

Haverford College Mathematics Undergraduate Senior Thesis

Minkowski Sum Decompositions of Convex Polygons

Robert Seater
Haverford College
Haverford, PA 19041
rseater@haverford.edu

May 1, 2002

Abstract

Minkowski sums provide a certain way of adding polytopes together and producing a new polytope. In this paper, we give an introduction to convex polytopes and address the question of decomposing polygons and polytopes into Minkowski summands. That is, given a polytope, how can we write it as a Minkowski sum of other polytopes?

We prove a theorem that allows us to use edge sets of polygons to calculate their Minkowski sum. Using this theorem, we show some other nice facts and provide an algorithm for finding Minkowski decompositions. We examine the set of all such decompositions of a given polygon, and show that it is itself a convex polytope.

Provided with this paper is a Mathematica notebook that allows the user to examine and experiment with Minkowski sums. It is a nice way to get an intuition for the background and theorems in this paper, as well as supplying empirical evidence for the conjectures introduced towards the end of the paper.

Keywords: *Minkowski Vector Sum Summand, Decomposition Polytope, Decomposable Indecomposable Reducible Irreducible, Convex Polygon Polytope Hull Combination, RMS DePo, Sliced Edge Set Edgeset Edge-Set, Stamping Algorithm, Mathematica Notebook Program Package Algorithm, Weight-Vector Edge-Vector Edge-Summand Weight Matrix*

Contents

1	Introduction	3
2	Background in Convex Polytopes	3
2.1	Convex Polytopes	3
2.2	Minkowski Sums	6
2.3	The Stamping Algorithm	8
2.4	Scaled Multiples of Polytopes	12
2.5	Decompositions of Polytopes	13
2.6	Hyperplanes, Supporting Hyperplanes, and Faces	13
2.7	Edge Sets	15
3	Minkowski Sums of Faces	16
3.1	Faces of Sums	18
4	Relating Minkowski Summation to Edge Sets	19
4.1	Edge Sets of Equivalent Polygons	19
4.2	Edge Sets and Decompositions	20
5	Decomposition Algorithm	21
5.1	Description of the Algorithm	22
5.2	Correctness of the Algorithm	23
6	Indecomposability	30
7	Symmetries	31
8	Finiteness and Decomposition	34
8.1	Infinite Aspects	34
8.2	Some Notation for Decompositions and Summands	35
8.3	The Decomposition Polytope	37
8.4	The Set of All Summands	38
8.5	Summary and Connections	40
9	Conclusion and Future Work	40
10	An Introduction to Time Complexity	43
11	The <i>Mathematica</i> Notebook: Usage Guide and Analysis	43
11.1	Functions and Syntax	44
11.2	Type Conversion Functions	44
11.3	Internal Workings of RMS (Recursive Minkowski Sum)	45
11.4	Complexity Analysis of RMS	47
11.5	Alternate Methods for Computing Minkowski Sums	47
11.6	The DePo Function (Decomposition Polytope)	48
12	Bounds on Lattice Volume	48
12.1	Background	48
12.2	Lattice Polygons	48
12.3	General Polygons	51
13	Acknowledgments	52

1 Introduction

Whenever there is a means of combining mathematical objects, there arises the questions of reversing the process. That is, given the result, could we have found the objects were combined to produce it? Is the decomposition unique? If not, what can we say about the set of all decompositions of a given object?

Minkowski summation provides a very salient means of combining a set of polytopes and producing a new polytope. One would like to be able to take any given a polytope and find all ways to express it as the Minkowski sum of other polytope. This process is particularly intriguing, since it turns out that the set of all such decompositions of a polygon is itself a polytope. We hope to provide the reader with the background necessary to understand and work with this “decomposition polytope” and appreciate its intricacies.

