Reflexive Lattice Polytopes

G14DIS

Mathematics 4th Year Dissertation

2019/20

School of Mathematical Sciences

University of Nottingham

Toby Willis

Supervisor: Dr. Alexander Kasprzyk

Project code: AK D1

Assessment type: Research-informed Investigation

I have read and understood the School and University guidelines on plagiarism. I confirm that this work is my own, apart from the acknowledged references.

Abstract

We provide an introduction to the field of Lattice Polytopes, with extensive examples to enable a new reader to develop a thorough and supported understanding of the basics. We build on this to develop computational methods for finding polytopes, and demonstrate how the implementation of these methods results in some interesting results, specifically with relation to the existence of reflexive lattice polytopes with width greater than 2.

Acknowledgements

I would like to acknowledge the considerable help and support given by many people during this academic endeavour, without which I could not have completed my dissertation. In particular, I would like to thank my supervisor, Alexander Kasprzyk for his invaluable feedback, positive criticism, and thorough support throughout this project, and above all for providing the opportunity to engage in further study throughout the year.

I would also like to thank Thomas Hall and Johannes Hofscheier, on whose work I built to develop my results, and to particularly thank Thomas for the assistance he provided me at several key moments of my writing process.

Lastly, I would like to acknowledge how the support and love of my family has sustained and helped me throughout. In particular to my parents Helen and Chris, who have supported me unconditionally, and my sister Caitlin for guiding me in converting some of my hand-drawn pictures into the professional-looking figure herein.

Contents

Notation and Definitions

We have:

$N, M :=$ integer lattices.

$S :=$ a sublattice of N.

$P, Q, R :=$ convex bounded polytopes

$\mathbb{P}$:= a projective space.

$\mathbb{P}(a_0, a_1, \ldots, a_n)$:= a weighted projective space with weights $(a_0, a_1, \ldots, a_n)$.

X^*:= the dual of the object X.

v:= a vertex of a mathematical object.

conv$(v_i, \ldots)$:= the convex hull of a number of points.

u:= a vector on a lattice.

H:= a hyperplane.

d:= a scalar representing primitive distance.

k_i:= a scalar used to help calculate weight a_i.

1 Introduction

1.1 Outline

The aim of this dissertation is to provide an introduction to the field of lattices and the properties of lattice polytopes, allowing a reader with an undergraduate pure mathematics background to develop a thorough understanding. Section 2 introduces the concepts of reflexive polytopes and their properties, with Section 3 examining the width of these polytopes. Section 4 introduces Projective Varieties, in particular Gorenstein weighted projective spaces, and discusses how these relate to lattice polytopes. Finally, Sections 5 and 6 discuss the process of finding Gorenstein weighted projective spaces in higher dimensions, and of examining the properties of the polytopes recovered. This will culminate in the demonstration of the existence of reflexive lattice polytopes in dimensions 5,6,7,8 and 9 with width ≥ 3. This is joint work with Thomas Hall.

2 Polytopes on the Integer Lattice

2.1 The Integer Lattice

The n-dimensional integer lattice is defined as follows:

$$N = \mathbb{Z}^n = \mathbb{Z} \times \mathbb{Z} \times \cdots \times \mathbb{Z},$$

where $\mathbb{Z}$ denotes the integers. Accordingly the 2-dimensional integer lattice is:

$$N = \mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}.$$

On this lattice, once two linearly independent basis vectors e_1 and e_2 and an origin point have been defined, any element of the lattice $x \in N$ can be written as $x = (x_1, x_2)$ (with $x_1, x_2 \in \mathbb{Z}$). This represents a move from the origin point to $x_1e_1 + x_2e_2$. The lattice in Figure 1 is an example of a 2-d lattice with orthogonal bases. It includes an x marking the origin point.

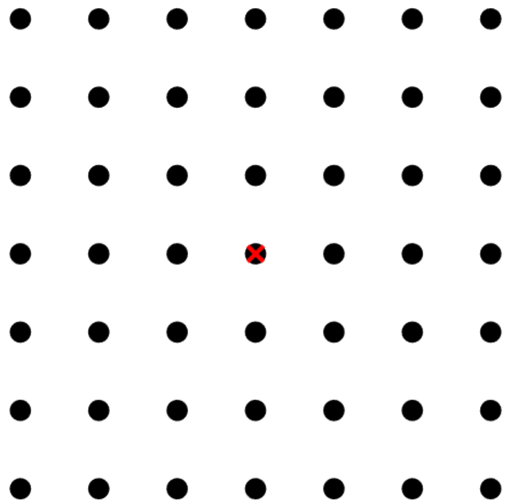

Figure 1: A 2-dimensional Orthogonal Lattice with the Origin Marked

The motivation for examining lattices is wide, and includes many areas of pure mathematics, such as Lie algebras, number theory and group theory. Applications also exist within the field of applied maths, for example in coding theory and cryptography.

2.2 Lattice Polytopes

A polytope is a geometric object with hyperplanar edges. A n-polytope is a polytope in n-dimensions, with n-1 dimensional facets (Jensen.A 2019). As such a 2-polytope is a polygon and a 3-polytope is a polyhedron. A lattice polytope requires that its vertices lie on points of the integer lattice.

Formally, a lattice polytope $P \subset \mathbb{Z}^n$ is the convex hull of finitely many points in $\mathbb{Z}^n$ (Balletti & Kasprzyk 2005). The convex hull of points $p_1, p_2, \ldots$ is written: $\text{conv}(p_1, p_2, \ldots)$. As such, our polytope P obeys the standard definition of convexity:

Polytope P is convex $\iff$ for $a, b \in P$ the closed segment with ends a and b is contained in P.

We have that the vertices of a polytope P are the minimal set of points required to construct P. This is to say, for any polytope P, the vertices $v_1, v_2, \ldots$ of P are such that $\text{conv}(v_1, v_2, \ldots) = P$, and removal of any of these does not result in P.

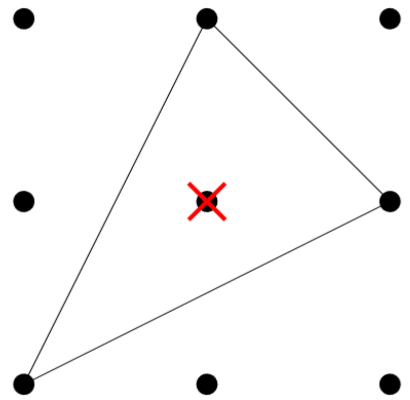

Figure 2: A Lattice Polytope

A hyperplane H is defined to span a space of degree one less than the space it lies within. In dimension 2, this therefore corresponds to a line. Any convex lattice polytope can be defined in two ways. Either, through a list of its vertices $[v_1, v_2, \ldots]$, or through a set of hyperplanes $[H_1, H_2, \ldots]$ which contain its facets.

Here we define lines as [u,(a,b)] where u is a vector that H is orthogonal to, and (a,b) is a lattice point that the line intersects (chosen arbitrarily).

It can be seen that any bounding line must travel through 2 vertices of the polytope, as the region where the line intersects an edge of the polytope will be bounded by vertices, which are lattice points. As such, there are the minimum 2 lattice points on the bounding line (at the vertices).

Using these definitions, the polytope given in Figure 2 is defined by either the set of points: (1,0),(0,1),(-1,-1), or the set of lines given by

$$[(-1,-1),(1,0)],[(2,-1),(0,1)],[(-1,2),(1,0)].$$

For the lattice polytope shown in Figure 3, the vertices are given by the set of points: (0,1),(-1,1),(-1,-1),(2,-1). The bounding lines are

$$[(1,1),(1,0)],[(0,1),(0,1)],[(-1,0),(-1,0)],[(0,-1),(0,-1)].$$

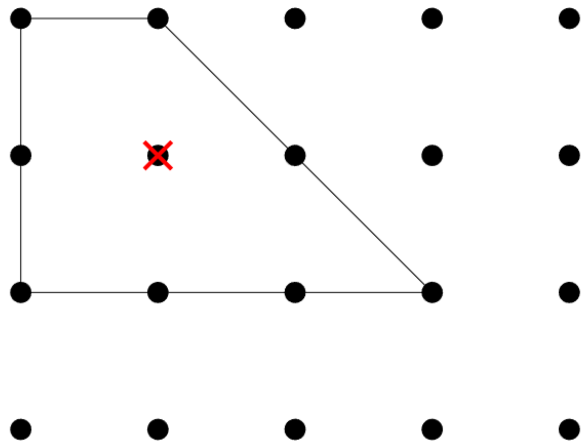

Figure 3: A Lattice Polytope

It can be seen that any bounded convex polytope on the integer lattice can be uniquely defined with respect to its vertices. The list $\{(-1,-1),(-1,1),(0,1),(2,-1)\}$ (which is the list of vertices of our example) can only refer to one polytope, regardless of order, as maintaining convexity allows only one arrangement.

2.3 Distance on the Integer Lattice

The distance between two points on the integer lattice is dependent on the direction it is being measured with respect to. We therefore introduce the notation:

$$\text{distance}_u(N_1, N_2)$$

where u is a vector represented by a point on the integer lattice. We require u to be primitive for ease of calculation (in this case this means (u_1, u_2) has $\mathbf{gcd}(u_1, u_2) = 1$, or if one of u_1 or $u_2 = 0$ then the other must be ± 1). As such, (1,0), (2,1) and (4,7) are all primitive vectors. (2,0) and (2,2) are not primitive.

Distance$_u(x, y)$ is a pseudometric, assigning equivalence classes along the vectors perpendicular to u which intersect a lattice point. Within this pseudometric, points in the same equivalence class have distance$_u$= 0 from one another, and a step from one equivalence class to an adjacent one is considered to have distance$_u$= 1.

Consider the following example. We have points $L_1 = (-1,-1)$ and $L_2 = (0,1)$ and we want to know distance$_{u_1}(L_1, L_2)$, where $u_1 = (0, 1)$. Lines orthogonal to u can be drawn, such that every lattice point has a line intersecting it representing the equivalence class it belongs to. By examining these equivalence classes, in particular the number of equivalence classes between the two points we are interested in, we can find distance$_u$ between these points. This approach to the measurement of distance$_{u_1}$ can be seen on Figure 4.

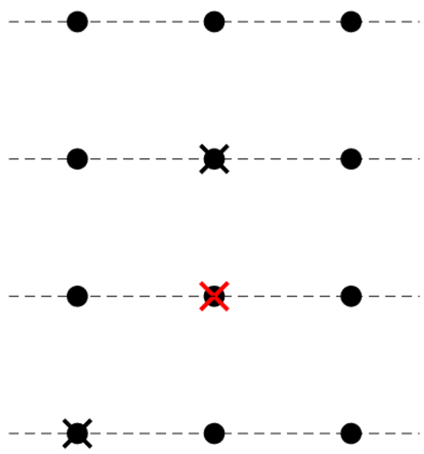

Figure 4: Diagram showing the equivalence classes for finding distance$_{u_1}$

In this example, the distance between L_1 and L_2 with respect to u_1 can be seen to be 2, as there is one line between them.

We also examine a second example, where L_1 and L_2 are the same but we use $u_2 = (1, 2)$ instead of u_1. Figure 5 shows the equivalence classes, and the distance$_{u_2}$ between L_1 and L_2 can be seen to be distance$_{u_2}((-1, -1), (0, 1))$=5.

2.4 Directional Evaluation of a Point

We define $\Pi_u(x) : N \to \mathbb{Z}$ as a function measuring the distance between the origin and point x. As an example, considering the same points discussed for Figures 4 and 5 for $u = (1, 2)$. Here, $\Pi_u(-1, -1) = -3$ and $\Pi_u(0, 1) = 2$. This is also shown in Figure 5. Unlike distance$_u(a, b)$ discussed before, this directional evaluation also allows negative results. The choice of vector is also important in this case. If $u = (-1, -2)$ was chosen instead, the results would be $\Pi_u(-1, -1) = +3$ and $\Pi_u(0, 1) = -2$.

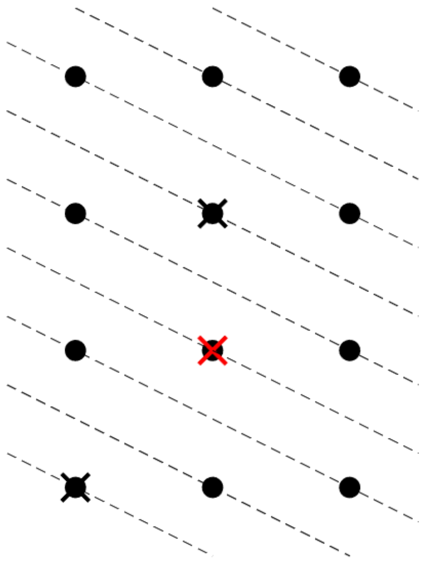

Figure 5: A Diagram showing the equivalence classes for u_2

2.5 Primitive Distance

Consider a vector $r \in \mathbb{Q}^n$. This vector r is parallel to exactly 2 primitive vectors t_- and t_+ with $|t_-| = |t_+|$. Thus $|t_+| = d|r|$ for some $d \in Q$. d is then the primitive distance of r. This means that (-1,0) has primitive distance 1, (2,0) has primitive distance 2 and $(\frac{1}{2}, \frac{1}{2})$ has primitive distance $\frac{1}{2}$.

2.6 Width of a Polytope

The width of a polytope P with respect to direction u is defined as follows:

$$\text{Width}_u(P) = \max_{p_1,p_2}\left(\text{distance}_u(p_1, p_2)\right)$$

with $p_1, p_2 \in P$.

From this the width of P is defined to be:

$$\text{Width}(P) = \min_u\left(\text{Width}_u(P)\right)$$

Due to having to evaluate the width$_u$ of the polytope with respect to every direction, finding width(P) is not as simple as finding the distance between points or the width$_u$

with respect to a given u. Some further examples of finding the width will be given later in Section 3.

2.7 Dual Lattice

So far, we have discussed the lattice $N := \mathbb{Z}^n$. We now introduce a second lattice $N^* := \mathbb{Z}^n$, and define it to be the "dual lattice" of N.

The concept of the dual lattice has a lot of uses. For example, there exists a relationship between lattice polytopes, and toric varieties (Trevisan 2007). As such, by finding lattice polytopes, we can take note of some toric varieties that will exist, and look for relationships between their properties. These relationships could lead to results that could have application in string theory (Cho, Kim, Lee & Park 2019). String theory is a theory within the field of fundamental physics, a branch of theoretical physics.

2.8 Dual Polytope

The concept of multiple definitions of a polytope P and of distance$_u$ on the integer lattice have been defined, and these allow us to define the Dual Polytope P^*. The properties of P^* are entirely derived from P, with P^* bounded by hyperplanes defined by the vertices of P embedded in the original lattice.

