomial $x^2y \oplus xy^2 \oplus x^2 \oplus xy \oplus y^2 \oplus x \oplus y$ factors into irreducible polynomials as $(x \oplus 0)(y \oplus 0)(x \oplus y)$ or as $(x \oplus y \oplus 0)(xy \oplus x \oplus y)$. 

17
other words, we will write $f = g$ to mean that $f$ and $g$ are functionally equivalent.

We will also use multi-index notation throughout this chapter. In other words, we will write a polynomial $f \in \mathbb{Q}[x_1, \ldots, x_n]$ as

$$f(x) = \bigoplus_{I \in \Delta} a_I x^I,$$

where $I = (I_1, \ldots, I_n) \in (\mathbb{Z}^+)^n$, and $x^i = x_1^{I_1} x_2^{I_2} \cdots x_n^{I_n}$. The set $\Delta$ is called the support of $f$. It is important to note that a tropical monomial $a_I x^I$ is written classically as $\langle x, I \rangle + a_I$, where $\langle \cdot, \cdot \rangle$ is the scalar product and we think of $x$ as $(x_1, \ldots, x_n)$. Therefore, we have

$$f(x_1, \ldots, x_n) = f(x) = \bigoplus_{I \in \Delta} a_I x^I = \min_{I \in \Delta} \{\langle x, I \rangle + a_I \}.$$

It is also important to note that since our base field is $\mathbb{Q} = \mathbb{Q} \cup \{\infty\}$, our calculations will sometimes include multiplication by or subtraction of infinity. We will use the convention that $\infty \cdot 0 = 0$ and $\infty - \infty = 0$.

### 3.1 The Corner Locus and Weighted Balanced Complex

#### 3.1.1 The Corner Locus

**Definition 5.** If $f \in \mathbb{Q}[x_1, \ldots, x_n]$ is a polynomial then the tropical zero locus or corner locus of $f$ is the set

$$\mathcal{Z}(f) := \{\alpha \in \mathbb{Q}^n : \text{two or more monomials of } f \text{ attain the minimum at } \alpha\}.$$

Even though it is not technically a zero locus, we call $\mathcal{Z}(f)$ the tropical zero locus because it is analogous to the classical zero locus. We often refer to $\mathcal{Z}(f)$ as the corner
3.1 The Corner Locus and Weighted Balanced Complex

The corner locus because it is in fact the projection of corners of the surface \( z = f(x_1, \ldots, x_n) \) into the hyperplane \( z = 0 \) (see Figure 3.1).

Let \( f(x_1, \ldots, x_n) \) be a polynomial, and let \( a_I x^I \) and \( a_J x^J \) be two monomials of \( f \). Then if the two monomials attain the minimum together at some \( y \in \mathbb{Q}^n \), they are equal, and thus satisfy the equation

\[
\langle y, I - J \rangle = a_J - a_I.
\]

In other words, the set of all points where \( a_I x^I \) and \( a_J x^J \) simultaneously attain the minimum of \( f \) is contained in a hyperplane that is perpendicular to the integer vector \( I - J \).

Also note that \( \mathcal{Z}(f) \) partitions \( \mathbb{Q}^n \) into connected regions, and on each of these connected regions, one of the monomials \( a_I x^I \) is strictly less than all of the others.

3.1.2 The Weighted Balanced Complex

Corner loci of tropical polynomials have several nice properties. They can be written as the union of a finite number of super-convex polyhedra (that is, convex polyhedra

\[
f(x, y) = x^2 \oplus xy \oplus 2y^2 \oplus 1x \oplus 3
\]

Figure 3.1: Left: the surface \( z = f(x, y) \). Right: the tropical zero locus of \( f \).
whose faces of every dimension are also convex) that intersect each other only along entire faces. The polyhedra can also be assigned weights and are balanced in a sense that we will discuss later. We formalize all of this in the definition of a weighted balanced complex. We also give the definition of a weighted balanced complex. Our definitions are based on Mikhalikin’s definitions, given in [9].

**Definition 6.** A subset \( \Pi \subset \mathbb{Q}^{n+1} \) is called a *rational polyhedral complex* (herein a *complex*) if it can be presented as a finite union of closed sets in \( \mathbb{Q}^{n+1} \) called *cells* with the following properties.

- Each cell is a closed convex (possibly semi-infinite or infinite) polyhedron. We call a cell of dimension \( k \) a *k-cell*.

- Each cell has rational slope (i.e. its affine span is the kernel of a rational matrix).

- The boundary of a *k-cell* (that is, the boundary in the corresponding affine span) is a union of \((k - 1)\)-cells.

- Different open cells (that is, the interiors of the cells in the corresponding affine spans) do not intersect.

For simplicity, we require all cells to be maximal in the sense that the union of two distinct \( k \)-cells is not a \( k \)-cell. The dimension of a complex is the maximal dimension of its cells. We call a complex of dimension \( n \) an *n-complex*.

**Definition 7.** A complex \( \Pi \subset \mathbb{Q}^{n+1} \) is called *weighted* if there is a natural number assigned to each \( n \)-cell of \( \Pi \).

We are now ready to discuss what it means for a weighted complex to be balanced. First, it is important to distinguish between boundary cells and interior cells.
Definition 8. In a complex $\Pi \subset Q^{n+1}$, a boundary cell is a cell $A$ that is contained in a boundary hyperplane of $Q^{n+1}$. In other words $A \subset \{ (x_1, \ldots, x_n) : x_i = \infty \}$ for some $i$. An interior cell is a cell not contained in any boundary hyperplane.

Definition 9. In a complex $\Pi \subset Q^{n+1}$, the integer covector of an interior $n$-cell $F$ is a vector $c \in \mathbb{Z}^n$ such that $c$ is normal to $F$, with the entries of $c$ relatively prime. Note an integer covector is only well-defined up to its sign. The covector becomes well defined when we associate it with one of the regions of $Q^{n+1}$ on either side of it. The integer covector of an $n$-cell with respect to a region is the integer covector of the $n$-cell that points from the $n$-cell into the region.

Definition 10 (Balancing Condition). Let $\Pi \subset Q^{n+1}$ be a weighted $n$-complex with the property that for $k < n$, every $(k - 1)$-cell is contained in the boundary of some $k$-cell. Let $r$ be an interior $(n - 1)$-cell of $\Pi$. Let $\{F_1, \ldots, F_k\}$ be the set of $n$-cells in $\Pi$ containing $r$ in their boundary. Then $\Pi$ is balanced at $r$ if for the weights $\omega_1, \ldots, \omega_k$ corresponding to the $F_i$, we have

$$\sum_{i=1}^{k} \omega_ic_i = 0,$$

where each $c_i$ is an integer covector corresponding to $F_i$ and no two $c_i$ correspond to the same region of the complement of $\Pi$.

We say that $\Pi$ is balanced at the boundary if each boundary hyperplane either contains no $n$-cells or is the union of a set of $n$-cells, all of which have the same weight.

We say that $\Pi$ is balanced (i.e. $\Pi$ is a weighted balanced complex) if $\Pi$ is balanced at each of its interior $(n - 1)$-cells and balanced at the boundary.