The problem of decomposing polytopes was first addressed by Gale in 1954 in [Gal54]. He stated, without proof, some basic properties of what he called “irreducible polytopes”. Grunbaum later formalized the language and basic results in his definitive treatment of convex sets in [Gru67]. Meyer proved more advanced results about Minkowski sums in [Mey74] and provided a more precise understanding of when one polytope is in the decomposition of another. In this paper, we focus our attention on the set of all decompositions of a given polytope.

Section 1 introduces the paper and gives a high level summary of the content. Section 2 gives a thorough introduction to convex polytopes, including basic terminology and fundamental theorems of the field. Section 3 introduces a theorem describing the relation between faces of a polytope and faces of its Minkowski summands. Section 4 introduces a very useful theorem for manipulating polygons, which gives a direct relation between the edge sets of a polygon and the edge sets of its Minkowski summands. Section 5 uses the tools introduced in previous section to prove an algorithm for finding a Minkowski decomposition of any polygon. Section 6 identified the indecomposable polygons. Section 7 correlates symmetries within a set of polygons to symmetries of their Minkowski sum. Section 8 discusses and identified the finite aspects of Minkowski sum decompositions of polygons. Section 9 concludes the paper and discusses directions of future work. Section 12 is a supplemental section with related, but not directly relevant, results. We gives tight bounds for the number of lattice points in a polytope as a function of the lattice points of its summands.

Another important part of this paper is the accompanying Mathematica notebook. This set of tools allows users to examine and experiment with Minkowski sums in order and get a better intuition for how they behave. It allows the user to efficiently perform Minkowski sums, as well as generating the decomposition polytope of any polygon. It is a nice way to get an intuition for the background and theorems in this paper, and generate additional examples where needed. Section 10 introduces some background in time complexity, and Section 11 describes the use of this tool, as well as proving its correctness and efficiency.

2 Background in Convex Polytopes

This section provides an overview of and introduction to convex polytopes, Minkowski sums, and related terminology and lemmas.

Subsection 2.1 introduces the formal definition of a convex polytope and proves some fundamental properties such as their convexity. Subsection 2.2 introduces the notion of a Minkowski sum, and provides some simple examples. Subsection 2.3 contains an algorithm for generating Minkowski sums without using the formal theorem. That algorithm is crucial to getting an intuition for Minkowski sums, since it allows us to quickly and accurately produce examples. Subsection 2.5 introduces the notion of decomposing a polytope into Minkowski summands. Subsection 2.6 gives us a thorough understanding of what a face is, in terms of supporting hyperplanes. Subsection 2.7 presents the idea of the edge set of a polygon, which will be linked to Minkowski summation later on in Section 4.

2.1 Convex Polytopes

Here we describe and define the notion of a convex polytope using the convex hull definition. There are other, equivalent, definitions of polytopes but we will not have need for them.

$$\text{Conv} \left(\begin{array}{c} \circ \\ \circ \end{array} \right) = \text{---}$$

$$\text{Conv} \left(\circ \right) = \circ$$

Figure 1: Here are two simple examples of taking the convex hull of a set of point.

$$\text{Conv} \left(\begin{array}{c} \circ \quad \circ \\ \circ \quad \circ \\ \circ \quad \circ \end{array} \right) = \text{---}$$

Figure 2: A less trivial example of a convex hull of points in \mathbb{R}^2 .

Definition 2.1 The convex hull of a set of vectors $V = \{v_1, \dots, v_k\}$ in \mathbb{R}^n is given by

$$\text{conv}(V) = \{x \in \mathbb{R}^n \mid \exists \lambda_1, \dots, \lambda_k \in \mathbb{R}_{\geq 0} \text{ for which } x = \sum_{i=1}^k \lambda_i v_i \text{ and } \sum_{i=1}^k \lambda_i = 1\}$$

where $\mathbb{R}_{\geq 0}$ denotes the non-negative real numbers. Each element of $\text{conv}(V)$ is called a convex combination of V . [Ewa96, Gru67]

That is, the convex hull of V is the set of all convex combinations of V . Illustrations of some sample convex hulls can be found in Figures 1 and 2.