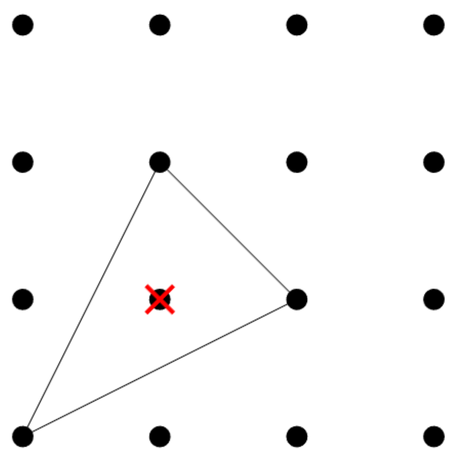

Figure 6: Polytope P

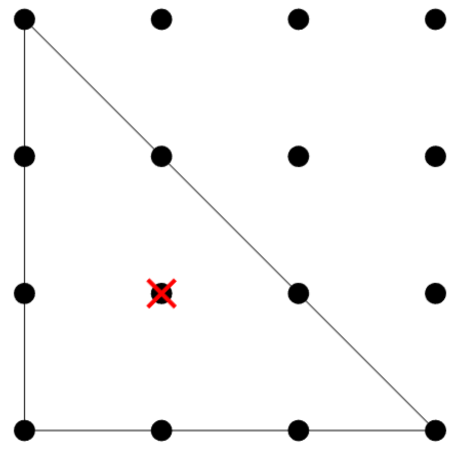

Figure 7: P* the dual of P

Figure 6 shows a polytope $P \in N$. Figure 7 shows its dual polytope $P^* \in N^*$. P^*. The dual polytope P^* of P is constructed as follows:

1. Primitive vectors u_n are constructed from each of the vertices v_n of P to the origin. We also take note of d_n, the primitive distance between v_n and 0.

2. We begin with these vectors, and construct a hyperplane H_n orthogonal to each of them, such that $\text{distance}_{u_n}(H_n, 0) = \frac{1}{d_n}$ and such that H_n is in the same direction as u_n.

3. The hyperplanes so generated are now the bounding hyperplanes of a new polytope, the dual polytope P^* on N^*.

Referring back to Section 2.2, the reason we define polytopes in two distinct ways will now have become clear.

Abstractly, this is rather difficult to follow. As such, I will now carefully go through three examples, in order to illustrate what this means in practice.

Examining Figure 6, with the intention of finding P^* given P, we can construct 3 vectors on N^*, $[u_1, u_2, u_3] = [(-1, 0), (0, -1), (1, 1)]$. These correspond to the vertices $[v_1, v_2, v_3] = [(1, 0), (0, 1), (-1, -1)]$ of P on N. As each of the u_i is a primitive vector each of the primitive distances will be 1, $d_1 = d_2 = d_3 = 1$.

First, $(u_1) = (-1, 0)$ is then used to construct a hyperplane on N^*. The hyperplane orthogonal to this goes through points of $\text{distance}_{u_1} = \frac{1}{d_1} = 1$. As such, the first bounding hyperplane on N^* is [(-1,0),(-1,0)].

Similarly, $(u_2) = (0, -1)$ is then used to construct a second hyperplane on N^*. The hyperplane orthogonal to this goes through points of distance$_{u_2} = \frac{1}{d_2} = 1$. As such, the second bounding hyperplane on N^* is [(0,-1),(0,-1)].

Finally, $(u_3) = (1,1)$ is then used to construct a hyperplane on N^*. The hyperplane orthogonal to this goes through points of distance$_{u_3} = \frac{1}{d_3} = 1$. As such, the final bounding hyperplane on N^* is [(1,1),(1,0)]. This results in the polytope seen in Figure 7.

Examining these bounding hyperplanes, we find $[v_1, v_2, v_3]$=$[(2,-1),(-1,2),(-1,-1)]$, the vertices of P^*. Constructing the dual of P^* with the intention of returning to P, we then generate a new set of $[u_1, u_2, u_3] = [(-2,1),(1,-2),(1,1)]$. The hyperplane H_1 on N corresponding to u_1 is [(-2,1),(0,1)], u_2 gives us [(1,-2),(1,0)] and, as before, u_3 corresponds to a bounding line of [(1,1),(1,0)].

Th polytope bounded by these hyperplanes is our original polytope P. As both P and P^* are lattice polytopes, they are both reflexive, a concept which will be discussed further in Section 2.11.

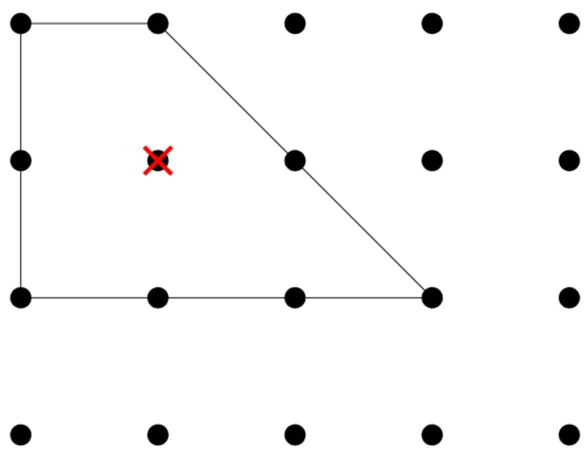

Figure 8: A Lattice Polytope Q

We examine a second example, illustrated as polytope Q in Figure 8. This has vertices $[v_1, v_2, v_3, v_4] = [(0,1),(-1,1),(-1,-1),(2,-1)]$. With the intention of finding it's dual Q^*, we construct 4 vectors: $[u_1, u_2, u_3, u_4] = [(0,-1),(1,-1),(1,1),(-2,1)]$. These each have $d_i = 1$.

As above, $(u_1) = (0,-1)$ is used to construct H_1. H_1 is orthogonal to u_1 and goes through points of distance$_{u_1} = \frac{1}{d_1}$. As such, H_1 is $[(0,-1)$,(0,-1)] on N^*.

Likewise, $(u_2) = (1,-1)$ is used to construct H_2. H_2 is orthogonal to u_2, and goes through points of distance$_{u_2} = \frac{1}{d_2} = 1$. As such, H_2 is [$(1, -1)$,(1,0)] on N^*.

Similarly $(u_3) = (1, 1)$ is used to construct H_3 on N^*. H_3 is orthogonal to u_3, and goes through points of distance$_{u_3} = \frac{1}{d_3} = 1$. As such, H_3 is [$(1, 1)$,(1,0)] on N^*.

Finally, $(u_4) = (-2, 1)$ is used to construct H_4. H_4 is orthogonal to u_4, and goes through points of distance$_{u_4} = 1$. As such, H_4 is [$(-2, 1)$,(0,1)] on $N*$. This results in the polytope seen in Figure 9.

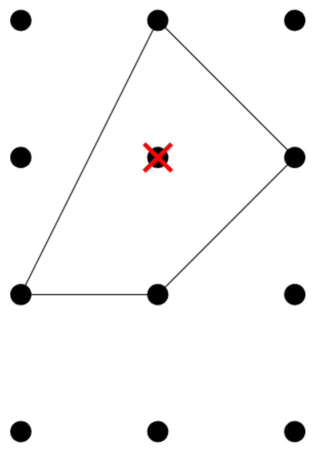

Figure 9: Q^* the dual of Q

The list of vertices of this dual polytope Q*: $[v_1, v_2, v_3, v_4] = [(1, 0), (0, 1), (-1, -1), (0, -1)]$. We can then generate in turn a new set of $[u_1, u_2, u_3, u_4] = [((-1, 0), (0, -1), (1, 1), (0, 1)]$ with the intention of finding $(Q^*)^*$.

As above: u_1 generates [$(-1, 0)$,(-1,0)] on N; u_2 generates [(0,-1),(0,-1)] on N; u_3 generates [(1,1),(1,0)] on N, and; u_4 generates [(0,1),(0,1)] on N. As expected, this is Q. As both Q and $Q*$ are lattice polytopes, they are also reflexive polytopes.

A final example, polytope R, is shown in Figure 10. Again, we intend to find R^* given R. Here we can see there are vertices at $(1, 0), (0, 2), (-1, -1)$. As R is similar to P, many of the steps are not reproduced here. Following our earlier procedure allows us to generate $[(u_1), (u_2), (u_3)] = [(-1, 0), (0, -1), (1, 1)]$. As for P, $d_1 = 1$ and $d_3 = 1$. However, $(2, 0)$ has primitive distance 2 (because (2,0) is not primitive, and is 2 times the magnitude of the primitive vector (1,0) in the same direction), so $d_2 = 2$. This is because $(1, 0)$ is a primitive vector, and $(2, 0)$ has magnitude twice as large as $(1, 0)$.

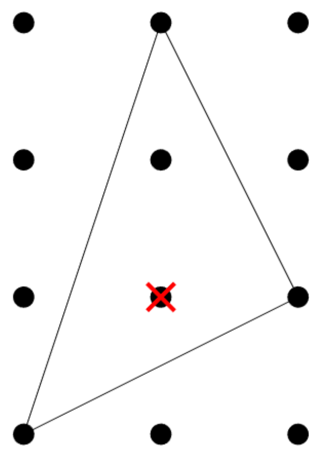

Figure 10: R

The bounding lines generated by u_1 and u_3 are the same, as are [(1,1),(1,0)] and [(-1,0),(-1,0)]. Only u_2 needs to be separately considered.

In this case, $(u_2) = (0, -1)$ is then used to construct H_2 on N^*. H_2 is orthogonal to u_2, and goes through points of distance $\frac{1}{2}$ from the origin. As such, H_2 is [$(0, -1)$,$(0, -\frac{1}{2})$] on N^*.

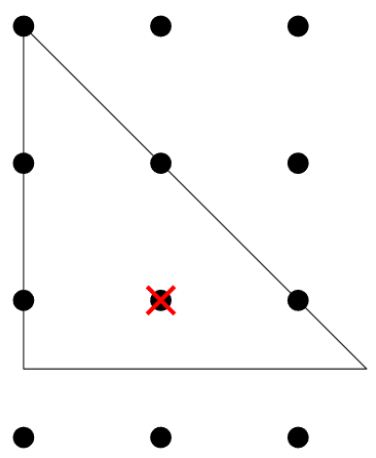

Figure 11: R^*_3

Observing Figure 11, it can be seen that this doesn't satisfy the definition of a lattice polytope. That is to say, not all of its vertices lay on the integer lattice. The list of vertices of this dual polytope R*: (-1,2),(-1,-$\frac{1}{2}$),($\frac{3}{2}$,-$\frac{1}{2}$) then generate in turn a new set of $[u_1, u_2, u_3] = [(1, -2), (2, 1), (-2, 1)]$. Using the same technique again: u_1 has $d_1 = 1$ and generates [$(-1, 2)$,(1,0)] on N, u_2 has $d_2 = \frac{1}{2}$ and generates [(2,1),(1,0)] on N and finally,

u_3 has $d_3 = \frac{1}{2}$ and generates [(-3,1),(-1,-1)] on N. This is R_3 (Figure 10).

As mentioned, R^* is not a lattice polytope. As such, neither R or R^* are reflexive lattice polytopes. Yet we still have that $(R^*)^* = R$.

2.9 Polytope Volume on a Lattice

The volume of a polytope is defined to be the number of empty triangles in the interior of the polytope. An empty triangle T is one that has vertices on three lattice points N_1, N_2, N_3, such that

$$\{N_1, N_2, N_3\} \in T \cup P.$$

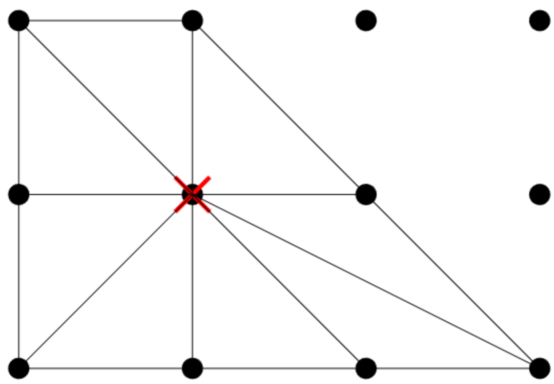

Figure 12: A demonstration of Volume on a Polytope Q

Continuing with polytope Q, through the following separation into empty triangles as seen in Figure 12, this example can be seen to have volume(T)= 8.

The second example shown in Figure 13 can be seen to have volume(T)= 3.

For a polytope K, we have:

K is bounded $\iff$ volume(K) is finite.

2.10 Isomorphism Classes of Polytopes

A change in the choice of basis vectors maintains the same polytope, though its appearance changes. The polytopes in Figures 14 and 15 below is an example of the effect of changing our original basis vectors from $\{(1,0),(0,1)\}$ to $\{(1,0),(1,1)\}$

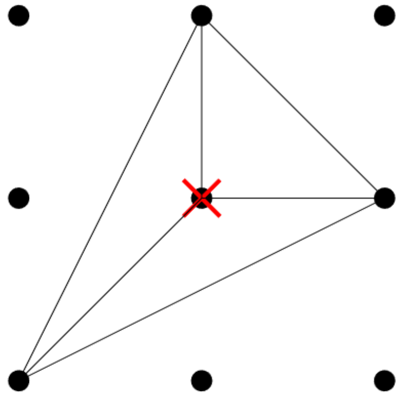

Figure 13: A second demonstration of Volume on a Polytope

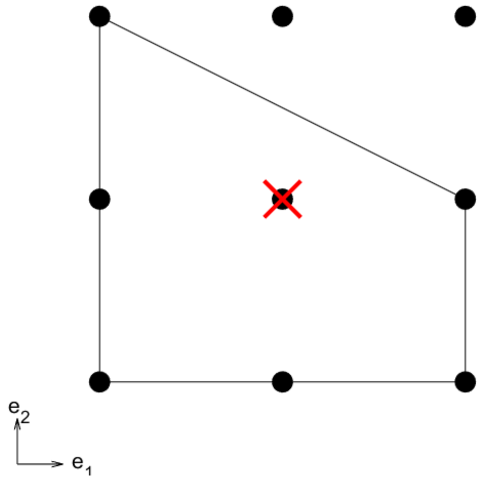

Figure 14: A polytope P

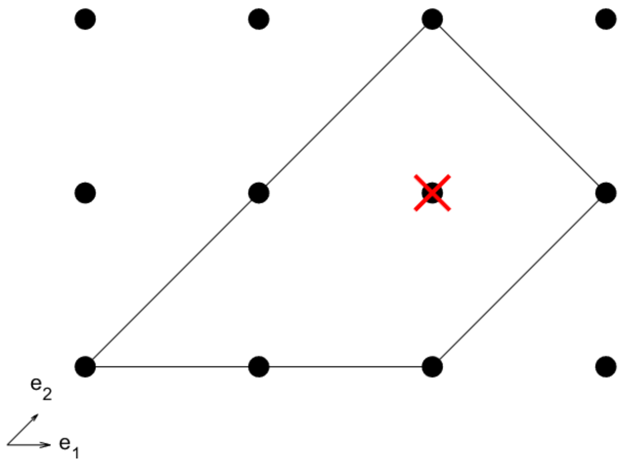

Figure 15: The same polytope P, after a change of basis

Notably, the number and relative location of lattice points on the boundary of the polytope, and the volume of the polytope remain the same.