Note. Much work has been done regarding so-called tropical graphs and tropical weighted balanced graphs that correspond to tropical curves (see [5, 13]). In $Q^2$, a
3.1 The Corner Locus and Weighted Balanced Complex

Figure 3.2: Corner locus of $x \oplus y \oplus z \oplus 0$: A weighted balanced 2-complex. Note that the 2-cells are balanced around the 1-cells

weighted balanced 1-complex is a weighted balanced graph. The 1-cells of a weighted balanced 1-complex correspond to the edges of a weighted balanced graph, and the 0-cells correspond to vertices. In this paper, we do not consider tropical graphs in $\mathbb{Q}^n$ for $n > 2$, since they do not correspond to the corner locus of a single polynomial. For the same reason, we do not consider polyhedral complexes of codimension greater than one.

Also, note that the balancing condition (together with our requirement that cells of a complex be maximal) implies that each interior $(n-1)$-cell is in fact the boundary of three or more $n$-cells.

3.1.3 Polynomial-Complex Correspondence

Theorem 7. There is a bijective correspondence between weighted balanced complexes and tropical polynomials, up to functional equivalence and multiplication of the polynomial by a scalar.
This theorem is proved by Mikhalkin in [9], although he does not use the language of tropical polynomials. We should also note that Mikhalkin only proves the correspondence up to multiplication by an indeterminant. It is easy to see that by using $\mathbb{Q}^n$, we do not have this ambiguity. We also note that Mikhalkin proves the theorem for $\mathbb{R}^n$, while we only use $\mathbb{Q}^n$. Since $\mathbb{Q}$ is algebraically closed, $\mathbb{Q}^n$ is sufficient.

Using this theorem, we can prove things about polynomials by proving things about their corresponding weighted balanced complexes. For convenience, we will give the correspondence explicitly.

**The Weighted Balanced Complex Corresponding to a Polynomial.** Let $f$ be a tropical polynomial. Mikhalkin has shown that $\mathcal{Z}(f)$ is a complex. We make $\mathcal{Z}(f)$ into a weighted complex by assigning each $n$-cell $F$ a weight as follows:

- If $F$ a boundary $n$-cell, contained in the hyperplane corresponding to $x_i = \infty$, then $x_i$ divides $f$. Let $m$ be the largest power of $x_i$ dividing $f$. The we assign the weight $m$ to $F$.

- If $F$ is an interior $n$-cell, then $F$ is the boundary between two $(n+1)$-dimensional regions in $\mathbb{Q}^{n+1}$. If $a_I x^I$ and $a_J x^J$ are the monomials corresponding to these regions, then we define the weight of $F$ to be the lattice length of the vector $I - J$.

**Definition 11.** The weighted balanced complex formed above is called the weighted balanced complex of $f$ and is denoted $\mathcal{W}(f)$.

**The Polynomial Corresponding to a Weighted Balanced Graph** Given a weighted balanced complex $\Pi \subset \mathbb{Q}^{n+1}$, we construct a polynomial as follows.

\[ \text{This weight is easily calculated to be the GCD of the entries of } I - J. \]
1. Choose a region $R_0$ of the complement of $\Pi$. Some monomial attains the minimum on this region. Call it $a_Ix^I$, where $a_I$ is arbitrary.

2. For every other region $R$ of the complement of $\Pi$, choose a sequence of regions $R_0, R_1, \ldots, R_{k-1}, R_k = R$ such that each $R_{i-1}$ is adjacent to the region $R_i$. For $1 \leq i \leq k$, let $F_i$ be the $n$-cell that separates $R_{i-1}$ from $R_i$. Let $\omega_i$ be the weight of $F_i$, and let $c_i$ be the integer covector corresponding to $F_i$ and $R_i$. Assign the monomial $a_Jx^J$ to $R$, where

$$J = I - \sum_{i=1}^{k} \omega_ic_i.$$

3. Adjust the entries of $I$ to be all positive, but as small as possible so that $x_i^m$ divides $f$ but $x_i^{m+1}$ does not, where $m$ is the weight of the $n$-cells in the $x_i$ boundary, if there are any, and 0 otherwise.

4. Assign coefficients using the $(n-1)$-cells.

### 3.2 Factors and Balanced Subcomplexes

**Lemma 8.** Let $a, b, c, d \in \mathbb{Z}^2$ with $b - a$ and $d - c$ multiples of the same primitive integral vector $e$. Then the lattice length from $a + c$ to $b + d$ is the sum of the lattice length from $a$ to $b$ and the lattice length from $c$ to $d$.

*Proof.* Let $b - a = \omega_1e$ and $d - c = \omega_2e$. Then $\omega_1$ and $\omega_2$ are the lattice lengths from $a$ to $b$ and $c$ to $d$ respectively. It is clear that $(b + d) - (a + c) = (b - a) + (d - c) = \omega_1e + \omega_2e = (\omega_1 + \omega_2)e$. So the lattice length from $a + c$ to $b + d$ is $\omega_1 + \omega_2$, as desired. \qed

**Lemma 9** (Aaron Hill). *If $f$ and $g$ are tropical polynomials, then the corner locus of $fg$ is equal to the union of the corner locus of $f$ and the corner locus of $g$. In other words, $\mathcal{Z}(fg) = \mathcal{Z}(f) \cup \mathcal{Z}(g)$.***
3.2 Factors and Balanced Subcomplexes

Proof. The proof is straightforward, and is given in [3].

Lemma 10. Let \( f, g \in \mathbb{Q}[x_1, \ldots, x_{n+1}] \), and let \( F \) be an \( n \)-cell of \( \mathcal{W}(fg) \subset \mathbb{Q}^{n+1} \). Let \( \omega_f \) be the weight of the \( n \)-cell of \( \mathcal{W}(f) \) that contains \( F \), if such an \( n \)-cell exists, or 0 if no such \( n \)-cell exists. Define \( \omega_g \) the same way. Then the weight of \( F \) is equal to \( \omega_f + \omega_g \).

Proof. If \( F \) is a boundary \( n \)-cell, then this follows directly from Definition [11]. So we may assume that \( F \) is an interior \( n \)-cell.

We will now prove that \( F \) is either contained in an \( n \)-cell of \( \mathcal{W}(f) \) or the interior of \( F \) is disjoint from \( \mathcal{W}(f) \). Note that every point of a weighted balanced complex is either in the interior of an \( n \)-cell or in the boundary of an \( n \)-cell. Small enough neighborhoods of an interior point of an \( n \)-cell are contained in a hyperplane, while neighborhoods of points on \( n \)-cell boundaries are not. Suppose that there is a point \( \alpha \in F^\circ \cap \mathcal{W}(f) \). Since \( \alpha \) is in the interior of \( F \), it must be in the interior of some \( n \)-cell \( G \) of \( \mathcal{W}(f) \). Let \( \beta \) be any other point of the interior of \( F \). Since \( F \) is convex, every point on the line segment from \( \alpha \) to \( \beta \) is in the interior of \( F \), and therefore not in the boundary of \( G \). So in fact, \( \beta \) must be in the interior of \( G \), and we have \( F \subset G \).

If \( F \) is contained in an \( n \)-cell of \( \mathcal{W}(f) \), then there is one monomial of \( f \) for each side of \( F \) that attains the minimum of \( f \) in that region. If the interior of \( F \) is disjoint from \( \mathcal{W}(f) \), then the same monomial attains the minimum on each side of \( F \). The same is of course true for \( \mathcal{W}(g) \).