Definition 2.2 P is a convex polytope if and only if there exist vectors $V = v_1, \dots, v_k$ such that $P = \text{conv}(V)$. [Ewa96]

Example 2.1 To get some intuition for what convex hulls look like, consider the case where $k = 2$. That is, we are taking the convex hull of just two points in \mathbb{R}^n .

$$\text{conv}(\{v_1, v_2\}) = \{x \in \mathbb{R}^n \mid \exists \lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0} \text{ for which } x = \lambda_1 v_1 + \lambda_2 v_2 \text{ and } \lambda_1 + \lambda_2 = 1\}$$

which is equivalent to saying

$$\text{conv}(\{v_1, v_2\}) = \{x \in \mathbb{R}^n \mid x = \lambda v_1 + (1 - \lambda)v_2 \text{ for } \lambda \text{ in } \mathbb{R} \text{ with } 0 \leq \lambda \leq 1\}$$

which is simply the parametric definition of a line segment with endpoints at v_1 and v_2 . That is, the convex hull of two points is just a line segment with those endpoints.

Definition 2.3 A polygon is a polytope in \mathbb{R}^2 .

That is, a polygon is the convex hull of a finite number of points in \mathbb{R}^2 .

Remark: A polygon has the same number of edges as vertices, as long as it is not just a single point. This can be proven by induction on the number of vertices, but is unenlightening and not necessary to this paper.

Example 2.2 *Triangles, quadrilaterals, and line segments are all polygons. Figures 1 and 2 show some polygons presented as convex hulls.*

We will now prove a lemma which is implied by the choice of terminology; convex hulls are convex. This result follows rather easily from the definitions of convexity and convex hull, but takes quite a bit of writing to work out rigorously. Careful proofs of fundamental properties of polytopes are vital to a solid understanding of later results.

Definition 2.4 *A set $S \subset \mathbb{R}^n$ is convex if and only if for any two points $x, y \in S$, the line segment \overline{xy} is contained in S . [Str89]*

Lemma 2.1 *The convex hull of a set of points is convex.*

Proof:

Let x and y be points in the set $P \subset \mathbb{R}^n$, such that

$$\begin{aligned} P &= \text{conv}(v_1, v_2, \dots, v_k) \\ x &= a_1 v_1 + a_2 v_2 + \dots + a_k v_k \\ y &= b_1 v_1 + b_2 v_2 + \dots + b_k v_k \end{aligned}$$

such that

$$a_i, b_i \in \mathbb{R}_{\geq 0}, \text{ for } 1 \leq i \leq k$$

$$\sum_{a=1}^k a_i = 1$$

$$\sum_{b=1}^k b_i = 1$$

Now consider the line segment l which connects x and y .

$$l = \text{conv}(x, y)$$

Any point z on l can be described by the equation

$$z = \lambda x + (1 - \lambda)y$$

for $0 \leq \lambda \leq 1$. We can rewrite that equation by substituting in our previous definitions of x and y .

$$\begin{aligned} z &= \lambda(a_1 v_1 + a_2 v_2 + \dots + a_k v_k) \\ &\quad + (1 - \lambda)(b_1 v_1 + b_2 v_2 + \dots + b_k v_k) \end{aligned}$$