The relationship between the vertices of the polytope also remain the same under this change of basis. This can be most easily seen in Figure 16. Which shows polytope

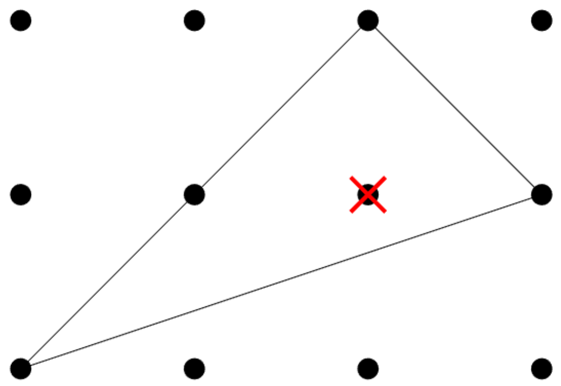

Figure 16: Q

Q. This polytope has 3 vertices, $[v_1, v_2, v_3] = [(1, 0), (0, 1), (-2, -1)]$. As these all have integer co-ordinates, it is straightforward to show there exists some non-zero sum of these vectors which returns to the origin. This is also true of any other lattice polytope with the origin in the interior. In this case, the sum would be [1,1,2], and it can easily be seen that $1(-2, 1) + 1(0, 1) + 2(1, 0) = \mathbf{0}$. Polytope R (Figure 17 also has the same sum of vertices

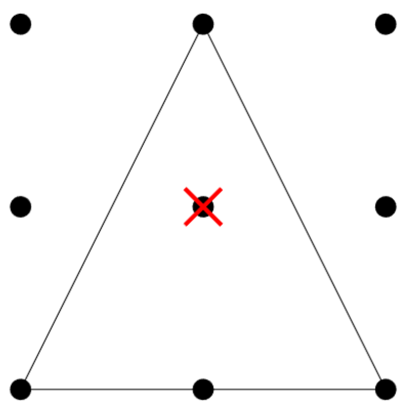

Figure 17: The same polytope R which is isomorphic to Q

which add to 0 with weights of [1,1,2]. This resolves to $1(1, -1) + 1(-1, -1) + 2(0, 1) = \mathbf{0}$. It can also be seen that both Q and R have 4 lattice points on their respective boundaries.

2.11 Reflexive Polytopes

We have discussed polytopes on N, and defined them with respect to their vertices and bounding hyperplanes. We have universally that $(P^*)^* = P$.

$$P \text{ is reflexive } \iff P^* \in N^*$$

That is to say, for a lattice polytope P, if P^* is a lattice polytope, both P and P^* are reflexive. Returning to the examples discussed in Section 2.8, in each case the dual polytope P^* and Q^* were also lattice polytopes. As such, they are both reflexive (as was asserted).

2.12 Proof of some Properties of Reflexive Polytopes

Every reflexive lattice polytope has all vertices at primitive vectors. This can be seen through the following steps. Assume towards a contradiction that there exists a reflexive lattice polytope P on N which has a vertex v located at a non-primitive location. In this case v has primitive distance $d \neq 1$. As P is a lattice polytope, we also have that $d < 1$. Accordingly, the hyperplane H that is then constructed on N^* has distance from the origin $\neq 1$. However, as discussed in Section 2.8, all lattice points have integer distance from the origin. Accordingly, H does not intersect any lattice points, and some of the vertices of P^* are not lattice points. As such, P^* is not a lattice polytope, which implies neither P nor P^* are reflexive. This is a contradiction, and as such, there do not exist any reflexive lattice polytopes with a vertex at a non-primitive point.

It can further be shown that all reflexive lattice polytopes have all faces at distance 1 as follows. Assume towards a contradiction there exists a reflexive lattice polytope P bounded a hyperplane H with distance $\neq 1$ from the origin. This implies there exists a polytope P^* with a vertex point at a non-primitive location. We have already shown that this is not the case for reflexive lattice polytopes, so this is a contradiction. As such, all reflexive lattice polytopes have all faces at distance 1 from the origin.

2.13 Isomorphism Classes of Reflexive Polytopes in Dimension 2

The collection of polytopes shown in Figure 18 organises the exhaustive list of the isomorphism classes of 2-d polytopes. Each class is represented by a polytope, typically the simplest polytope in the class. On this diagram, the number of sides increase, beginning at 3 on the top row and ending at 6 on the final row. This diagram has the dual P^* of each polytope P in the isomorphism class that appears in the mirrored position. In each row, there is a self-dual polytope (such that $P = P^*$). It can be seen that the number of lattice points on the boundary of P and P^* sums to 12. Accordingly, any 2-d polytope will be self-dual if and only if it has 6 lattice points on its boundary. The diagram is also arranged such that the number of edge points increases from left to right.

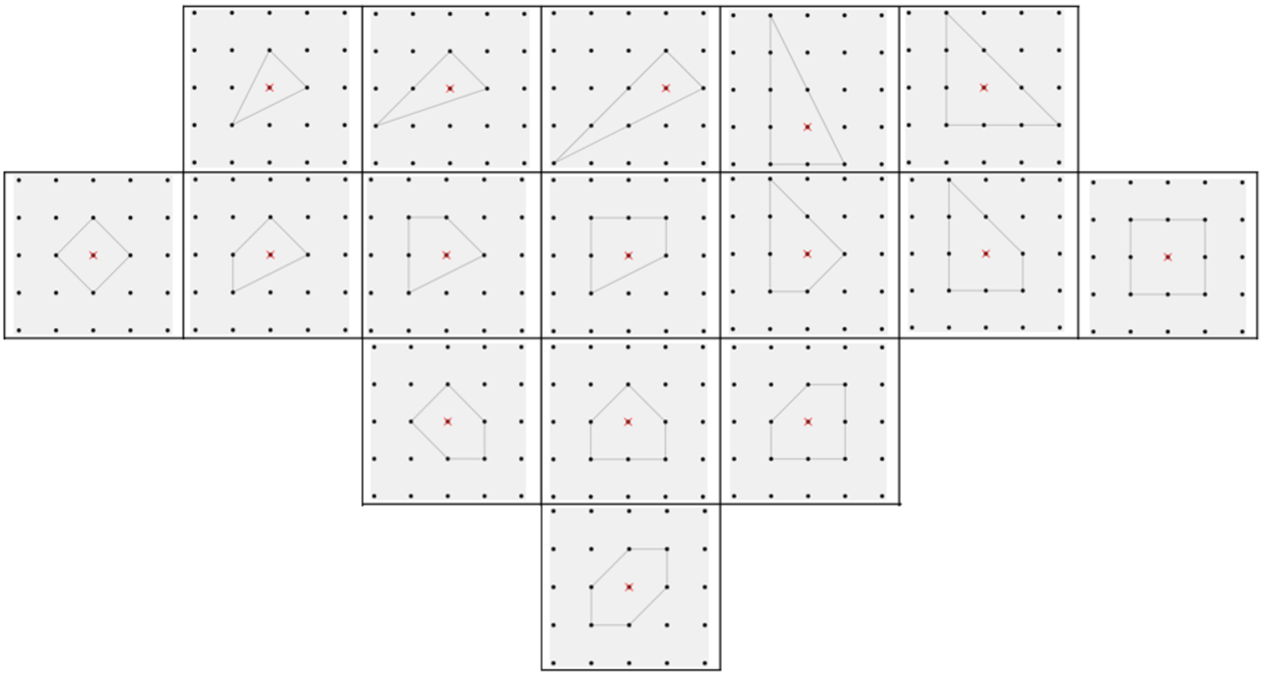

Figure 18: An organised representation of the 16 2-D Reflexive Polytopes

A few examples of relations between the vertices of polytopes have already been demonstrated, in particular the 2 leftmost polytopes in the top row. In each case, the relations between vertices listed are sufficent to completely describe the isomorphism class. For example, the second polytope in row 2 has relations $[1, 0, 1, 0]$ and $[1, 1, 0, 1]$. It can be seen that $(1, 0) + (-1, 0) = \mathbf{0}$ and $(1, 0) + (0, 1) + (-1, -1) = \mathbf{0}$. There exist many other combinations of variables which sum to the origin, for example $2(1, 0)+(0, 1)+(-1, 0)+(-1, -1)$, represented by $[2,1,1,1]$. However, this relation is not linearly independent to the two rows listed, and so does not require a separate row.

Figure 19 shows the weights of the vertices of the isomorphism classes for all polytopes shown in Figure 19.

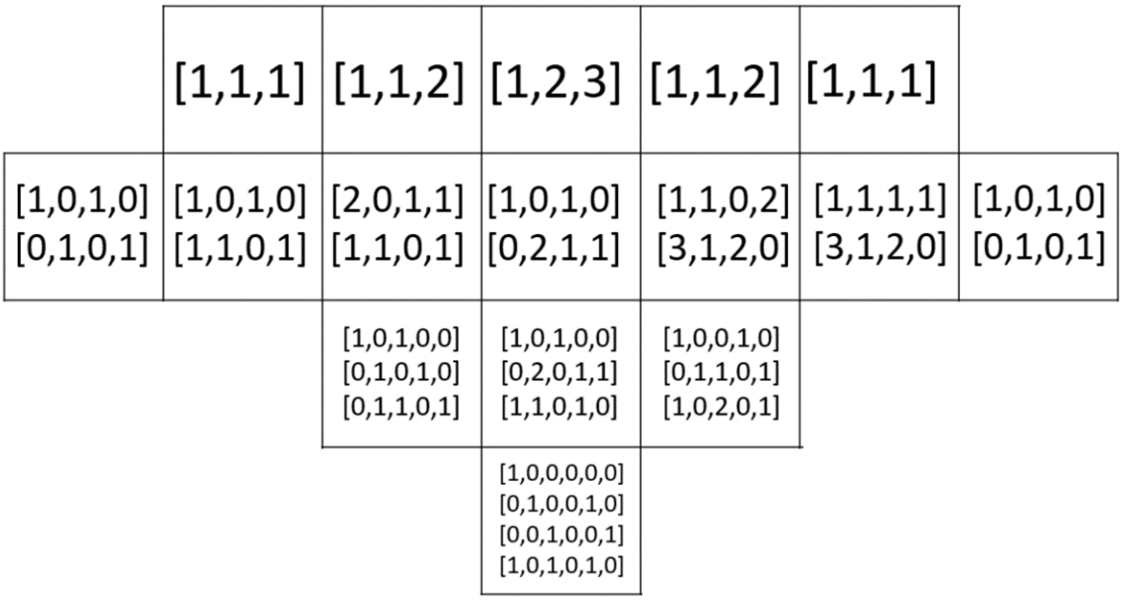

Figure 19: A Diagram Displaying the Weights of the Vertices of the Polytopes in the Isomorphism Classes

2.14 Simplices

As discussed, one method of examining polytopes is by looking at the relationship between their vertices. Simplices are polytopes with one more point than the dimension of the space they are embedded in. This results in triangular facets. For example, a triangle in 2 dimensions or a tetrahedron in 3, as seen in Figure 20.

2.15 Sublattices

A sublattice S of a lattice N, is such that $S \subset N$ is a lattice with the same operations as N. The sublattice generated by a set of points $s_1, s_2, \cdots \in N$ is the lattice generated if these points are used as the bases for a new lattice. When discussing the sublattice generated by a polytope P, we are considering the sublattice generated by a set of vertices

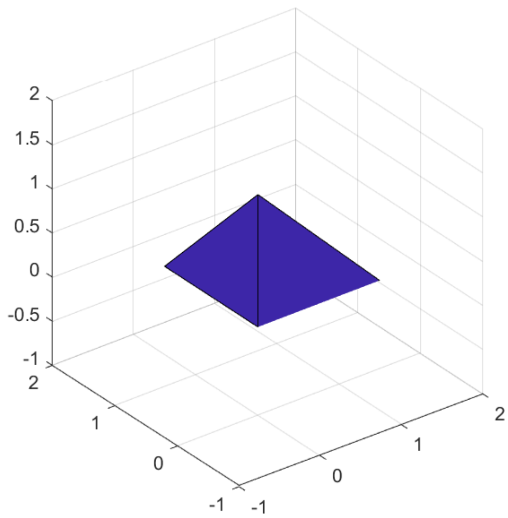

Figure 20: A 3-d simplex: A Tetrahedron

of P: $v_1, v_2, \ldots$.

For example, consider the polytope with vertices at $(1, 0), (0, 1)$ and $(-1, -1)$. It is easy to see how in this case, the sublattice S will be the lattice N, as both of the basis vectors are included. In any situation where there exists a sum of the vertices equal to each of the bases of N then $S = N$.

For an example of a polytope in which there does not exist a sum of vertices equal to (1,0) and (0,1), consider $P = \text{conv}((-1, -1), (-1, 2), (2, -1))$. The sublattice generated by P in this case is the lattice generated by the bases $(1, 1)$ and $(2, -1)$. This can be seen in Figure 21.

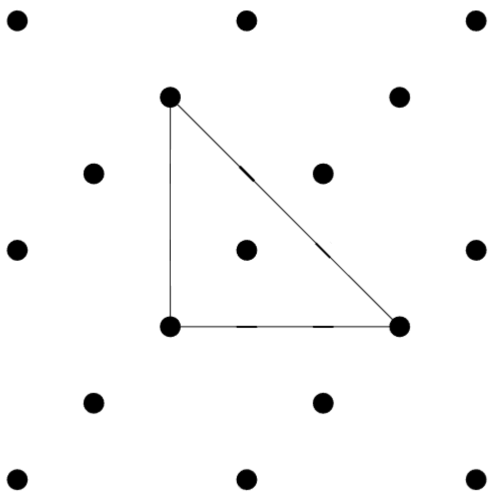

Figure 21: The Sublattice generated by P

It can be seen that the points of this sublattice are all possible to reach through integer multiplication and addition of the vertices of P.

3 Examining the Width of Polytopes

In this section some proofs will be demonstrated of some results on N. We shall also examine the width of some polytopes.