Let \( a \) and \( b \) correspond to the exponents of the monomials of \( f \) that attain the minimum on either side of \( A \), and let \( c \) and \( d \) correspond to the exponents of the monomials of \( g \) that attain the minimum on either side of \( A \) (It is possible that \( c = d \)). Now \( b - a = \omega_f e \) for some primitive integer vector \( e \). It is clear that \( d - c = \omega_g e \) for the same vector \( e \). So by Lemma [8] the weight of \( F \) is the lattice length from \( a + c \).
3.2 Factors and Balanced Subcomplexes

to $b + d$ which is simply the lattice length from $a$ to $b$ plus the lattice length from $c$ to $d$, which is $\omega_f + \omega_g$, as desired.

Figure 3.3: Weighted balanced graphs of $f$, $g$, and $fg$. Note that when a segment overlaps in the product, the weights add (see Lemma 10).

**Definition 12.** If $G$ is a weighted complex, then a subset $H$ of $G$ is a subcomplex of $G$ if $H$ is a complex, and each $n$-cell $F$ of $H$ has weight less than or equal to the weight of the $n$-cell of $G$ that contains $F$.

**Lemma 11.** If $f$ is a factor of $g$ then $W(f)$ is a balanced subcomplex of $W(g)$.

*Proof.* By definition, $W(f)$ is a weighted balanced complex. Lemma 10 guarantees that $W(f)$ is a subcomplex of $W(g)$. □

**Theorem 12.** If $G$ is a subcomplex of $W(f) \subset Q^{n+1}$ that satisfies the balancing condition, then the polynomial corresponding to $G$ is a factor of $f$.

*Proof.* Consider $W(f)$ as a collection of $n$-cells $\{F_1, \ldots, F_k\}$, with each $F_i$ assigned a weight $\omega_i$. We can think of $G$ as the same collection of $n$-cells, with each $F_i$ assigned a weight $\sigma_i$, with $0 \leq \sigma_i \leq \omega_i$. Note that we assign an $n$-cell $F_i$ a weight of $\sigma_i = 0$ in
3.3 A Factoring Algorithm

$G$ if $F_i$ is not in $G$, otherwise $\sigma_i$ corresponds to the weight of $F_i$ in $G$. Also note that while $G$ may not be maximal in the sense that we required in Definition 6, this is of little importance. Let $H$ be the weighted complex given by $\{F_1, \ldots, F_k\}$ with each $F_i$ assigned a weight $\omega_i - \sigma_i$.

We first prove that $H$ satisfies the balancing condition. Let $r$ be an interior $(n-1)$-cell of $H$ and let $\{F_{n_1}, \ldots, F_{n_m}\}$ be the set of $n$-cells adjacent to $r$. Let $\{c_{n_1}, \ldots c_{n_m}\}$ be the integer covectors of the $F_{n_i}$, no two with respect to the same region of the complement of $H$. Then we have

$$\sum_{i=1}^{m} (\omega_{n_i} - \sigma_{n_i})c_{n_i} = \sum_{i=1}^{m} \omega_{n_i}c_{n_i} - \sum_{i=1}^{m} \sigma_{n_i}c_{n_i} = 0 - 0 = 0.$$

Thus, $H$ is balanced at all of its interior $(n-1)$-cells. It is clear that $H$ is also balanced at the boundary.

Let $f_1$ be the polynomial corresponding to $G$ and $f_2$ the polynomial corresponding to $H$. By Lemma 10, $W(f_1f_2) = W(f)$. So in fact $cf_1f_2 = f$ for some constant $c$. Therefore, the polynomial $f_1$ corresponding to $G$ is a factor of $f$. \qed

3.3 A Factoring Algorithm

3.3.1 NP-completeness

**Theorem 13.** The problem of factoring tropical polynomials in two or more variables is NP-complete.

---

3 NP stands for nondeterministic polynomial time, and we say that a problem is NP if it is possible to verify a solution in polynomial time (though it may not be possible to find a solution to such a problem in polynomial time). To say that factoring is NP-complete means that if we could find factors in polynomial time (as opposed to just being able to verify them), then we could solve every NP problem in polynomial time. In other words, this theorem asserts that it is extremely unlikely that a polynomial-time factoring algorithm exists. For more information, see [http://mathworld.wolfram.com/NP-CompleteProblem.html](http://mathworld.wolfram.com/NP-CompleteProblem.html)
3.3 A Factoring Algorithm

Proof. It is clear that the problem of factoring tropical polynomials is NP, since the weighted balanced complex of a polynomial can clearly be found in polynomial time, and verifying that one polynomial is a factor of another is a matter of checking that one weighted balanced complex is a subcomplex of the other.

In [4], Gao and Lauder show that the problem of decomposing Newton polyhedra is NP-complete. We prove that any algorithm that would factor tropical polynomials would decompose Newton polyhedra as well.

Let $P$ be a two-dimensional Newton polyhedron, let $v_0$ be a vertex of $P$ and for $1 \leq i \leq k$, let $e_i$ be a primitive integer vector and $\omega_i$ a weight such that for $1 \leq j \leq k$, $v_k = v_0 + \sum_{i=1}^{j} \omega_i e_i$. Since $P$ is a polygon, $\sum_{i=1}^{k} \omega_i e_i = 0$. So there is a weighted balanced 1-complex with $k$ edges (1-cells), each of which has weight $\omega_i$ and is perpendicular to $e_i$. If we factor the polynomial corresponding to this complex then we find a subcomplex, which corresponds to a summand of $P$ by Lemma 13 of [4] (see Figure 3.4). Since decomposing Newton polyhedra is NP-complete, so is factoring tropical polynomials.

![Figure 3.4](image.png)

Figure 3.4: If there were a polynomial-time algorithm to factor tropical polynomials, then the same algorithm would decompose Newton polyhedra. Since decomposition of Newton polyhedra is NP-complete, so is tropical factoring.

3.3.2 A factoring algorithm

Let $f \in \mathbb{Q}[x_1, \ldots, x_{n+1}]$ be a tropical polynomial. Let $G = W(f) \subset \mathbb{Q}^{n+1}$. 
1. Choose an $n$-cell $F$ of $G$. Let $S = \{F\}$.

2. For each element $F_i$ of $S$, analyze the $(n-1)$-cells in the boundary of $F_i$ to determine if and how each $(n-1)$-cell can be decomposed. If necessary, choose one of the decompositions. Add all the $n$-cells of the decomposition to $S$.

3. Repeat Step 2 until no new $n$-cells are added to $S$.

When the algorithm is finished, $S$ will be a balanced subcomplex of $G$. So by Theorem 12, the polynomial corresponding to $S$ is a factor of $f$. It is possible, however, that $S = G$, in which case we have found a trivial factor. Even if $S \neq G$, there may be multiple (distinct) factorizations of $f$. It is necessary to repeat the whole process, using all possible combinations of choices in Step 2 to determine all possible factorizations of $f$ or to determine that $f$ is irreducible.

**Example** To see how the algorithm works, we will look at an example (see Figure 3.5). Let $f = x^3 \oplus x^2y \oplus 1xy^2 \oplus 1y^3 \oplus x^2 \oplus xy \oplus y^2 \oplus x \oplus y \oplus 2$. The weighted balanced graph $G$ of $f$ consists of 8 segments (1-cells) of weight 1 and one segment of weight 2. There are two 3-valent vertices (0-cells) and one 5-valent vertex (shown in part 1 of the figure).