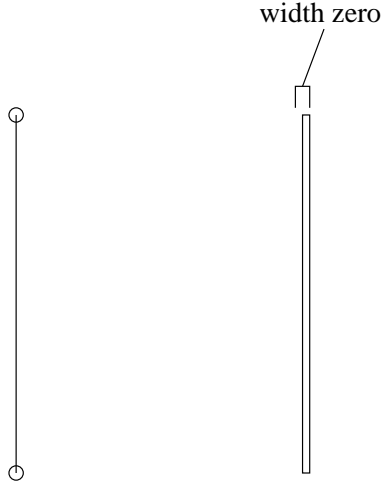
Examining the coefficients, we find that

$$\begin{aligned} &\lambda a_1 + \lambda a_2 + \dots + \lambda a_k + (1 - \lambda)b_1 + (1 - \lambda)b_2 + \dots + (1 - \lambda)b_k \\ &= \lambda(a_1 + a_2 + \dots + a_k) + (1 - \lambda)(b_1 + b_2 + \dots + b_k) \\ &= \lambda(1) + (1 - \lambda)(1) \\ &= 1 \end{aligned}$$

which means that z is in the convex hull of v_1, \dots, v_k , which is just P . Since $z \in P$ for any z on l , $l \subset P$ and P is convex.

Remark: Under our definitions, all polytopes are convex polytopes. We will often repeat the fact that our polytopes are convex for the sake of clarity, but in this paper all polytopes and polygons are convex unless explicitly stated otherwise.

Remark: As we observed earlier, the convex hull of a pair of points is a line segment. For our purposes, we will think of line segments as degenerate rectangles. That is, they are rectangles with zero width but which still have 2 edges. This representation is shown in Figure 3, and will become important in Section 2.7 when we talk about edge sets. For now, we will continue with the background.



A line segment seen as the convex hull of two points.

A line segment seen as a degenerate rectangle.

Figure 3: A line segment can be seen as a degenerate rectangle.

2.2 Minkowski Sums

Now we will describe a means by which we can “add” together two polytopes. That is, how we can take two polytopes and produce a third that behaves in a way that we would intuitively expect a sum to behave (commutativity, associativity, etc.). This notion of being able to add together subsets of \mathbb{R}^n builds the foundation upon which the rest of the paper is based.

Definition 2.5 For any two sets $P, Q \subset \mathbb{R}^n$, the Minkowski Sum of P and Q is defined to be

$$P + Q = \{p + q \mid p \in P, q \in Q\}$$

[Gru67]

Minkowski summation is sometimes called *vector summation* in older texts.

Lemma 2.2 For any two subsets of \mathbb{R}^n , $V = \{v_1, \dots, v_k\}$ and $U = \{u_1, \dots, u_m\}$,

$$\text{conv}(V) + \text{conv}(U) = \text{conv}(U + V)$$

That is,

Proof:

Let $P, Q \subset \mathbb{R}^n$ be polytopes defined by $P = \text{conv}(V)$ and $Q = \text{conv}(U)$ for some sets $V = \{v_1, \dots, v_k\}$ and $U = \{u_1, \dots, u_m\}$. We can write the sum as

$$P + Q = \{p + q \mid p \in P, q \in Q\}$$

For some scalar coefficients $a_1, \dots, a_k \in \mathbb{R}$ and $b_1, \dots, b_k \in \mathbb{R}$, we then have

$$\begin{aligned} \text{conv}(V) + \text{conv}(U) &= \{a + b \mid a \in \text{conv}V, b \in \text{conv}(U)\} \\ &= \{(a_1v_1 + \dots + a_kv_k) + (b_1u_1 + \dots + b_ku_m)\} \\ &= \{(a_1v_1 + \dots + a_kv_k + b_1u_1 + \dots + b_ku_m)\} \\ &= \{\sum_{i=1}^k \sum_{j=1}^m (\frac{a_i}{m} + \frac{b_j}{k})(v_i + u_j)\} \\ &= \{x \mid x \in \text{conv}(\{(v_i + u_j) \text{ for } 0 \leq i \leq k \text{ and } 0 \leq j \leq m\})\} \\ &= \{x \mid x \in \text{conv}(V + U)\} \end{aligned}$$

And so we see that the set of all convex polytopes is closed under Minkowski summation. That is, the Minkowski sum of two polytopes is itself a polytope.