Throughout this section, we will specifically be examining simplices (that is polytopes in dimension n with number of vertices $n+1$). This reduces the number of polytopes we are considering, which will later be important for computational reasons. However, as there exist simplices in this space, our results remain significant.

3.1 Distance Invariance Under Translation

Demonstrating that the a change of origin has no effect on distance on the lattice is straightforward, and beginning with the equation of distance for two points translated away from the origin by v as follows:

$|\Pi_u(p_1 + v) - \Pi_u(p_2 + v)|$

$=|\Pi_u(p_1) + \Pi_u(v) - \Pi_u(p_2) - \Pi_u(v)|$

$=|\Pi_u(p_1) - \Pi_u(p_2)|.$

This is the equation for $\text{distance}_u(p_1, p_2)$, and as such, the pseudometric: $\text{distance}_u(x, y)$ is invariant under translation.

3.2 Finding Upper Bounds on Width through Examination

Let us return to our first example of a polytope.

Examining the first of these, P (Figure 22) with respect to $u = (1, 0)$, we can see that the two points furthest apart are $(1, 0)$ and $(-1, -1)$. These can be evaluated through $\Pi_{u_1}(v)$ as having value $+1$ and -1 respectively. As such, $\text{width}_{u_1}(P) \leq 2$.

Similarly, in P^* (Figure 23) the furthest two points with respect to u are $(2, -1)$ and $(-1, -1)$. Evaluation $\Pi_{u_1}(v)$ gives $+2$ and -1. Thus $\text{width}_{u_1}(P^*) \leq 3$.

These calculations provide an upper bound for the width of these polytopes. This is because the width is taken as the minimum of width_u for all u, $\text{width}(P) \leq \text{width}_u(P)$.

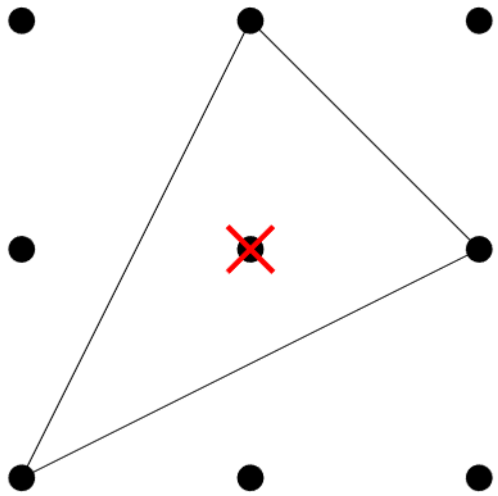

Figure 22: Polytope P

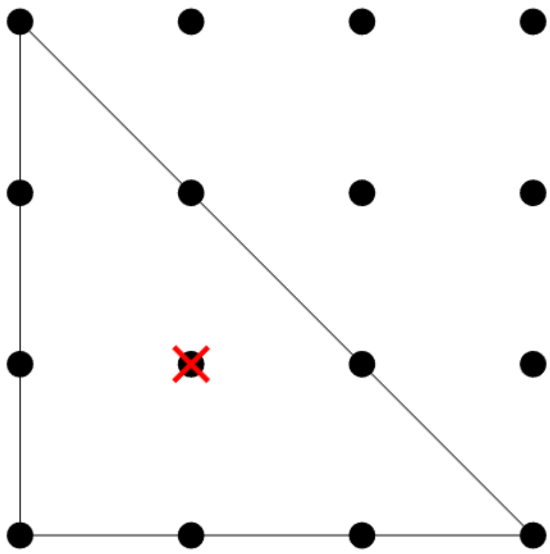

Figure 23: P^*

This then gives width(P)$\leq$2 and width($P*$)$\leq$3. We can combine these with other results to be get stronger statements about the widths of these polytopes.

3.3 Proof of Interior Point Theorem

Theorem: Let Q be a polytope with the origin $\mathbf{0} \in$ interior(Q) and let $u \in N$. Then $\text{width}_u(Q) \leq 2$.

Throughout we assume that the interior point of our polytope is the origin, however from our proof of distance invariance this can be any point without loss of generality.

Proof: $\Pi_u : N \to \mathbb{Z}$, a linear function mapping points of the lattice to a value. As such, $\text{width}_u(Q) \in \mathbb{Z}$

For any choice of u, $\Pi_u(\mathbf{0}) = 0$. We then assume towards a contradiction that there are no points with $\Pi_u(Q) > 0$.

We have that $\mathbf{0} \in Q$ and so either $\mathbf{0} = v$ (that is, the origin is a vertex point of Q), or there exist points v_1, v_2 with $\Pi_u(v_1) = \Pi_u(v_2) = 0$. Examples of this can be seen below in Figures 24 and 25:

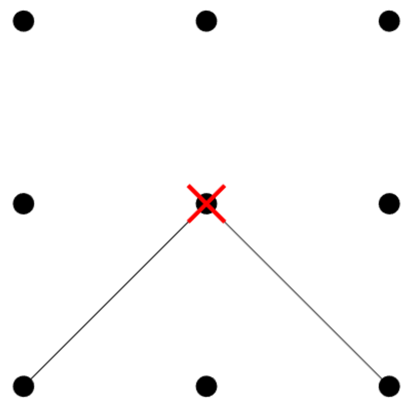

Figure 24: One possible polytope with vertices (a, b) where $b \leq 0$

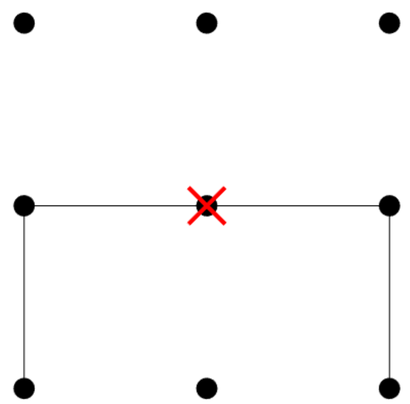

Figure 25: Another possible polytope with vertices (a, b) where $b \leq 0$

Clearly in either case, $0 \notin$ interior(Q) and so one of our initial requirements is broken, giving a contradiction. As such, there exists a point v_+ with $\Pi_u(v_+) \geq 0$.

A similar technique can be used to demonstrate the existence of v_-, with $\Pi_u(v_-) \leq 0$.

As we also have that $\Pi_u(v_+) \in \mathbb{Z}$ and $\Pi_u(v_-) \in \mathbb{Z}, \Pi_u(v_+) \geq 1$ and $\Pi_u(v_-) \leq -1$ we can then formulate that $\text{width}_u(Q) \geq 2, \forall u$.This is the original result we set out to prove.

3.4 Finding the width of P

Using the result from section 3.3 thus have that P (Figure 22) has width 2, as there is an upper and lower bound on its width, both of 2 ($2 \leq \text{width}(P) \leq 2$).

3.5 Finding the width of P*

We already have an upper bound of 3 for P^*, given by examining for $u = (1, 0)$, resulting in $\text{width}_u = 3$.

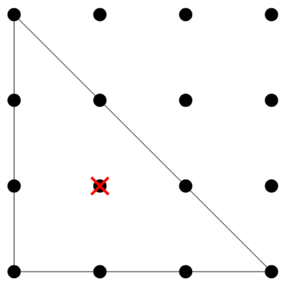

Figure 26: P^*

We also have that there exists an interior point and so there is a lower bound of 2. As such, $2 \leq \text{width}(P^*) \leq 3$. It is easy to see without finding u that $\text{width}_u(P^*) = 3$; however there is no way of finding this width using the methods discussed so far.

Considering another approach, by observing that $(2, -1) = (-1, -1) + 3 \cdot (1, 0)$, and that for primitive vector u, such that $u \neq (0, 1)$, $\text{distance}_u((0, 0), (1, 0)) = 1$. Thus for any $u \neq (0, 1)$, $\text{distance}_u(v_1, v_2) = 3$ by distance invariance. Additionally, in the case where $u = (0, 1)$, it can easily be seen that $\text{width}_u(P^*) = 3$.

As such, we have $\text{width}_u(P^*) = 3$ for any choice of u, and therefore $\text{width}(P^*) = 3$.

4 Projective Varieties

4.1 Projective Space

A *Projective Space* $\mathbb{P}^n$ is the set of all one-dimensional subspaces of the vector space $\mathbb{C}^n$. That is, $\mathbb{P}^n$ is the set of all complex lines through the origin.

We can define

$$\mathbb{P}^n = \mathbb{C}^{n+1} \backslash \{0\} / \sim$$

where $\sim$ denotes the equivalence relation of points lying on the same line:

$$(x_0, \ldots, x_n) \sim \ (y_0, \ldots, y_n)$$

if and only if there exists λ in $\mathbb{C}\backslash\{0\}$ such that $(y_0, \ldots, y_n) = (\lambda x_0, \ldots, \lambda x_n) | \lambda \in \mathbb{C}\backslash\{0\}$. A *point* in $\mathbb{P}^n$ is an equivalence class; i.e. a line through the origin. We denote these classes by

$$(x_0, \ldots, x_n) = \{(\lambda x_0, \ldots, \lambda x_n)\} | \lambda \in \mathbb{C}\backslash\{0\}$$

.

$(0 : \cdots : 0)$ is not a point in $\mathbb{P}^n$, since $(0, \ldots, 0)$ is not a member of any of the equivalence classes (Fulton 1969).

4.2 Weighted Projective Space

The definition of a weighted projective space is very similar to that of a projective space. We first introduce the concept of weights. Let $a = (a_0, \ldots, a_n)$. We now define the equivalence relation $\sim$:

$$(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n)$$

if and only if there exists $\lambda \in \mathbb{C}$ such that

$$(y_0, \ldots, y_n) = (\lambda^{a_0} x_0, \ldots, \lambda^{a_n} x_n)$$

We call $a = (a_0, a_1, \ldots, a_n)$ the *weights*.

Accordingly, a *weighted projective space* is

$$\mathbb{P}(a_0, a_1, \ldots, a_n) = \frac{\mathbb{C}^{n+1} \backslash \{0\}}{\sim}$$

It can be seen that the equivalence relation $\sim$ is invariant under a linear change of the weights. That is to say, for $k \in \mathbb{R}$, $\mathbb{P}(a_0, a_1, \ldots, a_n) = \mathbb{P}(ka_0, ka_1, \ldots, ka_n)$. As such, we choose the set of weights to be $(a_0, a_1, \ldots, a_n)$, such that $\gcd(a_0, a_1, \ldots, a_n) = 1$ (Fulton 1969).

4.3 Simplex P

Let P be an n-dimensional lattice simplex containing the origin in its strict interior, and such that the vertices $v_0, \ldots, v_n$ are all primitive. Let $a_0, a_1, \ldots$ be the unique sequence of coprime positive integers such that

$$a_0 v_0 + a_1 v_1 + \cdots + a_n v_n = 0$$

It can be seen that, without loss of generality, we can relabel the vertices so that $a_0 \leq a_1 \leq \cdots \leq a_n$, a property we will later depend upon.

4.4 Fake Weighted Projective Space

Via Toric Geometry, polytope P corresponds to a *fake* weighted projective space $\frac{\mathbb{P}(a_0, a_1, \ldots, a_n)}{G}$. Here, $\mathbb{P}(a_0, a_1, \ldots, a_n)$ is weighted projective space, and G is a group whose order is equal to the index of the sublattice $v_0\mathbb{Z} + v_1\mathbb{Z} + \ldots v_n\mathbb{Z}$ generated by the vertices of P (Buczy'nska 2008).

In particular, P corresponds to a weighted projective space if and only if the vertices of P $\mathbb{Z}$-generate the lattice.

4.5 Gorenstein Weighted Projective Space

A weighted projective space is *Gorenstein* if and only if each of the a_i divides f, where f is the sum of the weights. In such a space, the corresponding simplex P is reflexive; that is the simplex P^* is also a lattice simplex. The converse of this statement is not true: just because P is a reflexive simplex, this does not mean it corresponds to a Gorenstein weighted projective space. In general, a reflexive simplex corresponds to a Gorenstein *fake* weighted projective space. Here all weights are assumed to be well formed. (Kasprzyk 2013)

4.6 Well Formed Weighted Projective Space

A Weighted Projective Space $\mathbb{P}(a_0, a_1, \ldots, a_n)$ is considered *well formed* if $\gcd(a_0, a_1, ..., a_n) = 1$ and, if we omit any a_i, the gcd of the remaining n weights is also 1. Whenever we discuss *fake* weighted projective space, we take it as part of the definition that the weights are well-formed (Kasprzyk 2013).

4.7 Recovering P from a Weighted Projective Space with $a_0 = 1$

Given a weighted projective space $\mathbb{P}(a_0, a_1, \ldots, a_n)$, recovering P is algorithmic. P is defined by a set of v_i of which it is the convex hull. These v_i will therefore satisfy the condition $a_0v_0 + a_1v_1 + \cdots + a_nv_n = 0$.

This is straightforward in the case where $a_0 = 1$ (remembering that we have relabelled in order to have a_0 the smallest of the a_i). By setting each of the remaining points as a basis of the lattice (that is $v_1 = (1, 0, \ldots, 0), v_n = (0, 0, \ldots, 1)$), v_0 will have co-ordinates $(-a_1, -a_2, \ldots, -a_n)$.

In the case where $a_0 \neq 1$, finding these co-ordinates is more complicated, but can be done through application of Cramer's rule.

4.8 Cramer's Rule

Consider a system of n linear equations for n unknowns, represented in matrix multiplication form as follows:

$$Ax = b$$

where the $n \times n$ matrix A has a nonzero determinant, and the vector $x = (x_1, \ldots, x_n)^T$ is the column vector of the variables. Then the theorem states that in this case, the system has a unique solution, whose individual values for the unknowns are given by:

$$x_i = \frac{\det(A_i)}{\det(A)} \qquad\qquad i = 1, \ldots, n$$

where A_i is the matrix formed by replacing the i^{th} column of A by the column vector b. (Cramer 1750)

4.9 Recovering P from a Weighted Projective Space with $a_0 = 1$

This can be found through the following process. We define for a polytope P=$(v_0, v_1, \ldots, v_n)$ that

$$q_i := |\det(v_0, \ldots, \hat{v}_i, \ldots, v_n)|.$$

From Cramer's rule and (Conrads 2001), we have $q_0 = 5$, $q_1 = 3$ and $q_2 = 2$. We make a guess for $v_0 = (1; 1)$. This let's us form the following equations:

$$q_1 = 3 = v_{21} - v_{22}$$

$$q_2 = 2 = v_{12} - v_{11}$$

which give us relationships between the co-ordinates of each of the points. This results in a recovered polytope as shown in Figure 27.