We first choose one segment of $G$, as indicated in part 2 of the figure. By choosing the segment, we have also chosen two vertices. We analyze each of these vertices to decompose them. The upper right vertex cannot be decomposed, so we must add the other two adjacent segments to our set. The lower left vertex can be decomposed, in two different ways. We choose one of these decompositions and add the corresponding segments to our set (shown in part 3). It is important to note that if we had chosen the other decomposition, we would have gotten a different factorization, as decomposing an $(n-1)$-cell is equivalent to decomposing a two-dimensional Newton polyhedron. For an algorithm that can do this, see [4].
this polynomial can be factored in two different ways. Finally, this step has given us one more vertex. It cannot be decomposed, so we add two more segments to our set (part 4).

Since all of the vertices are now balanced, we have found a balanced subgraph, which by Theorem 12 corresponds to a factor of $f$. This (red) balanced subgraph corresponds to the polynomial $x^2 \oplus y^2 \oplus x \oplus y \oplus 2$, while the remaining (blue) part corresponds to the polynomial $x \oplus y \oplus 0$. The reader can check that these two factors indeed multiply to give $f$.

### 3.3.3 Analysis

The efficiency of this algorithm depends on the number of decompositions of a the $(n-1)$-cells of $\mathcal{W}(f)$. For polynomials of a fixed degree, the number of decompositions of the vertices of the corner loci of these polynomials is inversely proportional to the number of vertices. So in general, for polynomials of a fixed degree, polynomials whose corner loci have more vertices can be factored more efficiently.

One interesting attribute of this factoring algorithm is that regardless of the dimension of the polyhedral complex, the problem of decomposing $(n-1)$-cells is the same. Decomposing $(n-1)$ cells is equivalent to the problem of decomposing a two-dimensional Newton polyhedron. In [4] it was shown that decomposing higher dimensional Newton polyhedron can help factor classical polynomials. Since decomposing an $n$-dimensional Newton polyhedron is equivalent to factoring a tropical polynomial that has a single 1-cell, we can decompose an $n$-dimensional Newton polyhedron simply by decomposing its two-dimensional faces.
1. Start with the weighted balanced graph of $f$.

2. Choose one segment of the weighted balanced graph.

3. Decompose each of the chosen vertices and add the resulting segments to $S$.

4. Continue until there are no more unbalanced vertices.

Figure 3.5: The process of factoring a polynomial by decomposing its weighted balanced graph. The numbers correspond to the weights of the segments. It is especially important to note that in 3. we have arbitrarily chosen one of the two non-trivial decompositions of the 5-valent vertex.
Chapter 4

Conclusion

I have shown that tropical polynomials with rational coefficients can be factored into a product of linear polynomials. My elementary proof explains some of the nuances of functional equivalence and introduces least-coefficient polynomials as representatives of functional equivalences classes. I have given an algorithm to convert a polynomial into least-coefficient form and show that least-coefficient polynomials can be factored by inspection. Overall, my research provides a basis by which others can better understand one-variable polynomials.

I have also shown that the problem of factoring tropical polynomials of two or more variables is NP-complete. I have discussed weighted balanced complexes and their usefulness in factoring tropical polynomials. I have also given an algorithm that can factor tropical polynomials. My research on multi-variable polynomials provides a better view at the geometry behind tropical polynomials.

I have also written some Maple programming code that can graph and manipulate tropical polynomials. This code is included in Appendix 4 and has been very useful to the BYU Tropical Algebra research group and will also be useful to others who study tropical polynomials.
Appendix: The Tropical Maple Package

During my research for this thesis and other related problems in tropical geometry, I produced the Tropical Maple Package. It is compatible with Maple versions 10 and higher. It is capable of graphing and otherwise manipulating tropical polynomials. It can also compute tropical resultants and determinants of tropical matrices.

Perhaps the most useful of these procedures are plotgraph and graph, which together with the internal procedure dual determine the corner locus of a polynomial in two variables. Note that there is not an implementation of our factoring algorithm, since this implementation was not completed at the time of printing.

Tropical:= module()
option package;
local trop, endpoints, Add, Multiply, Det, Isect, t, symcheck,
    maxmin, cv, bestrange, LATEX, dual;
export Latex, tropical, coefs, graph, plotgraph, dualgraph, multiply,
    add, deriv, minfunc, plotgraphs, Poly, toLCP, isLCP, det,
    SylvesterMatrix, resultant, makeparametric, makepoints, setCoordinates,
    setMaxMin, maxfunc, colorlist, isect, toCCP;
maxmin:=-1; #change this to 1 for max or -1 for min
cv:=(a)->[a[1],a[2]];
colorlist:=[red, green, blue, magenta, cyan, black, tan, yellow, navy,
    coral, plum, grey, aquamarine];

trop:=proc(poly)
description "Convert a tropical polynomial into a list of classic
polynomials. This allows the user to enter tropical polynomials in standard polynomial form.

if type(poly,'+') then op(map(trop,[op(poly)]))
elif type(poly,'*') then '+'(op(map(trop,[op(poly)])))
elif type(poly,'^') then '*'(op(map(trop,[op(poly)])))
elif type(poly,list(list)) then
  seq(a[1]+a[2]*x+a[3]*y+'if'(nops(a)>3,a[4]*z,0), a in poly)
elif type(poly,Array) then
  seq(poly[a,1]+poly[a,2]*x+poly[a,3]*y+'if'(op([2,2,2],poly)>3,
poly[a,4]*z,0),a=op([2,1],poly))
elif type(poly,list(algebraic)) then op(poly)
else poly
fi;
end proc:

minfunc:=proc(poly)
local s1,s2;
description "Returns a functional representation of a polynomial";
if nops([args])>1 then
  if type(op(2,[args]),symbol) then
    s1:=convert(op(2,[args]),string)
  else
    s1:=convert(op(2,[args]),string)[2..-2]
  fi;
  s2:=convert(min(op(tropical(poly))),string);
  parse("("||s1|| ")->" ||s2);
else
  min(op(tropical(poly)))
fi;
end proc;

maxfunc:=proc(poly)
local s1,s2;
description "Returns a functional representation of a polynomial";
if nops([args])>1 then
  if type(op(2,[args]),symbol) then
    ...
Appendix: The Tropical Maple Package

s1:=convert(op(2,[args]),string)
else
s1:=convert(op(2,[args]),string)[2..-2]
fi;
s2:=convert(max(op(tropical(poly))),string);
parse("("||s1|| ")->"||s2);
else
max(op(tropical(poly)))
fi;
end proc;

coefs:=proc(poly)
local f,g,clist,a,b,templist;
description "Returns an array of the coefficients and powers of
polynomial terms for a polynomial in x,y,and z. Used internally,
but also useful to the user."
clist:=[];
f:=tropical(poly);
for g in f do
b:=[if(type(g,'+'),op(g),g)];
templist:=[0,0,0,0];
for a in b do
if member('x',indets(a)) then
templist[2]:=a/x
elif member('y',indets(a)) then
templist[3]:=a/y
elif member('z',indets(a)) then
templist[4]:=a/z
else templist[1]:=a
fi
od;
clist:=[op(clist),templist];
od;
if nops([args])>1 then
clist
else
Array(1..nops(clist),1..4,clist);
fi;
end proc:

c := proc (poly)
local duallines, lines, l1, l2, dir, p1, p2, p3, proj, t, opts, temp, s;