Example 2.3 If Q is any polytope and P is a single point polytope, then $P + Q$ is just a translation of Q by the vector P . Performing the Minkowski sum in this case means adding the only point in P to each element of Q , hence shifting each point in Q by the vector equivalent of P . The shape of Q remains the same.

Remark: In this thesis, we work with the shape of the polytopes, not their position in space. Since any translation can be written as a Minkowski sum with a single point polytope, we will ignore translations of polytopes unless otherwise stated. When we refer to a polytope P , we are really referring to an arbitrary translation of P .

Example 2.4 Let P be any polytope, and let Q be a line segment. To visualize the Minkowski sum $P + Q$, recall that adding P to any one of the points in Q is just a translation of P . Thus adding P to each of the points in Q will place multiple (overlapping) translated copies of P .

Lemma 2.3 For any two convex polytopes P and Q ,

$$P + Q = \bigcup_{q \in Q} (P + q)$$

This lemma is nothing more than a rephrasing of the previous definition of the Minkowski sum, but it will be useful to have in mind in Section 2.3 when we prove the Stamping Algorithm (Algorithm 2.1).

We will now prove some other nice facts about Minkowski sums, and show that they fit our intuition for how addition “should” behave.

Lemma 2.4 Minkowski summation is associative. That is, for any polytopes P , Q , and R , we have

$$(P + Q) + R = P + (Q + R)$$

Proof:

This lemma follows directly (and almost trivially) from Definition 2.5 and the associativity of “+” over \mathbb{R} .

$$\begin{aligned} (P + Q) + R &= P + \{q + r \mid q \in Q, r \in R\} \\ &= \{p + (q + r) \mid p \in P, q \in Q, r \in R\} \\ &= \{(p + q) + r \mid p \in P, q \in Q, r \in R\} \\ &= \{p + q \mid p \in P, q \in Q\} + R \\ &= P + (Q + R) \end{aligned}$$

Lemma 2.5 Minkowski summation is commutative. That is, for any polytopes P , Q , and R , we have

$$P + Q = Q + P$$

Proof:

This lemma follows easily from Definition 2.5 and the commutativity of “+” over \mathbb{R} .

$$\begin{aligned} P + Q &= \{p + q \mid p \in P, q \in Q\} \\ &= \{q + p \mid p \in P, q \in Q\} \\ &= Q + P \end{aligned}$$

Intuitively, both associativity and commutativity are inherited from addition over the reals.

Remark: It is impossible for any paper to discuss Minkowski sums without at least touching upon the study of zonotopes.

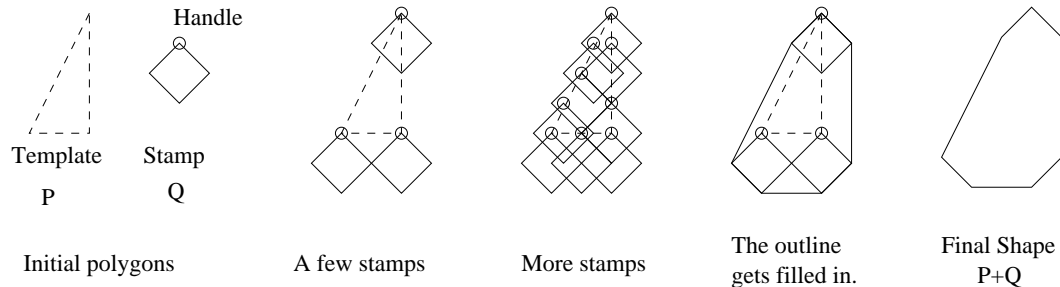


Figure 4: A demonstration of the stamping algorithm.

Definition 2.6 A zonotope is a Minkowski sum of line segments.

Example 2.5 A line segment is a trivial zonotope. A hexagon is a less trivial example, and can be seen in Figure 13.