The blue cross denotes the point we have selected for v_0, and the lines represented the

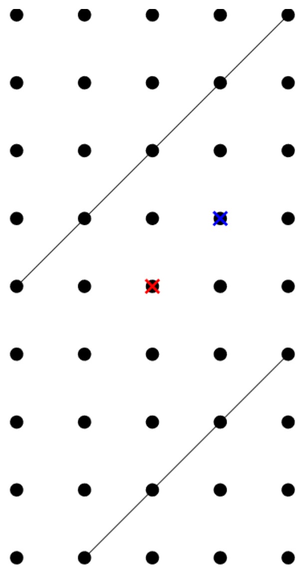

Figure 27: Possible Solutions for v_1 and v_2

set of points where the other two vertices could exist. We then use the equation

$$q_0 = 5 = \det(v_{11}, v_{21}; v_{12}, v_{22})$$

and substitute in to find

$$-3v_{11} - 2v_{22} = 11,$$

which further implies

$$2v_{21} = -5 - 3v_{11}.$$

Figure 28 shows the diagram with this line for one choice of v_1. It can be seen that choosing

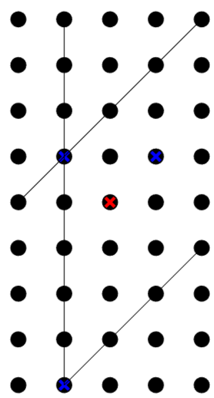

Figure 28: The Facet after a choice of v_1 has been made to the now required v_2

a value for either of the remaining vertices creates a completed polytope. Choosing vertices such that the facets in $H = (u, (a, b))$ are distance$_u$=1 from the origin, so as to prevent the resulting polytope from not being reflexive helps us in making a choice of v_1. Only one choice for v_1 results in an edge of distance 1, which is $v_1 = (-1, 1)$. This in turn results in the final vertex $v_2 = (-1, -4)$.

Therefore this is the polytope with $v_0, v_1, v_2 = (1, 1), (-1, 1), (-1, -4)$. That is, a triangle with both the origin and an additional lattice point in its interior. It can be seen that though v_0 and v_1 were chosen with the intention of making P a reflexive polytope, it is not one. This was expected as $\mathbb{P}(2, 3, 5)$ is not Gorenstein. Figure 29 shows the final polytope.

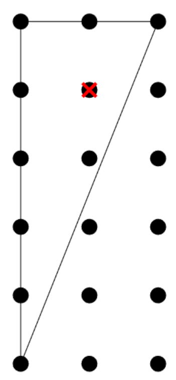

Figure 29: Polytope recovered from the Gorenstein weighted projective space $\mathbb{P}(2, 3, 5)$

4.10 Width Relations

The polytope P recovered from a Gorenstein fake weighted projective space is a reflexive lattice simplex with dimension n. A result found by Hall & Hoffscheier (2019) tells us that this polytope will have some properties based on the weights of the Gorenstein weighted projective space $\mathbb{P}(a_0, \ldots, a_n)$. For example, if for $i_1, \ldots \in I$ and, $j_1, \cdots \in J$ with $i_k \neq j_l$,

$$\sum_{i \in I} a_i = \sum_{j \in J} a_j$$

Then the polytope will have width 2.

Some further examples of width relations are as follows. Consider a set of weights $(a_0, \ldots, a_n)$. We say that there exists a width 2 relation on $\mathbb{P}(a_0, \ldots, a_n)$ if, for some $x_i \in X$, with $X = \{-1, 0, 1\}$, we have

$$\sum_{i \in I} x_i a_i = 0.$$

It is easy to see this is an equivalent statement to the width 2 relation given above.

Similarly, we say that there exists a width 3 relation on $\mathbb{P}(a_0, \ldots, a_n)$ if, for some $x_i \in X$, with $X = \{-1, 0, 1, 2\}$, we have

$$\sum_{i \in I} x_i a_i = 0$$

Ascertaining the existence of width relations for widths ≥ 4 is more difficult. For width 4, we must examine both $X = \{-1, 0, 1, 2, 3\}$ and $X = \{-2, -1, 0, 1, 2\}$.

5 Searching for Reflexive Lattice Simplices

One major area of interest is in the properties of Reflexive Lattice Simplices. This is related to our motivation, as discussed in Section 2.7. Reflexive Lattice Simplices are a very specific subset of lattice polytopes. We are particularly interested in reflexive lattice simplices with width greater than 2, and many aspects of our search use this particular interest heavily to reduce the search space.

5.1 Important Equations

We recall some equations from earlier sections, and introduce others which will be very important throughout the tree search based algorithm we develop in this section.

When the weighted projective space we are considering is Gorenstein (as ours are designed to be) we have for:

$$f := \sum_{i=0}^{n} a_i \tag{5.1}$$

that

$$a_i | f \quad \forall 0 \leq i \leq n \tag{5.2}$$

and also that without loss of generality we can state

$$a_0 \leq a_1 \leq \cdots \leq a_n \tag{5.3}$$

for a Gorenstein weighted projective space which generates a polytope P.

From Equation 5.1 and Equation 5.2

$$\text{for each } a_i \; \exists \; k_i \in \mathbb{N} \text{ such that } a_i k_i = f. \tag{5.4}$$

From Equation 5.3 and Equation 5.4

$$k_n \geq k_{n-1} \geq \cdots \geq k_0. \tag{5.5}$$

As discussed in Section 4.10, we also have that:

$$\text{If } \exists\ [i_0, i_1, \ldots, i_m], [j_0, j_1, \ldots, i_p] \text{ such that } \Sigma_{k=0}^{m} a_{i_k} = \Sigma_{q=0}^{p} a_{j_q} \text{ then width}(P) = 2 \tag{5.6}$$

From which we can state:

$$a_0 < a_1 < \cdots < a_n \tag{5.7}$$

and

$$k_n > k_{n-1} > \cdots > k_0. \tag{5.8}$$

5.2 Reflexive Lattice Simplices in Dimension 1

This is a trivial case which we examine purely for exhaustive reasons.

For a weighted projective space to be Gorenstein, it must as always obey the conditions stated in Equations 5.1 and 5.2.

$$f := \sum_{k=0}^{n} a_k$$

that

$$a_k | f \quad \forall k$$

This results in

$$a_0 | a_0 + a_1$$

$$a_1 | a_0 + a_1$$

which simplifies to

$$a_0 | a_1$$

$$a_1 | a_0$$

this implies $a_0 = a_1$. This Gorenstein weighted projective space (G.w.p.s) allows the recovery of the polytope $[v_0, v_1] = [(-1), (1)]$ in N. This has width 2. This case can be seen in Figure 30.

Figure 30: Illustration of the single reflexive lattice polytope and its Width in 1 Dimension

5.3 Reflexive Lattice Simplices in Dimension 2

There are only 3 Gorenstein weighted projective spaces in dimension 2. This can be easily demonstrated with some elementary algebra.

From Equations 5.3 and 5.1. We can write:

$$a_0 | a_0 + a_1 + a_2,$$
$$a_1 | a_0 + a_1 + a_2,$$
$$a_2 | a_0 + a_1 + a_2.$$

These can be simplified to:

$$a_0 | a_1 + a_2,$$
$$a_1 | a_0 + a_2,$$
$$a_2 | a_0 + a_1.$$

As $a_2 \geq a_1, a_0$, either $a_0 = a_1 = a_2$ or $a_2 = a_0 + a_1$. In the first case, clearly $[a_0, a_1, a_2] = [1, 1, 1]$, as any other values would then be reduced. In the second case, we can then write

$$a_0 | a_0 + 2 \cdot a_1$$
$$a_1 | 2 \cdot a_0 + a_1$$

simplifying to

$$a_0|2 \cdot a_1$$

$$a_1|2 \cdot a_0$$

clearly either $a_1 = a_0$, in which case (as $a_2 = a_0 + a_1$) $[a_0, a_1, a_2] = [1, 1, 2]$ or

$a_1 = 2 \cdot a_0$, in which case (as $a_2 = a_0 + a_1$) $[a_0, a_1, a_2] = [1, 2, 3]$.

As such, we have our result in 2 dimensions, the only Gorenstein weighted projective spaces will have weights: $[1, 1, 1]$, $[1, 1, 2]$ or $[1, 2, 3]$.

It has been discussed how Gorenstein weighted projective spaces can be used to find polytopes. It has also been discussed how it is possible to examine the width of these polytopes through the method of finding a width relation. For these G.w.p.s an interior product to **0** can be seen, with a vector containing only {-1,0,1} in each case.

$$\langle [1, -1, 0], [1, 1, 1] \rangle = \mathbf{0}$$

$$\langle [1, -1, 0], [1, 1, 2] \rangle = \mathbf{0}$$

$$\langle [1, 1, -1], [1, 2, 3] \rangle = \mathbf{0}$$

As such, this method doesn't find any reflexive lattice polytopes with width 3 in dimension 2. However, we have already demonstrated the polytope

$$[(v_0, v_1, v_2)] = [(-1, -1), (2, -1), (-1, 2)]$$

has width 3. This is demonstrative of a weakness of this method, as it can only provide implication in one direction. That is, it can find that polytopes exist, but it is not exhaustive.

6 Computational Methods

6.1 Tree Search

To this point finding all Gorenstein weighted projective spaces (G.w.p.s) algebraically has been possible. For higher dimensions however this becomes more difficult to do, both for space reasons, but also that the number of G.w.p.s that exist increases swiftly. However, this number of cases are still possible for a computer to find through the method outlined below. Inspiration for this tree search was provided by Thomas Hall (2020).

Because the aim is to find polytopes with width ≥ 2, many of the steps involve reducing the search space whilst not eliminating any G.w.p.s that would have width relation ≥ 3.

Recalling f. In any G.w.p.s, f is the sum of the weights. It is also true that $a_i|f$ for any i. As such, there exists k_i such that $k_i \cdot a_i = f$. There are some results about k_i which should be observed.

First, that for $i < j$, $k_i \geq k_j$. This comes from our previous result which stated that for $i < j$, $a_i \leq a_j$.

Second, that if $k_n = 2$, then $a_n = \frac{f}{2}$ and so $a_n = \sum_{i=0}^{n-1} a_i$.

Third, if $k_i = k_j$ for $i \neq j$ then clearly $a_i = a_j$.

We will develop a tree search based approach in order to find G.w.p.s in order to recover polytopes and examine their properties. We begin this tree search by creating a set k_n containing all possible values k_n could hold for dimension n. In dimension 5 for example, it contains $3, 4, 5$. The minimum element is chosen to be 3 because as discussed, if $k_5 = 2$ then the polytope has width 2 which we are not interested in. We can then use Equation 5.1 and Equation 5.4 to get:

$$k_n a_n = a_0 + a_1 + \cdots + a_n.$$

Using 5.3 we can re-arrange to

$$k_n a_n < (n+1)a_n,$$

a_n is a positive integer, so

$$k_n < n + 1.$$

The core principle of this tree search is creating a set of values for the k_i on the level currently being examined, with all higher levels being fixed. Anytime an integer value for k_0 is found, the vector of weights is stored. A fully worked example of the tree search in practice is shown later in Section 6.4 and Section 6.5.

The tree search alone does not guarantee that all remaining G.w.p.s will allow the recovery of polytopes with width ≥ 2 as we will see, but all polytopes recovered from G.w.p.s which the search will eliminate will have width $= 2$.

Continuing with our development of the tree search approach, we then examine what values k_{n-1} could take for the first value of k_n. This range will begin at k_n+1. This is to prevent $a_i = a_j$, in which case width(P)=2. The range will end at $n(1 - \frac{1}{k_n})^{(-1)}$. The derivation for this is as follows.
We recall 5.1

$$a_i | f \quad \forall 0 \leq i \leq n$$

We have a value has already been chosen for k_n. We know that $k_{n-1} > k_n$, so we can attach a lower bound of $k_{n-1} \geq k_n + 1$. If $k_n a_n = f$ by Equation (5.4), then

$$f - a_n = f - \frac{f}{k_n}$$

$$f = (f - a_n)\left(1 - \frac{1}{k_n}\right)^{-1}$$

$$k_{n-1}a_{n-1} = f = a_0 + a_1 + \cdots + a_n = (f - a_n)\left(1 - \frac{1}{k_n}\right)^{-1}$$

$f - a_n = a_0 + a_1 + \cdots + a_{n-1}$ so

$$k_{n-1}a_{n-1} = (a_0 + a_1 + \cdots + a_{n-1})\left(1 - \frac{1}{k_n}\right)^{-1}$$

Using 5.3

$$k_{n-1}a_{n-1} < na_{n-1}\left(1 - \frac{1}{k_n}\right)^{-1}$$

We have $a_i \in \mathbb{N}$, so we can cancel, and thus find

$$k_{n-1} < n\left(1 - \frac{1}{k_n}\right)^{-1}$$

which becomes our upper bound for k_{n-1}.

6.2 Implementing a Check for the Width of the Polytope

In the previous step we identified candidate Gorenstein weighted projective spaces from which polytopes can be recovered. For example, the first obtained result in a search in dimension 5 is $[1, 2, 3, 4, 6, 8]$. This result corresponds to a Gorenstein weighted projective space $\mathbb{P}(1, 2, 3, 4, 6, 8)$. This can be seen as the sum of these is $1+2+3+4+6+8 = 24$, and each of these divides 24. For example $(6|24)$ as required. However, it is straightforward to see that the polytope this will generate has width 2, as $a_0 + a_1 = a_2$. Not all cases are as trivial as this, and when many results are being examined, a program is also needed to automatically assess whether the widths of the polytopes obtained are width 2. This is done through the steps set out below with the intention of sieving the results to remove all width 2 polytopes.

First, a matrix was generated such that it has dimensions $(n + 1) \times 3^{(n+1)}$ (with n the dimension we are examining). An example matrix for dimension 2 can be seen in Figure 31. The order of the columns is not significant, but it is easy to see that all possible

-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
-1	-1	-1	0	0	0	1	1	1	-1	-1	-1	0	0	0	1	1	1	-1	-1	-1	0	0	0	1	1	1
-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1

Figure 31: Dimension 2 Check Matrix for Width 2

columns of the elements $\{-1, 0, 1\}$ appear.

Second, matrix multiplication is used with the weights found by the tree search. Consider for example, the two sets of weights, [1,2,3] and [1,1,2]. These are both Gorenstein and in our code, would be present as a 2x3 matrix, which allows them to be checked simultaneously. Multiplication of this 2x3 and 3x27 will result in a 2x27 matrix. Every possible combination of subtraction, addition and null of each element will be present as shown in Figure 32.