graph:=proc(poly)
local duallines, lines, l1, l2, dir, p1, p2, p3, proj, t, opts, temp, s;
description "Returns a symbolic representation of the corner
locus of a polynomial";
if convert([seq(a in {Ray,Segment,Line,Point},a in
{seq(op(0,b),b in poly)}),'and' ) then
lines:=poly;  # if poly is a symbolic graph, then we are already done.
else
  if type(poly,'+') and # some basic error checking
    not type(poly,polynom(extended_numeric,[x,y,zero,one,‘1’,‘0’]))
    and nops({seq(degree(i,[x,y,z]),i in poly)})>1 then
      WARNING("Received a non-homogenous polynomial in three or more
variables. I will try to scale out z to form a polynomial in
two variables.");
    fi;
  duallines:=dual(poly);
  # the dual procedure does the actual work. The rest of this is
  # just converting the result into a symbolic tropical graph.
  lines:=[ ];
  while nops(duallines)>0 do
    l1:=duallines[1];
    duallines:=duallines[2..-1];
    l2:=NULL;
    for s from 1 to nops(duallines) while l2=NULL do
      if evalb(l1[1,1] in duallines[s,1] and l1[1,2] in duallines[s,1])
        then
          l2:=duallines[s];
          duallines:=[op(duallines[1..s-1]),op(duallines[s+1..-1])];
        fi;
      od;
    if l2=NULL then # we use projections to find a direction vector.
      p1:=op([1,1],l1);
      p2:=op([1,2],l1);
      p3:=l1[3];
    if dir[1]=0 and dir[2]=0 then
      lines:=[op(lines),Line(cv(l1[2]),cv(l1[2]),cv(dir),
        igcd(l1[1,1,1]−l1[1,2,1],l1[1,1,2]−l1[1,2,2]));
    else
      lines:=[op(lines),Ray(cv(l1[2]),cv(dir),
        igcd(l1[1,1,1]−l1[1,2,1],l1[1,1,2]−l1[1,2,2]));
      fi
else #that is, if l2 is not NULL
    lines:=\[op(lines), Segment(cv(l1[2]), cv(l2[2]-l1[2]),
        igcd(l1[1,1,1]-l1[1,2,1], l1[1,1,2]-l1[1,2,2]))\];
fi;
od;
fi;
opts:=\[args[2..-1]\];
temp:=\'default\';
hasoption(opts,type,'temp','opts');
#return section
if temp='parametric' then
    makeparametric(lines)
else
    lines;
fi;
end proc:

plotgraph:=proc(poly)
    local ff, gra, temp, opts;
    description "Draws the tropical zero locus of a tropical polynomial";
    #some basic error checking
    if type(poly, list) and not (type(poly, list(list)) or
        type(poly, list(anything))) then
        WARNING("First argument of plotgraph must be a single polynomial."
            Use 'plotgraphs' for multiple polynomials."
       );
    elif not type(poly, polynom) and not type(poly, list(list)) and
        not type(poly, list(anything)) then
        WARNING("Unexpected first argument."
    fi;
    opts:=\[args[2..-1]\];
    gra:=graph(poly);
    #view
    temp:=bestrange(gra);
    hasoption(opts, view,'temp','opts');
    opts:=\[op(opts), view=temp\];
    #thickness
    temp:=2;
    hasoption(opts, thickness,'temp','opts');
    if temp='weights' then
        temp:=\[seq(op(3,a), a in gra)\];
    fi;
    opts:=\[op(opts), thickness=temp\];
    #color
temp:=red;
hasoption(opts,color,'temp','opts');
if evalb(temp<>'default') then
  if evalb(temp='weights') then
    temp:=[seq(colorlist[((op(3,a)-1) mod nops(colorlist))+1],
            a in gra)];
  fi;
  opts:=[op(opts),color=temp];
  fi;
if nops(gra)>0 then
  temp:=makeparametric(gra);
  if nops(temp)>0 then
    plots[display](plot(makeparametric(gra),op(opts)),makepoints(gra),
                   symbol=cross,symbolsize=20);
  else
    plots[display](makepoints(gra),symbol=cross,symbolsize=20,
                   op(opts));
  fi;
else
  WARNING("empty corner locus. Returned NULL");
  return NULL;
fi;
end proc:

plotgraphs:=proc(polylist)
local s,opts,graphs,f,thicklist,sglist,cl;
description "Plots multiple graphs together, each a different color";
cl:=colorlist;
opts:=[args[2..-1]];
#some error checking
if nops(opts)>0 and type(opts[1],polynom) then
  error "Invalid 2nd argument. Put all polynomials in a list
    first argument: plotgraphs([f,g]);";
fi;
thicklist:=[2];
hasoption(opts,color,'cl','opts');
hasoption(opts,thickness,'thicklist','opts');
if cl='weights' then error "Invalid option: color=weights
    can be used with plotgraph command only."; fi;
if not type(cl,list) then cl:=[cl] fi;
if thicklist='weights' then error "Invalid option: thickness=
    weights can be used with plotgraph command only."; fi;
if not type(thicklist,list) then thicklist:=[thicklist] fi;
s:=0;
sglist:=map(graph,polylist); #make symbolic graphs
if not(hasoption(opts,view)) then opts:=[op(opts),view=bestrange([seq(op(o),o in sglist)])] fi;
graphs:=[];
for f in sglist do
  graphs:=[op(graphs),plotgraph(f,op(opts),color=
    cl[(s mod nops(cl))+1],thickness=thicklist[(s mod
    nops(thicklist))+1])];
s:=s+1;
od;
plots[display](graphs);
end proc;

dual:=proc(poly)
  local f,g,u,v,w,vertices,f0,fmin,t,points,h,hh,lines,xx,yy,det,
    vertices2,ncc;
  description "Analyzes the dual of a polynomial in order to
    draw the dual graph or the corner locus of a polynomial."
  f:=eval(tropical(poly),z=0);
g:=coefs(poly);
  vertices:={}:
  vertices2:={}:
  ncc:={}:
  for u from 1 to nops(f) do
    if u in ncc then next fi;
    for v from u+1 to nops(f) do
      if v in ncc then next fi;
      for w from v+1 to nops(f) do
        if w in ncc then next fi;
        det:=(g[u,2]-g[v,2])*(g[u,3]-g[w,3])-(g[u,2]-g[w,2])*
          (g[u,3]-g[v,3]);
        if det<>0 then
          xx:=(g[u,3]-g[w,3])/det*(g[v,1]-g[u,1])+(g[v,3]-g[u,3])/det*
            (g[w,1]-g[u,1]);
          yy:=(g[w,2]-g[u,2])/det*(g[v,1]-g[u,1])+(g[u,2]-g[v,2])/det*
            (g[w,1]-g[u,1]);
          if not [xx,yy] in vertices2 then
            f0:=eval(f,[x=xx,y=yy]);
            fmin:=maxmin*max(op(maxmin*f0));
            if fmin=f0[u] then
              next fi;
          end if;
        end if;
      end do;
    end do;
  end do;
end proc;
points := {};
for t from 1 to nops(f0) do
if f0[t] = fmin then points := points union {t} fi
od;

hh := simplex[convexhull]([seq([maxmin*g[a,2], maxmin*g[a,3]], a in points))];
vertices := vertices union {{hh, [xx, yy]}];
vertices2 := vertices2 union {{xx, yy}};
for h in points do
if not [maxmin*g[h,2], maxmin*g[h,3]] in hh then
ncc := ncc union {h}
fi
od;
fi
fi
od
od
od;