Zonotopes have many interesting properties and invariants, most of which are outside the scope of this paper. However interested readers will find discussions of Zonotopes in almost any text on convex polytopes. Some good starting points are the relevant sections of [Ewa96], [Zie95], [McM71], and [She74]. In the context of this paper, zonotopes will just be useful examples and not any different from other polytopes.

2.3 The Stamping Algorithm

Here we provide a geometric algorithm for calculating the Minkowski sum of two polytopes. This algorithm will be useful for quickly producing examples as well as providing an intuition for what is going on. It is a rigorous expansion of the algorithm briefly sketched by Ziegler in [Zie95].

Algorithm 2.1 Given two convex polytopes P, Q , we can calculate the Minkowski sum $P + S$ as follows:

- (1) Choose one of the polygons to be the template. Without loss of generality, choose P .
- (2) The other polytope, Q , will be the stamp.
- (3) Choose a point to be the handle of the stamp. Note that this handle does not have to be on the stamp.
- (4) Pick up the stamp by the handle, and place a copy of the stamp down for every possible placement of the handle within the template.

If we choose the handle to be the origin, then the resulting set of points will be the exact Minkowski sum $P + Q$. If we choose any other point as a handle, then we will still get the same shaped result, but will differ by a translation. In this thesis, we are concerned with shape (not translation), so we usually pick a handle that is more convenient than the origin (such as a vertex of the stamp).

Example 2.6 A demonstration of the stamping algorithm is given in Figure 4. Figure 5 demonstrates that reversing the choice of stamp and template does not change the result of the algorithm. Figures 6 and 7 show a similar demonstration that the handle choice is irrelevant.

Remark: Since Minkowski summation is associative (Theorem 2.4), an algorithm which allows us to calculate a 2-part sum will also let us calculate a sum with any number of summands.

Some examples will give intuition for the algorithm and for how Minkowski sums behave.

Example 2.7 Let P be an equilateral triangle and Q be a single point polygon. Their sum, $P + Q$, is shown in Figure 8.

This demonstrates how adding a single point to a polytope translates the polytope, but does not change its shape. Since we are only concerned with shapes of polytopes in this thesis, we consider single-point additions to have no effect at all.

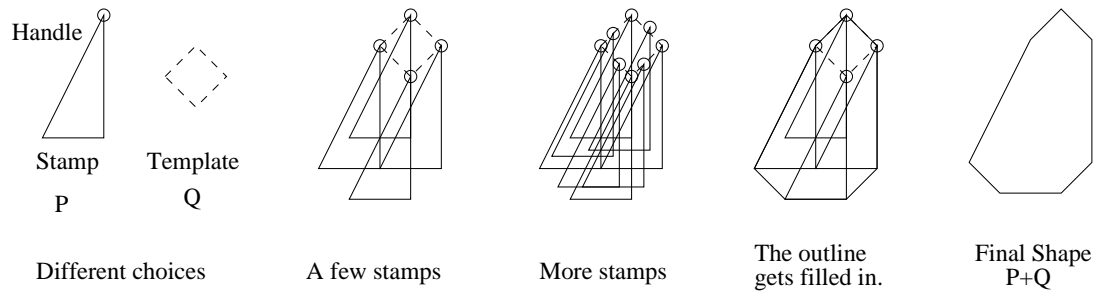


Figure 5: Choosing the stamp and template differently does not affect the outcome.

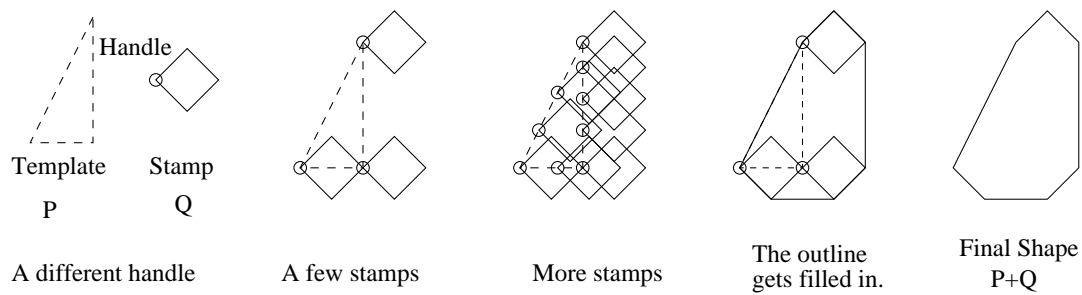


Figure 6: Choosing a different handle does not affect the outcome.