-6	-3	0	-4	-1	2	-2	1	4	-5	-2	1	-3	0	3	-1	2	5	-4	-1	2	-2	1	4	0	3	6
-4	-2	0	-3	-1	1	-2	0	2	-3	-1	1	-2	0	2	-1	1	3	-2	0	2	-1	1	3	0	2	4

Figure 32: Results of the check on [1,2,3] and [1,1,2]

Now, if any of these are 0 then that case can be removed from the list, as it no longer corresponds to a G.w.p.s from which a polytope with potential width ≥ 3 can be recovered (excluding the [0;0;0] column, which trivially will always be 0). In our examples, more than one 0 can be seen in both rows and as such, both of the corresponding polytopes are found to have width 2 (as was already shown in section 5.3).

After this sieving step, any remaining G.w.p.s will not allow the recovery of polytopes with width 2.

This method can easily be expanded to check whether or not the corresponding polytopes have width 3, by including an additional element 2 (though replacing it with -2 would have the same result). An example of this can be seen in Figure 33.

Columns 1 through 18

-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0
-1	-1	-1	0	0	0	1	1	1	2	2	2	-1	-1	-1	0	0	0
-1	0	1	2	-1	0	1	2	-1	0	1	2	-1	0	1	2	-1	0

Columns 19 through 36

1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
1	1	1	2	2	2	-1	-1	-1	0	0	0	1	1	1	2	2	2
1	2	-1	0	1	2	-1	0	1	2	-1	0	1	2	-1	0	1	2

Figure 33: Width 2 and 3 Check Matrix for sieve in Dimension 2

This matrix (Figure 33) then has dimension $(n+1) \times 4^{(n+1)}$. It is know that there are no G.w.p.s from which polytopes are recovered with width 3 or greater until dimension 5 6.7, in which this width 3 check matrix has 24576 elements. As such, this matrix will not appear here. Sieving using this matrix would result in all remaining elements from the tree search having width ≥ 4.

The technique then has the potential to be expanded to check for width in higher dimensions. However, instead of a single matrix multiplication and subsequent checks being required, multiple checks would need to be made. For example, for width 4, two checks would be required. One examining whether the set [-2,-1,0,1,2] can be used to demonstrate width 4, and and second for [-1,0,1,2,3]. These cannot be combined as the union of these, [-2,-1,0,1,2,3] would also eliminate some G.w.p.s which generate polytopes of width 5.

6.3 Computational Examination Method

After the sieving step has been performed, a number of further steps are required. First, a set of weights is output. Then, each of the columns of the check matrix which reveal this is width $\leq x$ is output (where x is the width the check matrix examines up to). This will involve some repeats. For example, if [-1,-1,0,0,1] is a set of weights, then [1,1,0,0,-1] will also be. Any G.w.p.s from which a polytope with insufficient width to be of interest would will be removed, completing the sieve, and finally there will be an output of all G.w.p.s from which polytopes with width $> x$ are recovered.

6.4 Worked Example for Dimension 3

As mentioned in 6.1, it is no longer possible to find all Gorenstein weighted projective spaces in dimension 3 by hand. However, it is easily possible to examine the tree search for G.w.p.s by hand, which will allow us to demonstrate than no G.w.p.s from which polytopes with width ≥ 3 are recovered exist in dimension 3.

Following the method outlined in 6.1, It can be seen that

$$k_3 a_3 = a_0 + a_1 + a_2 + a_3$$

$a_3 > a_2 > a_1 \ldots$ and as such, we can re-arrange to get

$$(k_3 - 1) < 3 \text{ and so } 2 < k_3 < 4$$

$$\implies k_3 = 3.$$

From this we have

$$3a_3 = a_0 + a_1 + a_2 + a_3$$

$$a_3 = \frac{a_0 + a_1 + a_2}{2}$$

using

$$f = a_0 + a_1 + a_2 + a_3$$

substituting for a_3

$$f = \frac{3}{2}(a_0 + a_1 + a_2)$$

yields

$$k_2 a_2 = \frac{3}{2}(a_0 + a_1 + a_2)$$

so

$$(k_2)a_2 < \frac{3}{2}3a_2.$$

Therefore

$$k_2 < \frac{9}{2} \implies k_2 = 4.$$

Examining k_2=4, here

$$f = 4a_2 = \frac{3}{2}(a_0 + a_1 + a_2),$$

$$\frac{5}{2}a_2 = \frac{3}{2}(a_0 + a_1),$$

$$a_2 = \frac{3}{5}(a_0 + a_1),$$

$$f = \frac{12}{5}(a_0 + a_1),$$

$$k_1 a_1 = \frac{12}{5}(a_0 + a_1),$$

so

$$(k_1 - \frac{12}{5}) < \frac{12}{5}$$

and

$$k_1 < \frac{24}{5} < 5 \implies k_1 = 4.$$

However, $k_2 = 4$, and we have for $i \neq j$ $a_i = a_j \implies \text{width}(P) = 2$. As such, this branch of the tree search will not give us a result with width$\geq$2, and we need proceed no further.

It can be seen that this tree search never expands beyond a single branch, however this is a unique case. For searches in dimension $n \geq 4$, it will expand at the k_n level, and often at lower levels as well. This can be seen in Figure 34.

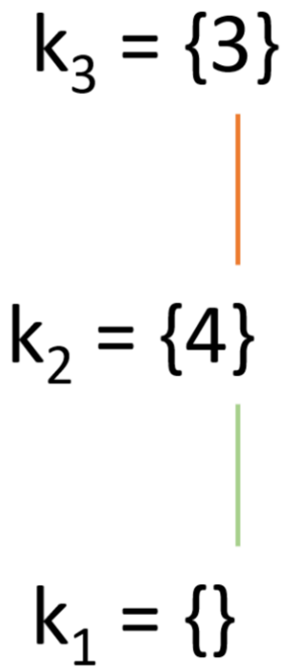

Figure 34: Diagram representing the path of the Tree Search for Dimension 3

6.5 Worked Example for Dimension 4

The tree for Dimension 4 is rather large, and demonstrating all the calculation used would be very time consuming. As such, we will consider two paths through the tree, one which leads to a solution, one of which does not.

We begin with:

$$f = k_4 a_4 = a_0 + a_1 + a_2 + a_3 + a_4$$

from which we can derive as above

$$k_4 < 5$$

using Equation 5.3. Accordingly:

$$k_4 = \{3, 4\}$$

In this case, just as the code (Appendix 48), we begin with the smallest value, $k_4 = 3$:

$$f = k_4 a_4 = 3a_4 = a_0 + a_1 + a_2 + a_3 + a_4$$

We have asserted $a_4 = \frac{f}{3}$ and accordingly

$$\frac{2f}{3} = a_0 + a_1 + a_2 + a_3$$

so

$$f = k_3 a_3 = \frac{3}{2}(a_0 + a_1 + a_2 + a_3)$$

using equation 5.3 we have

$$k_3 a_3 < \frac{3}{2}(4a_3)$$

Therefore

$$k_3 < 6$$

. Thus, we have

$$k_3 = \{4, 5\}$$

knowing $k_3 > k_4 = 3$ from equation 5.4. Again, we choose the smallest value in this range, and so set $k_3 = 4$:

$$f = k_3 a_3 = 4a_3 = \frac{3}{2}(a_0 + a_1 + a_2 + a_3)$$

We have asserted $a_3 = \frac{f}{4}$ and $a_4 = \frac{f}{3}$ and accordingly

$$a_0 + a_1 + a_2 = (1 - \frac{1}{3} - \frac{1}{4})f$$

$$f = k_2 a_2 = \frac{12}{5}(a_0 + a_1 + a_2)$$

We again use 5.3 to derive

$$k_2 a_2 < \frac{12}{5}(3a_2)$$

$$k_2 < \frac{36}{5} < 8$$

thus

$$k_2 = \{5, 6, 7\}$$

Once again, selecting the leftmost value: $k_2 = 5$, we have:

$$a_0 + a_1 = (1 - \frac{1}{3} - \frac{1}{4} - \frac{1}{5})f,$$

$$f = k_1 a_1 = \frac{60}{13}(a_0 + a_1).$$

Using 5.3 to derive

$$k_1 a_1 < \frac{60}{13}(2a_1),$$

$$k_1 < \frac{120}{13} < 10$$

so

$$k_1 = \{6, 7, 8, 9\}.$$

Selecting the smallest of these, $k_1 = 6$, we can derive

$$a_0 = f - \frac{f}{k_4} - \frac{f}{k_3} - \cdots - \frac{f}{k_0}$$

$$a_0 = (1 - \frac{1}{3} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6})f$$

$$a_0 = \frac{f}{20} \implies k_0 = 20$$

This leads to the set of weights

$$[k_0, k_1, k_2, k_3, k_4] = [20, 6, 5, 4, 3].$$

From this we can derive a G.w.p.s, and in order to do this we find the lowest common multiple of these lcm(3,4,5,6,20)=60. Dividing 60 by each of the recovered k_i yields the G.w.p.s

$$\mathbb{P}(a_0, a_1, a_2, a_3, a_4) = \mathbb{P}(3, 10, 12, 15, 20).$$

This is the only result of this branch of the tree search, and so the final output of our code will be a 1x5 matrix, containing the elements

$$[3, 10, 12, 15, 20].$$

The alternate branch where $k_4 = 3, k_3 = 4, k_2 = 5, k_1 = 7$ will now be examined. This is the next branch that the code would follow.

$$a_0 = (1 - \frac{1}{3} - \frac{1}{4} - \frac{1}{5} - \frac{1}{7})f$$
$$a_0 = \frac{31}{420}f \implies k_0 = \frac{420}{31}$$

However, we have that $k_i \in \mathbb{N}\ \forall i$ such that $0 \leq i \leq n$ from Equation 5.4. As such $k_0 \notin \mathbb{N}$, which implies this path does not lead to any G.w.p.s.

Returning to our output, $\mathbb{P}(3, 10, 12, 15, 20)$, it is easy to see that $3 + 12 = 15$, so P is width 2, using Equation 5.6. This corresponds to the column of the check matrix with elements $[1, 0, 1, -1, 0, 0]^T$. There also exists a corresponding second column of the table $[-1, 0, -1, 1, 0, 0]^T$.

Figure 35 shows the full structure of the tree search for dimension 4. Note should be drawn in particular to the successful path on the far left covered above.

6.6 Dimension 4 Computational Results

The code used in this section can be found in the Appendix Figure 48.

We run the tree search on dimension 4, and receive an output matrix B with the following results:

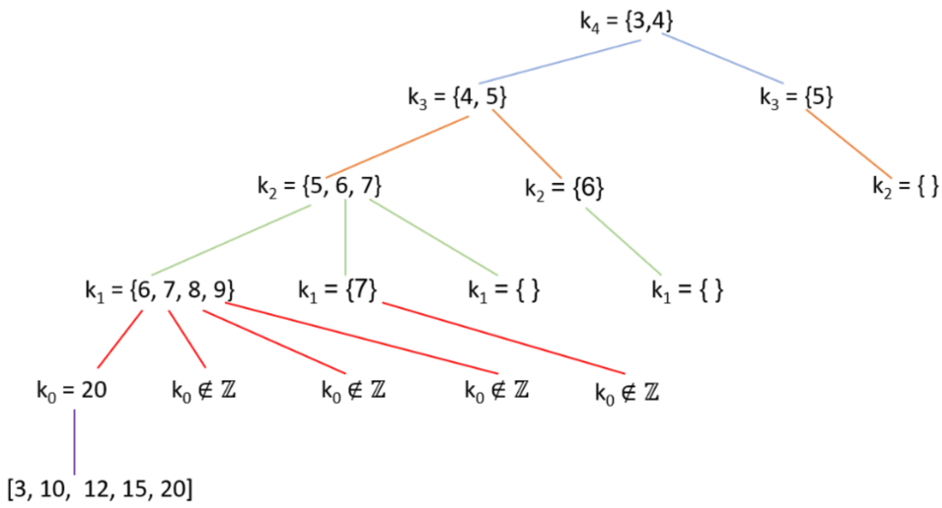

Figure 35: The full tree for a Tree Search of Weights in Dimension 4

$$\begin{bmatrix} 3 & 10 & 12 & 15 & 20 \end{bmatrix}.$$

Therefore, in fact this tree search only results in one G.w.p.s. Matrix multiplication with the width 2 check matrix (Section 6.2) then results in the output shown in Figure 36.

As discussed in Section 6.3, the results shown in Figure 36 are the first (and in this case only) G.w.p.s found. Then follows columns of the check matrix which demonstrate that P has width$\leq$ 2. Finally, B, the matrix of the G.w.p.s which passes through the sieve (and so construct P with width > 2). Our single example $\begin{bmatrix} 3 & 10 & 12 & 15 & 20 \end{bmatrix}$ has width 2, and so the output is empty, giving the result shown in Figure 37. There exist no G.w.p.s in dimension 4 which recover (through the method discussed) polytopes with width > 2.

For demonstration purposes, we also examine the computational results of the check for polytopes of width 3. This can be applied either to the results of the tree search, in one step removing all G.w.p.s with P of width 2 and 3, or to the results of the width 2 sieve, leading to the same result.

In the case seen in Figure 37, the sieve was done to the immediate results of the tree search. Accordingly, our single result isn't eliminated prior to this sieve. As such, the sieve results correspond to the check-matrix in which the polytope has width 2 (if no

```
P =

     3    10    12    15    20


Demonstration_of_width_relation =

    -1     0    -1     1     0


Demonstration_of_width_relation =

    -1     1    -1    -1     1


Demonstration_of_width_relation =

     1    -1     1     1    -1


Demonstration_of_width_relation =

     1     0     1    -1     0


B =

  0×5 empty double matrix
```

Figure 36: Table showing the sieved results for width 2

```
P =

     3    10    12    15    20


Demonstration_of_width_relation =

    -1    -1    -1    -1     2


Demonstration_of_width_relation =

    -1     0    -1     1     0


Demonstration_of_width_relation =

    -1     1    -1    -1     1


Demonstration_of_width_relation =

    -1     2    -1     1    -1


Demonstration_of_width_relation =

     0    -1     0     2    -1


Demonstration_of_width_relation =

     0     2     0     0    -1


Demonstration_of_width_relation =

     1    -1     1     1    -1


Demonstration_of_width_relation =

     1     0     1    -1     0


Demonstration_of_width_relation =

     1     2     1    -1    -1


Demonstration_of_width_relation =

     2    -1     2     0    -1


B =

  0×5 empty double matrix
```

Figure 37: Table showing the sieved results for width 3

weight of 2 appears) or width 3 (if there is a weight of 3). It is of note that the results as before are repeated after inversion for width 2 but not for width 3, as though +2 appears, -2 does not appear.

6.7 Dimension 5 Computational Results

The code used in this section can be found in Appendix 49.

Running the tree search on dimension 5, results in an output matrix B as shown in Figure 38.