lines := [];
for v in vertices do
hh := v[1];
lines := [op(lines), seq([{hh[t], hh[t+1]}, v[2], hh['if'(t+2>nops(hh),
t-1, t+2)]], t = 1..nops(hh)-1), [{hh[1], hh[-1]}, v[2], hh[2]]];
od;
if nops(lines) = 0 then
# this is in case there are no polygons in the dual (only edges)
vertices := {};
for u from 1 to nops(f) do
for v from u+1 to nops(f) do
if g[u,2] = g[v,2] then
xx := 0;
yy := (g[v,1]-g[u,1])/(g[u,3]-g[v,3]);
else
xx := (g[v,1]-g[u,1])/(g[u,2]-g[v,2]);
yy := 0;
fi;
f0 := eval(f, [x=xx, y=yy]);
fmin := maxmin*max(op(maxmin*f0));
points := {};
for t from 1 to nops(f0) do
if f0[t] = fmin then
Appendix: The Tropical Maple Package

points:=points union \{maxmin* [g[t,2], g[t,3]]\}
fi;
od;
if nops(points)>1 then
  vertices:=vertices union \{points, [xx, yy]\}
fi;
od;
lines:=[];
for v in vertices do
  hh:=[op(endpoints(v[1]))];
  lines:=[op(lines), [{hh[1], hh[2]}, v[2], hh[2]]];
  od;
fi;
lines;
end proc:
dualgraph:=proc(poly)
  local d, g, pairedlines, lines, side, dotlist, pair;
  description "Graphs a tropical polynomial dual graph";
  g:=coefs(poly);
  pairedlines:=dual(poly);
  if nops(pairedlines)=0 then
    error "Empty corner locus"
  else
    lines:={};
    for pair in pairedlines do
      lines:=lines union \{pair[1]\};
    od;
    if member('dots', [args]) then
      d:=degree(Poly(poly));
      dotlist:=seq(seq(plottools[point]([[maxmin* xx, maxmin* yy]]),
                     xx=0..(d-yy), yy=0..d);
    else
      dotlist:=NULL
    fi;
    plots[display]([seq(plottools[line](op(1,1), op(2,1)), l in lines),
                   dotlist], tickmarks=[[\[-1="1\]],[\[-1="1\]]], axes=none,
                   scaling=constrained);
  fi;
end proc:
Multiply:=proc(poly)
    local ff,p,f1,f2;
    description "Multiplies two or more polynomials [internal for faster computing];
    if nargs=0 then return 0 fi;
    ff:=Add(poly);
    for p in args[2..-1] do
        ff:=Add([seq(seq(a+b,a in coefs(p,list)),b in coefs(ff,list))])
    od;
    ff;
end proc;

multiply:=proc()
    description "Multiplies two polynomials [for user]"
    Poly(Multiply(args));
end proc;

Add:=proc(poly)
    local t,ff,s,w;
    description "Adds a number of polynomials";
    w:=false;
    ff:=[seq(op(coefs(p,list)),p in [args]]);
    for t from 1 to nops(ff) while t<=nops(ff) do
        if type(ff[t,1],infinity) then
            #if ff[t,1]<>infinity then
                #w:=true fi;
            ff:=[op(ff[1..t-1]),op(ff[t+1..nops(ff)])];
            t:=t-1;
        else
            for s from t+1 to nops(ff) while s<=nops(ff) do
                if ff[t,2]=ff[s,2] and ff[t,3]=ff[s,3] and ff[t,4]=ff[s,4] then
                    ff:=[op(ff[1..t-1]),[maxmin*max(maxmin*ff[t,1],maxmin*ff[s,1]),
                        ff[t,2],ff[t,3],ff[t,4]],op(ff[t+1..s-1]),op(ff[s+1..nops(ff)])];
                    s:=s-1;
                fi;
            od;
        fi;
    od;
    #if w then WARNING("w") fi;
    if nops(ff)=0 then
        infinity
    else
        ff
end proc;
change:
end proc;

add:=proc()
    Poly(Add(args));
end proc;

deriv:=proc(poly,vars)
    description "Takes the derivative of a tropical polynomial";
    local d,a,var,dlist,onevar;
    if type(vars,symbol) then onevar:=true else onevar:=false fi;
    dlist:=[ ];
    for var in vars do
        d:=[ ];
        for a in tropical(poly) do
            if member(cat(var),indets(a)) then d:=[op(d),a-var] fi;
        od;
        dlist:=[op(dlist),d]
    od;
    if onevar then Poly(op(dlist)) else map(Poly,dlist) fi;
end proc;

dendpoints:=proc(points)
    local mx,Mx,my,My,p,ep;
    description "Finds the endpoints of a line segment [internal]";
    mx:=min(seq(p[1],p in points));
    Mx:=max(seq(p[1],p in points));
    my:=min(seq(p[2],p in points));
    My:=max(seq(p[2],p in points));
    ep:={};
    for p in points do
            ep:=ep union {p}
        fi
    od;
    ep;
end proc;

Poly:=proc(poly)
    local g,t,c;
    description "Makes a polynomial out of a tropical polynomial";
    if type(poly,polynom) then
Appendix: The Tropical Maple Package

poly
else
g:=coefs(poly);
c:=op([2,1,2],g);
if c=0 then return infinity fi;
for t from 1 to c do
if g[t,1]=1 and (g[t,2]<0 or g[t,3]<0 or g[t,4]<0) then
  g[t,1]:='1'
elif g[t,1]=0 then
  if g[t,2]=0 and g[t,3]=0 and g[t,4]=0 then
    g[t,1]:='0'
  else
    g[t,1]:=1
  fi;
fi;
od;
'+'(seq(g[t,1]*x^g[t,2]*y^g[t,3]*z^g[t,4],t=1..c));
fi;
en end proc;

isLCP:=proc(poly1)
local g,corners,u,v,tf,notinf,vertices,t,poly;
description "Checks if a polynomial is least coefficients.";
tf:=true;
poly:=eval(poly1,z='0');
g:=coefs(poly);
notinf:=[seq([g[a,2],g[a,3]],a=op([2,1],g))];
corners:=simplex[convexhull](notinf);
for u from min(seq(g[a,2],a=op([2,1],g))) to
  max(seq(g[a,2],a=op([2,1],g))) while tf do
  for v from min(seq(g[a,3],a=op([2,1],g))) to
    max(seq(g[a,3],a=op([2,1],g))) while tf do
    if not evalb([u,v] in simplex[convexhull]([op(corners),[u,v]]))
      and not evalb([u,v] in notinf) then
      tf:=false
    fi
  od #v
od; #u
if tf then
  vertices:={seq(a[2],a in dual(poly))};
  for t from 1 to op([2,1,2],g) while tf do
    if g[t,1]>min(g[t,1],max(seq(eval(minfunc(poly),[x=v[1],y=v[2]]))-

tf:=false;
fi
od;
fi;

end proc;

end proc;

toLCP:=proc(poly1)
local g,corners,u,v,notinf,vertices,monomial,t,inf,s,poly;

description "Determines a least coefficient polynomial
for a polynomial in two variables";
poly:=eval(poly1,z='0');
g:=coefs(poly);
notinf:=[seq([g[a,2],g[a,3]],a=op([2,1],g))];
inf:=[];
corners:=simplex[convexhull](notinf);
for u from min(seq(g[a,2],a=op([2,1],g))) to max(seq(g[a,2],
a=op([2,1],g))) do
for v from min(seq(g[a,3],a=op([2,1],g))) to max(seq(g[a,3],
a=op([2,1],g))) do
if not evalb([u,v] in simplex[convexhull]([op(corners),[u,v]])) and
not evalb([u,v] in notinf) then
inf:=[op(inf),[u,v,0]];
fi
od #v
od; #u
s:=op([2,1,2],g);
g:=Array(1..s+nops(inf),1..4,g);
for t from 1 to nops(inf) do
g[t+s,1]:=infinity;
g[t+s,2]:=inf[t,1];
g[t+s,3]:=inf[t,2];
g[t+s,4]:=inf[t,3];
od;
vertices:={seq(a[2],a in dual(poly))};
for t from 1 to op([2,1,2],g) do
g[t,1]:=min(g[t,1],max(seq(eval(minfunc(poly),[x=v[1],y=v[2]])-
g[t,2]*v[1]-g[t,3]*v[2],v in vertices)));
od;
Poly(g);
endproc;
Appendix: The Tropical Maple Package