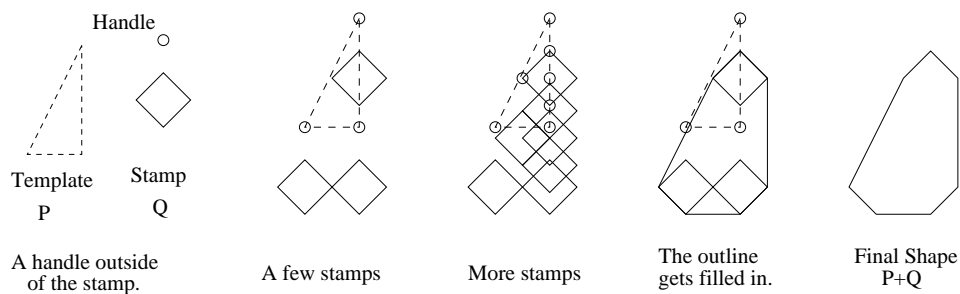


Figure 7: Choosing a handle outside of the stamp is acceptable.

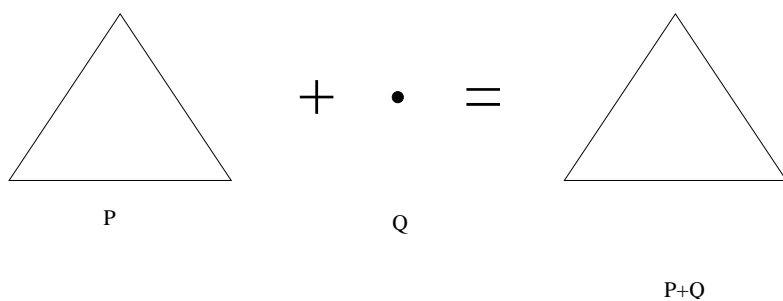


Figure 8: Example of a Minkowski sum acting as a translation.

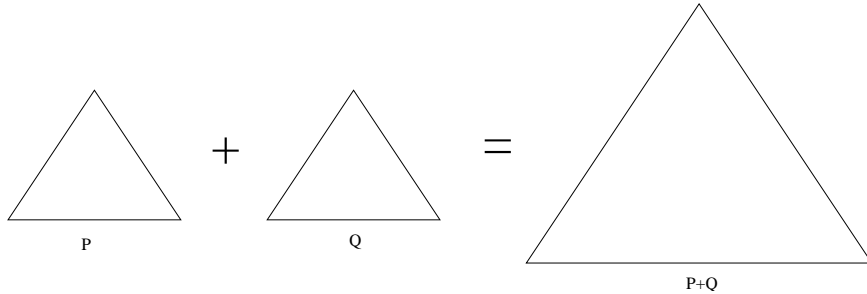


Figure 9: Example of a Minkowski sum acting as a scaling.

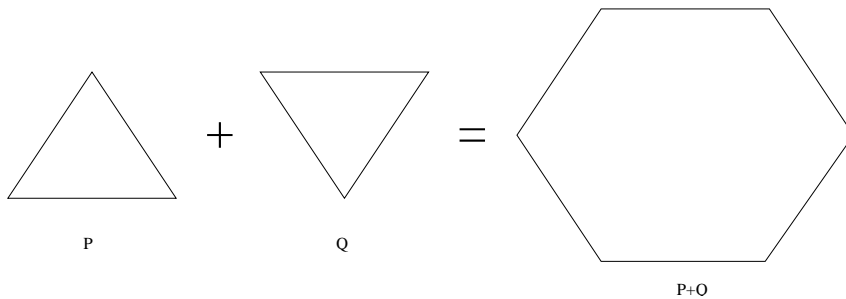


Figure 10: Example of a Minkowski sum.