1	2	3	4	6	8
1	2	10	12	15	20
1	5	20	24	30	40
1	8	9	12	18	24
1	10	15	24	30	40
1	18	20	36	45	60
1	20	70	84	105	140
1	30	60	84	105	140
1	120	165	264	330	440
2	3	4	6	9	12
2	5	6	12	15	20
2	7	12	14	21	28
3	4	6	12	15	20
3	6	12	14	21	28
3	8	15	24	30	40
3	12	15	20	30	40
3	12	50	60	75	100
3	28	60	84	105	140
3	30	110	132	165	220
3	42	60	70	105	140
4	5	6	10	15	20
4	5	30	36	45	60
4	15	20	36	45	60
5	6	15	24	30	40
6	15	70	84	105	140
9	10	20	36	45	60
10	21	60	84	105	140

Figure 38: Table showing results of the Tree Search in Dimension 5

This results in 27 G.w.p.s. Matrix multiplication the width 2 check matrix (as discussed in Section 6.2) sieves this set of results, leaving only two from which P with width ≥ 3 are recovered, as seen in Figure 39.

1	30	60	84	105	140
1	120	165	264	330	440

Figure 39: Table showing elements not removed by the width 2 sieve

The exact width of the polytopes these construct is still of interest to us. Accordingly, we take these results, and sieve them to examine if they are width 3. The results of this step can be seen in Figure 40.

```
P =

     1    30    60    84   105   140


Demonstration_of_width_relation =

    -1    -1    -1    -1    -1     2


Demonstration_of_width_relation =

    -1     0     2    -1     1    -1


Demonstration_of_width_relation =

    -1     2     1    -1     1    -1


Demonstration_of_width_relation =

     0     2    -1     0     0     0


Demonstration_of_width_relation =

     2    -1     0     2     0    -1


Demonstration_of_width_relation =

     2     1    -1     2     0    -1
```

```
P =

     1   120   165   264   330   440


Demonstration_of_width_relation =

    -1    -1    -1    -1    -1     2


Demonstration_of_width_relation =

    -1    -1     1    -1     2    -1


Demonstration_of_width_relation =

     0     0     2     0    -1     0


Demonstration_of_width_relation =

     2     2     0     2    -1    -1


B =

  0×6 empty double matrix
```

Figure 40: The Results of the Width 3 Sieve on Dimension 5

The final output of the code is an empty matrix. As such, the polytopes recovered from both of these G.w.p.s are of width 3, as both G.w.p.s therefore had a width 3 relation. This gives us the following result:

There exist reflexive polytopes of width>2 in dimension n=5.

6.8 Dimension 6 Computational Results

The code used in this section can be found in Appendix 50.

We apply the tree search to find polytopes in dimension 6 and, as previously, receive an output matrix B. The size of the tree search expands rapidly, and the table of results becomes far too large to be included here. This set of results contains 948 G.w.p.s. A small subset of the results are shown as examples as seen in Figure 41.

B =

1	2	3	6	8	12	16
1	2	3	20	24	30	40
1	2	4	6	12	15	20
1	2	6	9	12	18	24
1	2	6	12	14	21	28
1	2	6	30	36	45	60
1	2	8	15	24	30	40
1	2	12	15	20	30	40
1	2	12	50	60	75	100
1	2	15	60	72	90	120
1	2	24	27	36	54	72
1	2	28	60	84	105	140
1	2	30	45	72	90	120
1	2	30	110	132	165	220
1	2	42	60	70	105	140
1	2	54	60	108	135	180
1	2	60	210	252	315	420
1	2	90	180	252	315	420
1	2	360	495	792	990	1320
1	3	4	5	12	15	20
1	3	5	6	10	15	20
1	3	5	30	36	45	60
1	3	6	8	12	18	24
1	3	8	12	16	24	32
1	3	8	40	48	60	80
1	3	10	12	24	30	40
1	3	14	24	28	42	56
1	3	15	20	36	45	60

Figure 41: Some Results of the Tree Search in Dimension 6

The next step, as with dimension 4 and 5 is to sieve for width 2 polytopes. This results in the removal of 756 G.w.p.s, each of which are therefore found to generate polytopes which have width 2. Thus, there exists at least 193 reflexive polytopes in dimension 6 with width ≥ 3. Figure 42 shows demonstrations of a width 2 relation for one G.w.p.s and some G.w.p.s from which polytopes are recovered that may have width > 2.

Next, the sieve for width 3 was performed. The last few lines of the results for this are shown in Figure 43.

$P = [33, 40, 7860, 12045, 19272, 249090, 32120]$ and the columns of the check matrix that demonstrate P is of width 3 are visible in Figure 43.

B is empty, and as such, no reflexive polytopes with width ≥ 4 in dimension 6 have been found.

```
P =

          60          70         273         780        1092        1365        1820

Demonstration_of_width_relation =

    -1    -1     1    -1     1     1    -1

Demonstration_of_width_relation =

     0     0    -1     0    -1     1     0

Demonstration_of_width_relation =

     0     0     1     0     1    -1     0

Demonstration_of_width_relation =

     1     1    -1     1    -1    -1     1

B =

           2          15         510        1020        1428        1785        2380
           2          15        2040        2805        4488        5610        7480
           2          21         690        1380        1932        2415        3220
           2          33        4200        5775        9240       11550       15400
           2          35        1110        2220        3108        3885        5180
           2          55        6840        9405       15048       18810       25080
           2         105         420        1020        1428        1785        2380
           2         105        3210        6420        8988       11235       14980
           2         165       20040       27555       44088       55110       73480
           3           4         210         420         588         735         980
           3           4         840        1155        1848        2310        3080
           3           5         240         480         672         840        1120
           3           5         960        1320        2112        2640        3520
           3           7         300         600         840        1050        1400
           3           8        1320        1815        2904        3630        4840
           3          10         390         780        1092        1365        1820
           3          10        1560        2145        3432        4290        5720
           3          11        1680        2310        3696        4620        6160
           3          14         510        1020        1428        1785        2380
           3          20          70         180         252         315         420
           3          20         690        1380        1932        2415        3220
```

Figure 42: Some of the Results of the Width 2 Sieve on Dimension 6

```
P =

          33          40        8760       12045       19272       24090       32120

Demonstration_of_width_relation =

    -1    -1    -1    -1    -1    -1     2

Demonstration_of_width_relation =

    -1    -1    -1     1    -1     2    -1

Demonstration_of_width_relation =

     0     0     0     2     0    -1     0

Demonstration_of_width_relation =

     2     2     2     0     2    -1    -1

B =

  0×7 empty double matrix
```

Figure 43: Results of the Width 3 Sieve on Dimension 6

6.9 Dimension 7 Computational Results

The tree search on Dimension 7 is much larger than that required for Dimension 6, as expected from the exponential expansion of the tree. This results in 194,185 G.w.p.s, some of which are shown in Figure 44.

Of these 194,185, a total of 133,396 pass through the width 2 sieve.

1	2	3	4	5	10	15	20
1	2	3	4	8	12	18	24
1	2	3	5	15	24	30	40
1	2	3	6	12	16	24	32
1	2	3	6	40	48	60	80
1	2	3	8	12	24	30	40
1	2	3	9	12	18	27	36
1	2	3	12	18	24	36	48
1	2	3	12	24	28	42	56
1	2	3	12	60	72	90	120
1	2	3	15	18	36	45	60
1	2	3	15	70	84	105	140
1	2	3	16	30	48	60	80
1	2	3	21	36	42	63	84
1	2	3	24	30	40	60	80
1	2	3	24	100	120	150	200
1	2	3	30	120	144	180	240
1	2	3	48	54	72	108	144
1	2	3	56	120	168	210	280
1	2	3	60	90	144	180	240
1	2	3	60	220	264	330	440
1	2	3	84	120	140	210	280
1	2	3	108	120	216	270	360
1	2	3	120	420	504	630	840
1	2	3	180	360	504	630	840
1	2	3	720	990	1584	1980	2640
1	2	4	5	6	10	12	20
1	2	4	5	40	48	60	80
1	2	4	6	8	9	18	24
1	2	4	8	15	20	30	40
1	2	4	12	20	36	45	60

Figure 44: Some Results of the Tree Search in Dimension 7

The G.w.p.s listed below have some interesting properties. This will be discussed further below.

$$[2, 9, 280, 315, 360, 420, 504, 630]$$

$$[3, 8, 280, 315, 360, 420, 504, 630]$$

$$[4, 315, 8120, 9135, 10440, 12180, 14616, 18270]$$

The G.w.p.s listed above have some interesting properties, most significantly that the facet width of each is 4, as will be discussed below. Second, the f value for the first two G.w.p.s is the same, and is 2520 in both cases. That is, it can be seen that

$$2|2520, 3|2520, \ldots, 630|2520$$

This confirms that both w.p.s are Gorenstein.

Similarly, f for the third G.w.p.s is 73080, and it can be seen that

$$4|73080, 315|73080, \ldots, 18270|73080$$

These are also the results that successfully passed through the width 2 sieve, so they all have width $\geq$3.

The facet width of these polytopes is the k_n (here $k_n = k_7$) value. In each case, this is

$$\frac{2520}{630} = \frac{73080}{18270} = 4.$$

All G.w.p.s found in lower dimensions recover polytopes with facet width 3, so the existence of these polytopes is an interesting result.
All of these lattice polytopes also have width 3, and so in addition to having demonstrated the existence of lattice polytopes with lattice width $\geq$ 4, these also demonstrate the existence of lattice polytopes with width $\geq$ 3 in dimension 7.

6.10 Dimension 8 Computational Results

The computational methods currently used are not appropriate for a full tree search for dimension 8. A reduced search was performed however, considering only G.w.p.s from which polytopes with facet width $\geq$ 4 were recovered. This was achieved by excluding branches of the tree where $k_8 = 3$. This results in the existence having been demonstrated of more than 9200 unique G.w.p.s from which polytopes are recovered with facet width 4. There are no G.w.p.s which lead to polytopes in dimension 8 with facet width 5. One example of a G.w.p.s with facet width 4 is $\mathbb{P}(35, 333, 1260, 41440, 46620, 53280, 62160, 74592, 93240)$. Checks reveal that this is Gorenstein, well formed, and also that it has no width 2 relation, which implies the recovered polytope has width $\geq$ 3. In Figure 45 some of the other recovered results are shown, each representing a G.w.p.s from which a polytope with width $\geq$ 3 and facet width 4 can be recovered.

35	153	9520	10080	21420	24480	28560	34272	42840
35	156	80220	160440	260715	297960	347620	417144	521430
35	168	5220	138040	155295	177480	207060	248472	310590
35	171	630	21280	23940	27360	31920	38304	47880
35	180	258	12040	13545	15480	18060	21672	27090
35	180	1204	36120	40635	46440	54180	65016	81270
35	180	3096	84280	94815	108360	126420	151704	189630
35	184	540	19320	21735	24840	28980	34776	43470
35	184	55188	1410360	1586655	1813320	2115540	2538648	3173310
35	198	27960	717640	807345	922680	1076460	1291752	1614690
35	207	60984	1558480	1753290	2003760	2337720	2805264	3506580
35	238	7020	185640	208845	238680	278460	334152	417690
35	240	693	24640	27720	31680	36960	44352	55440
35	252	7380	195160	219555	250920	292740	351288	439110
35	261	23310	600880	675990	772560	901320	1081584	1351980
35	264	35880	920920	1036035	1184040	1381380	1657656	2072070
35	270	1708	51240	57645	65880	76860	92232	115290
35	276	78372	2002840	2253195	2575080	3004260	3605112	4506390
35	297	1120	36960	41580	47520	55440	66528	83160
35	312	145740	291480	473655	541320	631540	757848	947310
35	333	1260	41440	46620	53280	62160	74592	93240
35	342	2340	69160	77805	88920	103740	124488	155610
35	360	474	22120	24885	28440	33180	39816	49770
35	360	2212	66360	74655	85320	99540	119448	149310
35	360	5688	154840	174195	199080	232260	278712	348390
35	396	51720	1327480	1493415	1706760	1991220	2389464	2986830
35	414	113148	2891560	3253005	3717720	4337340	5204808	6506010
35	476	13140	347480	390915	446760	521220	625464	781830
35	492	7140	195160	219555	250920	292740	351288	439110
35	504	792	33880	38115	43560	50820	60984	76230
35	552	147924	3780280	4252815	4860360	5670420	6804504	8505630
35	632	19320	508760	572355	654120	763140	915768	1144710
35	657	2520	81760	91980	105120	122640	147168	183960
35	714	19260	509320	572985	654840	763980	916776	1145970
35	792	99240	2547160	2865555	3274920	3820740	4584888	5731110
35	828	217476	5557720	6252435	7145640	8336580	10003896	12504870
35	873	3360	108640	122220	139680	162960	195552	244440
35	952	25380	671160	755055	862920	1006740	1208088	1510110
35	1071	28440	752080	846090	966960	1128120	1353744	1692180
35	1080	6244	187320	210735	240840	280980	337176	421470
35	1148	4680	149240	167895	191880	223860	268632	335790

Figure 45: Some Results of the Tree Search in Dimension 8

6.11 Dimension 9 Computational Results

As with dimension 8, the computational methods used are not appropriate for a full tree search for dimension 9. As before, a reduced search was performed, considering only G.w.p.s from which polytopes with facet width ≥ 4 were recovered. This was again achieved by excluding branches of the tree where $k_9 = 3$. This does find some G.w.p.s, and one example of a G.w.p.s from which a polytope with width ≥ 3 can be recovered is:

$$\mathbb{P}(11, 360, 24255, 65280, 38808, 43120, 48510, 55440, 64680, 77616).$$

7 Conclusions

This dissertation has introduced the concepts of lattices, polytopes, several related properties, as well as projective spaces and several specific types of projective space. The existence of a reflexive lattice polytope with width 3 was demonstrated in dimension 2, through direct example. It was also demonstrated that there can exist no reflexive lattice polytopes with width 3 in dimension 1. No examples of polytopes with width 3 were found in dimensions 3 and 4, though this does not mean they cannot exist, rather that they were not demonstrated herein. It was also shown that there exist at least 2 in dimension 5, at least 193 in dimension 6, and at least 133,396 many in dimension 7.

It was also found that there were no polytopes derived from polytopes with facet width > 3 for dimension below 7, but that some do exist in dimension 7, dimension 8 and 9.

These results, both with respect to width and to facet width have application in the study of the properties of toric varieties, and their relation to reflexive lattice polytopes.

8 Further Work

One potential further area of study involves changing the tree search method used here in order to avoid the elimination of all G.w.p.s which generate polytopes with width 2. This updated formula can subsequently be applied to dimensions 2, 3, 4 and 5.

A major step which was not taken here was the explicit recovery of all polytopes, as designated by their vertices. The method was demonstrated and discussed, and a few simple examples were explicitly found, but implementing computational methods to find these polytopes would be an additional step.