\texttt{toCCP:=proc(poly)}
\texttt{description "Converts a polynomial to Contributing Coefficient Form";}
\texttt{local f,d,l,t;}
\texttt{d:=dual(poly); l:={seq(op([maxmin*op([1,1],g),maxmin*op([1,2],g)]),g in d); f:=coefs(poly);}
\texttt{Poly([seq('if'([f[t,2],f[t,3]]) in l,f[t,1]+f[t,2]*x+f[t,3]*y+f[t,4]*z,NULL),t=op([2,1],f))]);}
\texttt{end proc;}

\texttt{Det:=proc(mat)}
\texttt{#errorchecking}
\texttt{if type(mat,Matrix) then}
\texttt{if op([1,2],mat)<op([1,1],mat) then}
\texttt{error "expected nxn matrix but received %1x%2 matrix", op([1,1],mat),op([1,2],mat)}
\texttt{fi;}
\texttt{if op([1,1],mat)=1 then}
\texttt{mat[1,1]}
\texttt{elif op([1,1],mat)=2 then}
\texttt{Add(Multiply(mat[1,1],mat[2,2]),Multiply(mat[1,2],mat[2,1]));}
\texttt{else #n>2}
\texttt{Add(seq(Multiply(mat[1,i],Det(mat[2..op([1,1],mat),
[1..(i-1),(i+1)..op([1,1],mat)])]),i=1..op([1,1],mat)));}
\texttt{fi;}
\texttt{elif type(mat,list) and not type(mat,list(list)) then}
\texttt{if not type(sqrt(nops(mat)),integer) then}
\texttt{error "list length must be the square of an integer. (Received a list of length %1)"},nops(mat);\texttt{fi;}
\texttt{Det(Matrix(sqrt(nops(mat)),sqrt(nops(mat)),[seq([seq(
mat[sqrt(nops(mat))*(i-1)+j],j=1..sqrt(nops(mat))))],
 i=1..sqrt(nops(mat))))});}
\texttt{else}
\texttt{if not type(mat,list(list)) and not type(mat,array) then}
\texttt{WARNING("unrecognized input. Input should be of type Matrix or list");}
\texttt{fi;}
\texttt{Det(Matrix(mat));}
\texttt{fi;}
\texttt{end proc;}
Appendix: The Tropical Maple Package

det:=proc()
    Poly(Det(args));
end proc;

SylvesterMatrix:=proc(p1,p2,v)
    local M,t,s;
    M:=LinearAlgebra[SylvesterMatrix](p1,p2,v);
    for t from 1 to op([1,1],M) do
        for s from 1 to op([1,2],M) do
            if M[t,s]=0 then M[t,s]:=infinity fi;
            if M[t,s]=1 then M[t,s]:=0 fi;
        od;
    od;
    M;
end proc;

resultant:=proc(p1,p2,v)
    det(SylvesterMatrix(p1,p2,v));
end proc;

makeparametric:=proc(o)
    description "Returns a parametric version of a symbolic graph";
    if nops([args])>1 then map(makeparametric,[args])
    elif type(o,list) then map(makeparametric,o);
    elif op(0,o)='Line' then
        [op([1,1],o)+op([2,1],o)*t,op([1,2],o)+op([2,2],o)*t, t=-infinity..infinity]
    elif op(0,o)='Ray' then
        [op([1,1],o)+op([2,1],o)*t,op([1,2],o)+op([2,2],o)*t, t=0..infinity]
    elif op(0,o)='Segment' then
        [op([1,1],o)+op([2,1],o)*t,op([1,2],o)+op([2,2],o)*t, t=0..1]
    fi;
end proc:

makepoints:=proc(o)
    description "Returns a list of points extracted from a symbolic graph";
    if nops([args])>1 then map(makepoints,[args])
    elif type(o,list) then map(makepoints,o);
    elif op(0,o)='Point' then
        plottools[point](op(o));
    fi;
bestrange:=proc(symbolicgraph)
    description "Determines automatic range for a symbolic graph";
    local r1,s;
    r1:=[min(seq(op([1,1],g),g in symbolicgraph)),max(seq(op([1,1],g),
        g in symbolicgraph)),min(seq(op([1,2],g),g in symbolicgraph)),
        max(seq(op([1,2],g),g in symbolicgraph))];
    for s from 1 to 4 by 2 do
        if r1[s]=r1[s+1] then
            r1[s]:=r1[s]-1;
            r1[s+1]:=r1[s+1]+1;
        else
            r1[s]:=r1[s]-.5*(r1[s+1]-r1[s]);
            r1[s+1]:=r1[s+1]+.5*(r1[s+1]-r1[s]);
        fi
    od;
    s:=r1[2]-r1[1]-(r1[4]-r1[3]);
    if evalf(s)>0 then
        r1[4]:=r1[4]+s/2;
        r1[3]:=r1[3]-s/2;
    else
        r1[2]:=r1[2]-s/2;
        r1[1]:=r1[1]+s/2;
    fi;
    return [r1[1]..r1[2],r1[3]..r1[4]];
end proc;

Isect:=proc(object1,object2)
    description "Internal procedure to determine intersection of two
        Point-Segment-Ray-Line objects";
    local oo,o1,o2,p,t,r;
    o1:=symcheck(object1); o2:=symcheck(object2);
    if op(0,o1)=Point then
        oo:=o1;o1:=o2;o2:=oo;
    fi;
    if op(0,o2)=Point then #o2 is a point
        if op(0,o1)=Point then #if both are points
            if o1=o2 then return o1 else return NULL fi;
        fi;
        if op([2,2],o1)*(op([1,1],o1)-op([1,1],o2))<>op([2,1],o1)*
            (op([1,2],o1)-op([1,2],o2)) then
            return NULL;
        fi;
    fi;
end proc:
if \( \text{op}([2,1],o1)=0 \) then
    \[ t := \frac{\text{op}([1,2],o2)-\text{op}([1,2],o1)}{\text{op}([2,2],o1)} \]
else
    \[ t := \frac{\text{op}([1,1],o2)-\text{op}([1,1],o1)}{\text{op}([2,1],o1)} \]
fi;

if \( \text{op}(0,o1)=\text{Line} \) or
    (\( \text{op}(0,o1)=\text{Ray} \) and \( t \geq 0 \)) or
    (\( \text{op}(0,o1)=\text{Segment} \) and \( t \geq 0 \) and \( t \leq 1 \)) then
    return o2
else
    return NULL;
fi;

if \( \text{op}([2,1],o1)*\text{op}([2,2],o2) = \text{op}([2,2],o1)*\text{op}([2,1],o2) \) then
    \# (they are parallel)
    if \( \text{op}([2,2],o1)*(\text{op}([1,1],o1)-\text{op}([1,1],o2)) \neq \text{op}([2,1],o1)*\)
        (\( \text{op}([1,2],o1)-\text{op}([1,2],o2) \)) then
        return NULL;
    fi;
    if \( \text{op}(0,o2)=\text{Line} \) then
        oo := o1; o1 := o2; o2 := oo;
    fi;
if \( \text{op}(0,o1)=\text{Line} \) then
    \# (one of them is a line)
    return o2;
fi;