Another potential source of further study would be that of verifying results already demonstrated by (Kreuzer & Skarke 1997). They demonstrated an algorithm that, in addition to confirming the 16 isomorphism classes (Section 2.13) that exist for n=2, shows that there are 4319 for n=3 and 473,800,776 for n=4. Examining in particular how the polytopes generated by our G.w.p.s are distributed among these classes could be interesting.

Another area of further study relates to performing improvements on the efficiency of the code in order to perform full tree searches on dimension 8 and 9. It has been discussed how reduced searches were performed on these dimensions, examining facet width, but more complete results related to width could also provide interesting insight into lattice polytopes and toric varieties.

Implementing improvements to the code in order to make it easier to use without comprehending the full procedure would also provide additional work. Implementing recursive functions for the dimensions and the width relations would help with this. Additionally, implementing aspects of the sieve into the search as it is first performed would cut down on the size of the search space, reducing the number of width 2 results.

Most significantly, examining the resulting polytopes found by the tree searches and using them to examine the properties of toric varieties would provide a source of further work.

9 Appendix

9.1 Matlab Code

checkmatrix3.m checkmatrix2.m

```matlab
function A=checkmatrix2(n)

if n==0
    A=[-1,0,1]; %make the dimension 0 check matrix
    return;
else
    a=length(checkmatrix2(n-1));%attach 3 copies of the matrix below sets of -1,0,1 of the appropriate width
    A=[linspace(-1,-1,a),linspace(0,0,a),linspace(1,1,a);checkmatrix2(n-1),checkmatrix2(n-1),checkmatrix2(n-1)];
end
```

Figure 46: Matlab code for the generation of the check matrix for the width 2 Sieve

checkmatrix3.m

```matlab
function A=checkmatrix3(n)

if n==0
    A=[-1,0,1,2]; %make the dimension 0 width 3 check matrix
    return;
else
    a=length(checkmatrix3(n-1)); %attach 4 copies of the matrix below sets of -1,0,1,2 of the appropriate width
    A=[linspace(-1,-1,a),linspace(0,0,a),linspace(1,1,a),linspace(2,2,a);checkmatrix3(n-1),checkmatrix3(n-1),checkmatrix3(n-1),checkmatrix3(n-1)];
end
```

Figure 47: Matlab code for the generation of the check matrix for the width 3 Sieve

n4test.m

```matlab
v=zeros(1,5); %initalise v, which stores successful solutions before adding them to B
B=zeros(1,5); %initalise B, the principle output
n=4;          %set the dimension of the calculation to be 4
f=1;          %initalise f, the lcm of the ki
k4=3:n;       %set a range for k4
for i4=1:length(k4)      %in turn iterate through the elements of k4
    if 1-1/k4(i4)>0      %check that our choice is reasonable
        k3=(k4(i4)+1):(((1-1/k4(i4))^(-1))*4);  %set a range for k3
        for i3=1:length(k3)                       %this cycle repeats until
            if 1-1/k4(i4)-1/k3(i3)>0
                k2=(k3(i3)+1):(((1-1/k4(i4)-1/k3(i3))^(-1))*3);
                for i2=1:length(k2)
                    if 1-1/k4(i4)-1/k3(i3)-1/k2(i2)>0
                        k1=(k2(i2)+1):(((1-1/k4(i4)-1/k3(i3)-1/k2(i2))^(-1))*2);
                        for i1=1:length(k1)
                            if 1-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)>0
                                k0=ceil((1-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1))^(-1));  %here, the results of the calculation result in k0 is an integer
                                    f=lcm(k0,lcm(k1(i1),lcm(k2(i2),lcm(k3(i3),k4(i4))))); %calculate the appropriate f value
                                    if (1-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)-1/k0)<1e-10 && (1-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)-1/k0)>-1e-10 %check the choice of k0 was good
                                        v=[f/k0,f/k1(i1),f/k2(i2),f/k3(i3),f/k4(i4)];   %store the final values
                                        if gcd(v(1),gcd(v(2),gcd(v(3),gcd(v(4),v(5)))))==1 %check the output is reasonable
                                            B=[B;v];                                    %if it is, add the solution to the outputs
                                        end
                                    end
                            end
                        end
                    end
                end
            end
        end
    end
end
B(1,:)=[]; %remove the first row of the matrix (which is a row of zeros)
B          %output the final solution
```

Figure 48: Matlab code for the dimension 4 tree search

It is of note that the code as it appears in Figure 48 contains a number of inefficencies, particularly with relation to the calculation of gcd and lcm many more times than is necessary. Nevertheless, the small size of the tree means runtime is negligible.

```
n5test.m
v=zeros(1,6);
B=zeros(1,6);
n=5;
f=1;
k5=3:n;
for i5=1:length(k5)
    if 1-1/k5(i5)>1e-10
        k4=(k5(i5)+1):(((1-1/k5(i5))^(-1))*5);
        for i4=1:length(k4)
            if 1-1/k5(i5)-1/k4(i4)>1e-10
                k3=(k4(i4)+1):(((1-1/k5(i5)-1/k4(i4))^(-1))*4);
                c4=lcm(k5(i5),k4(i4));
                for i3=1:length(k3)
                    if 1-1/k5(i5)-1/k4(i4)-1/k3(i3)>1e-10
                        k2=(k3(i3)+1):(((1-1/k5(i5)-1/k4(i4)-1/k3(i3))^(-1))*3);
                        c3=lcm(c4,k3(i3));
                        for i2=1:length(k2)
                            if 1-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)>1e-10
                                k1=(k2(i2)+1):(((1-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2))^(-1))*2);
                                c2=lcm(c3,k2(i2));
                                for i1=1:length(k1)
                                    if 1-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)>1e-10
                                        k0=ceil((1-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1))^(-1));
                                        c1=lcm(c2,k1(i1));
                                            f=lcm(k0,c1);
                                            if (1-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)-1/k0)<1e-10 && (1-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)-1/k0)>-1e-10
                                                v=[f/k0,f/k1(i1),f/k2(i2),f/k3(i3),f/k4(i4),f/k5(i5)];
                                                B=[B;v];
                                                B=unique(B,'rows');
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
B(1,:)=[];
B
```

Figure 49: Matlab code for the dimension 5 tree search

```
n6test.m
v=zeros(1,7);
B=zeros(1,7);
n=6;
f=1;
k6=3:n;
for i6=1:length(k6)
    if 1-1/k6(i6)>eps
        k5=(k6(i6)+1):(((1-1/k6(i6))^(-1))*6);
for i5=1:length(k5)
    if 1-1/k6(i6)-1/k5(i5)>eps
        k4=(k5(i5)+1):(((1-1/k6(i6)-1/k5(i5))^(-1))*5);
        c5=lcm(k6(i6),k5(i5));
        for i4=1:length(k4)
            if 1-1/k6(i6)-1/k5(i5)-1/k4(i4)>eps
                k3=(k4(i4)+1):(((1-1/k6(i6)-1/k5(i5)-1/k4(i4))^(-1))*4);
                c4=lcm(k5(i5),k4(i4));
                for i3=1:length(k3)
                    if 1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)>eps
                        k2=(k3(i3)+1):(((1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3))^(-1))*3);
                        c3=lcm(c4,k3(i3));
                        for i2=1:length(k2)
                            if 1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)>eps
                                k1=(k2(i2)+1):(((1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2))^(-1))*2);
                                c2=lcm(c3,k2(i2));
                                for i1=1:length(k1)
                                    if 1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)>eps
                                        k0=ceil((1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1))^(-1));
                                        c1=lcm(c2,k1(i1));
                                            f=lcm(k0,c1);
                                            if (1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)-1/k0)<eps && (1-1/k6(i6)-1/k5(i5)-1/k4(i4)-1/k3(i3)-1/k2(i2)-1/k1(i1)-1/k0)>
                                                v=[f/k0,f/k1(i1),f/k2(i2),f/k3(i3),f/k4(i4),f/k5(i5),f/k6(i6)];
                                                B=[B;v];
                                                B=unique(B,'rows');
                                            end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
    end
end
B(1,:)=[]
```

Figure 50: Matlab code for the dimension 6 tree search

A number of improvements are visible in the code shown in Figure 51, particularly the use of c_i which reduces the number of calculations of lcm, and d_i which reduces the number of calculations of $1 - \frac{1}{kn} - \ldots$ etc. The removal of repeat results has also been

```
check_active42.m  ×  n5test.m  ×  n7test.m  ×  n4test.m  ×  +
v=zeros(1,8);           %initalise v
B=zeros(1,8);           %initialise B the matrix of outputs
n=7;                    %dimension 7
f=1;                    %initalise f
k7=3:n;                 %set a range for k7
for i7=1:length(k7)       %for each of the values in k7
    d7=1-1/k7(i7);        %setup d7, which holds the solution so it's not repeatedly recalculated
    if d7>1e-10           %check the choice of k7 is reasonable
        k6=(k7(i7)+1):((d7)^(-1))*7;       %set a range for k6
for i6=1:length(k6)                          %for each of the values in k6 based on the choice of k7
    d6=d7-1/k6(i6);                          %store d6, which will remain unil the next choice of i6
    if d6>1e-10                              %check the choice of k7 and k6 are reasonable
        k5=(k6(i6)+1):((d6)^(-1))*6;         %set a range of k5 based on the choices of k7 and k6
        c6=lcm(k7(i7),k6(i6));               %create c6, a process by which the lcm is calculated in stages in order to reduce redundant gcd subroutine calls
for i5=1:length(k5)
    d5=d6-1/k5(i5);
    if d5>1e-10
        k4=(k5(i5)+1):((d5)^(-1))*5;
        c5=lcm(c6,k5(i5));
        for i4=1:length(k4)
            d4=d5-1/k4(i4);
            if d4>1e-10
                k3=(k4(i4)+1):((d4)^(-1))*4;
                c4=lcm(c5,k4(i4));
                for i3=1:length(k3)
                    d3=d4-1/k3(i3);
                    if d3>1e-10
                        k2=(k3(i3)+1):((d3)^(-1))*3;
                        c3=lcm(c4,k3(i3));
                        for i2=1:length(k2)
                            d2=d3-1/k2(i2);
                            k2(i2)
                            if d2>1e-10
                                k1=(k2(i2)+1):((d2)^(-1))*2;
                                c2=lcm(c3,k2(i2));
                                for i1=1:length(k1)
                                    d1=d2-1/k1(i1);
                                 if d1>1e-10
                                     k0=ceil(d1)^(-1);           %set an integer value for k0 based on the choices of k7,k6,k5,k4,k3,k2,k1
                                     d0=d1-1/k0;                 %calculate the final d0
                                     c1=lcm(c2,k1(i1));          %calculate c1
                                         f=lcm(k0,c1);           %calculate f
                                         if d0<1e-10 && d0>-1e-10    %check the choices of k0,k1,k2,k3,k4,k5,k6,k7 are reasonable
                                             v=[f/k0,f/k1(i1),f/k2(i2),f/k3(i3),f/k4(i4),f/k5(i5),f/k6(i6),f/k7(i7)];    %generate the polytope this k values represent
                                             B=[B;v];                                                                      %add this to the matrix of outputs
                                         end
                                 end
                              end
                           end
                        end
                     end
                  end
               end
            end
         end
      end
end
    end
end
    end
end
B(1,:)=[];              %remove the first row which is only 0s
B=unique(B,'rows');     %remove repeat results from the outputs
B                       %output the results
```

Figure 51: Matlab code for the dimension 7 tree search

moved to the end, resulting in it being called once, rather than once for each polytope. This code takes several minutes to generate results, and so every efficiency improvement is important.

```
check_active42.m  ×  +
n4test                               %run the dimension 4 tree search
A=checkmatrix2(4);                   %generate A, the width 2 check matrix in dimension 4
C=B*A;                               %use matrix multiplication to obtain C, results of the check
weight=zeros(1,5);                   %initalise weight
for j=1:size(B,1)                    %for each of the polytopes generated by n4test
    P=B(j,:)                         %set and output P, to demonstrate which polytope the weights apply to
    for i=1:243
        if C(j,i)==0 && i~=122       %if the weight gives a result of 0, and it's not the always 0 central column
            weight(1,:)=A(:,i)       %output this weight
            B(j,:)=0;                %remove this polytope from B as it is width 2
        end
    end
end
ind = find(sum(B,2)==0) ;            %remove all 0 rows of B
 B(ind,:) = [] ;
 B                                   %output the results
```

Figure 52: Matlab code for the dimension 4 width 2 sieve

The code in Figure 53 is almost identical to the width 2 sieve, modified for $[-1, 0, 1, 2]$.

check_active43.m

```
%n4test
A=checkmatrix3(4);
C=B*A;
weight=zeros(1,5);
for j=1:size(B,1)
    P=B(j,:)
    for i=1:1024
        if C(j,i)==0 && i~=342
            weight(1,:)=A(:,i)
            B(j,:)=0;
        end
    end
end
B;
ind = find(sum(B,2)==0) ;
 B(ind,:) = [] ;
 B
```

Figure 53: Matlab code for the dimension 4 width 3 sieve

References

Gabrielle Balletti, Alexander M. Kasprzyk Three-dimensional lattice polytopes with two interior lattice points, arXiv:1612.08918 [math.CO] pg. 1 (2016)

Alathea Jensen Self-Polar Polytopes, arXiv:1902.00784 [math.CO]. (2019)

Alvise Trevisan Lattice Polytopes and Toric Varieties, Master's Thesis (2007).

Yunhyung Cho, Yoosik Kim, Eunjeong Lee, Kyeong-Dong Park Small resolutions of toric varieties associated to string polytopes of small indices, arXiv:1912.00658 [math.AG](2019).

William Fulton Algebraic curves. An Introduction to Algebraic Geometry (1969).

Weronika Buczy'nska Fake weighted projective spaces. arXiv:0805.1211v1(2008).

Alexander M. Kasprzyk Bounds on fake weighted projective space. arXiv:0805.1211v1 (2009).

Alexander M. Kasprzyk Classifying terminal weighted projective space, arXiv:1304.3029 [math.AG] (2013).

Gabriel Cramer Introduction à l'Analyse des lignes Courbes algébriques, pg 656-659 (1750).

Gabor Hegedus and Alexander M. Kasprzyk The Boundary Volume of a Lattice Polytope, arXiv:1002.2815 [math.CO](2011).

Maximilian Kreuzer and Harald Skarke On the Classification of Reflexive Polyhedra, arXiv:hep-th/9512204(1997).

Thomas Hall and Johannes Hofscheier Private Communication (December 2019).

Thomas Hall Private Communication (January 2020).

Heinke Conrads Weighted projetive spaces and reflexive simplices, manuscripta math. 107, 215–227 (2001).