if \( \text{op}(0,o1)=\text{Ray} \) and \( \text{op}(0,o2)=\text{Ray} \) then
    if \( \text{op}([2,1],o2)*\text{op}([2,1],o1)+\text{op}([2,2],o2)*\text{op}([2,2],o1)>0 \) then
        if \( (\text{op}([1,1],o1)-\text{op}([1,1],o2))*\text{op}([2,1],o2)+(\text{op}([1,2],o1)-\)
            \( \text{op}([1,2],o2))*\text{op}([2,2],o1)>0 \) then
            return o1
        else
            return o2
        fi;
    else
        if \( (\text{op}([1,1],o2)-\text{op}([1,1],o1))*\text{op}([2,1],o1)+(\text{op}([1,2],o2)-\)
            \( \text{op}([1,2],o2))*\text{op}([2,2],o2) \geq 0 \) then
            return \text{symcheck(Segment(op(1,o1),op(1,o2)-op(1,o1),
                min(op(3,o1),op(3,o2))))};
        else
            return NULL;
        fi;
    fi;
fi;
Appendix: The Tropical Maple Package

if op(0, o1) = Ray then
    oo := o1; o1 := o2; o2 := oo;
fi;

if op([2, 1], o1) = 0 then
    t := [(op([1, 2], o2) - op([1, 2], o1))/op([2, 1], o1),
         op([2, 1], o2)/op([2, 1], o1)]
else
    t := [(op([1, 1], o2) - op([1, 1], o1))/op([2, 1], o1),
         op([2, 1], o2)/op([2, 1], o1)]
fi;

if op(0, o2) = Ray then
    r := [0, infinity]
else
    r := [0, 1]
fi;

r := sort([t[1], t[1]] + t[2]*r);

if r[2] >= r[1] then
    return symcheck(Segment(op(1, o1) + op(2, o1)*r[1],
                              op(2, o1)*(r[2] - r[1]),
                              min(op(3, o1), op(3, o2))));
else
    return NULL;
fi; # end case: they are parallel

# find the point where they would intersect
p := [(op([2, 2], o2)*op([1, 1], o1) - op([1, 1], o2)) - op([2, 1], o2)*
     (op([1, 2], o1) - op([1, 2], o2))/
     (op([2, 2], o1)*op([2, 1], o2) - op([2, 1], o1)*op([2, 2], o2)),
     (op([2, 2], o1)*op([1, 1], o2) - op([1, 1], o1)) - op([2, 1], o1)*
     (op([1, 2], o2) - op([1, 2], o1))/
     (op([2, 2], o2)*op([2, 1], o1) - op([2, 1], o2)*op([2, 2], o1))];

for t from 1 to 2 do
    if op(0, [o1, o2][t]) = Ray and p[t] < 0 then
        return NULL;
    elif op(0, [o1, o2][t]) = Segment and ((p[t]) < 0 or (p[t]) > 1) then
        return NULL;
    fi;
    od;
    return Point(op(1, o1) + p[1]*op(2, o1));
end proc;
isect:=proc(o1)
    description "Returns the intersection of two symbolic graphs";
    if o1=[] then return [] fi;
    if type(o1,list({polynom,list})) then
        isect(op(o1));
    elif nargs=1 then
        graph(o1);
    elif nargs=2 then
        [op({seq(seq(Isect(a,b),a in graph(o1)),b in graph(args[2]))})];
    else
        isect(isect(o1,args[2]),args[3..-1]);
    fi;
end proc;

symcheck:=proc(object)
    description "Checks a symbolic graph for validity";
    if op(0,object)=Point and nops(object)<1 then
        error "Invalid Point Object";
    fi;
    if op(0,object) in {Ray,Segment,Line} then
        if nops(object)<>3 then
            error "Invalid "||(op(0,object));
        elif op(2,object)=[0,0] then
            return Point(op(1,object))
        fi;
    fi;
    object;
end proc;

setCoordinates:=proc()
    description "Sets the Coordinate System";
    local c;
    if nargs>1 then
        error "expected one argument of type procedure or symbol, received %1 arguments",nargs
    elif nargs=1 then
        c:=op(1,[args]);
        if type(c,procedure) then
            cv:=c;
        elif type(c,symbol) then
            if c=default then
                cv:=a->[a[1],a[2]];
            end if;
        end if;
    fi;
end proc;
elif c=equilateral then
    cv:=a->[a[1]-a[2]/2,sqrt(3)/2*a[2]];
else
    error "expected 'default' or 'equilateral' or user-defined procedure, received '%1'.",c;
    fi;
else
    error "expected 'default' or 'equilateral' or user-defined procedure, received type '%1',whattype(c);
    fi;
fi;
print(cv)
end proc;

setMaxMin:=proc(x)
    if x='max' then maxmin:=1; 'max';
    elif x='min' then maxmin:=-1; 'min';
    fi;
end proc;

Latex:=proc(poly)
    printf(LATEX(poly));
end proc;

LATEX:=proc(poly);
    if type(poly,'+') then
        cat("","seq(op([LATEX(op(i,poly))," \oplus "])),i=1..
            (nops(poly)-1)),LATEX(op(-1,poly)));
    elif type(poly,'*') then
        cat("","seq(LATEX(op(i,poly)),i=1..nops(poly)));
    elif type(poly,'~') then
        cat("","LATEX(op(1,poly)),seq(op(["~","op(i,poly),"}]),
            i=2..nops(poly)));
    elif type(poly,{numeric,symbol}) then
        if type(poly,infinity) then
            "\\infty 
        else
            convert(poly,string)
        fi;
    elif type(poly,Matrix) then
        cat("\\left[\\begin{array}{c\ldots\}n",op([1,2],poly)),"\\n",}
seq(op([seq(op([LATEX(poly[i,j]),"\"\"]),j=1..op([1,2],poly)-1),
        LATEX(poly[i,-1]),"\\n"]),i=1..op([1,1],poly)-1),
    seq(op([LATEX(poly[-1,j]),"\"\"]),j=1..op([1,2],poly)-1),
    LATEX(poly[-1,-1]),
    \"\n\end{array}\right\});
else
    WARNING("I don't know how do LaTeX that...I might have messed up.
             (Type: %1)",whattype(poly));
    convert(poly,string);
fi;
end proc;

end module:
Bibliography


Index

balancing condition, 23
boundary cell, 23
cell, 22
cell complex
  rational polyhedral, 22
  weighted, 22
  weighted balanced, 21 23
corner locus, 15 20
correspondence
  polynomial-complex, 24
factorization
  algorithm, 30
  non-unique, 19 31
  NP-completeness, 29
functional equivalence, 6 20
graph, tropical, 24
integer covector, 23
interior cell, 23
least coefficient, 7 8
least-coefficient polynomial, 8 12
notation, 3 20
Q, 3
Simon, Imre, 1
subcomplex, 28
tropical algebra
  applications, 1
  definition, 3
  fundamental theorem, 5 13
tropical graph, 24
tropical zero locus, 15 20
weighted balanced complex, 21
  of a polynomial, 25
  of a product, 27
  polynomial of, 25
weighted complex, 22