Example 2.8 Let P be an equilateral triangle, and Q be a copy of P . Their sum, $P + Q$, is shown in Figure 9.

This demonstrates how adding a polytope to itself produces a scaled version of that polytope. See Lemma 2.7 for a proof of this fact.

Example 2.9 Let P be an equilateral triangle, and Q be the same triangle but put up-side-down. Their sum, $P + Q$, is the regular hexagon shown in Figure 10. This example demonstrates that rotating the summands greatly influences the resulting polytope. We explore some similar results in Section 7.

Example 2.10 Let P be an equilateral triangle, and Q be a vertical line segment. Their sum, $P + Q$, is the “house” shape shown in Figure 11.

Example 2.11 Let P be a right triangle, and Q be the same triangle but flipped about the y -axis. Their sum, $P + Q$, is the “house” shape shown in Figure 12.

Notice that the Minkowski sums in Examples 2.10 and 2.11 are the same, even though the summands are different. This phenomena of different set of polytopes summing to the same polytope is the motivation

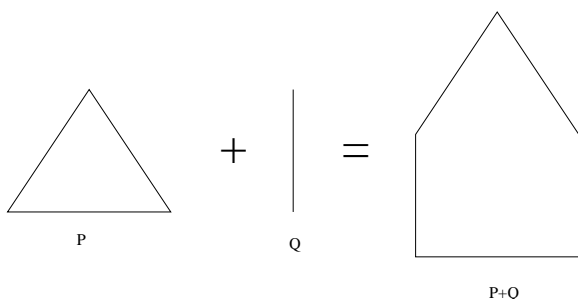


Figure 11: Example of a Minkowski sum.

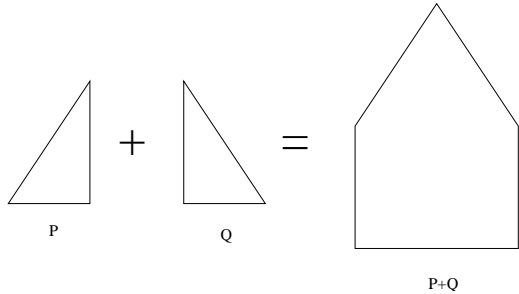


Figure 12: Example of a way to get the same Minkowski sum as in Example 11 but with different summands.

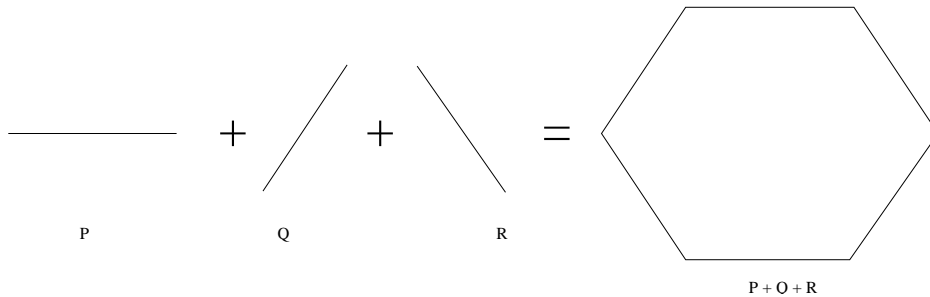


Figure 13: A different set of polygons that sum to a hexagon. Since these summands are all line segments, we know that regular hexagons are zonotopes.

for this paper. This non-uniqueness is such a vital result that we will show another example of it. We will see this idea again in Section 2.5.

Example 2.12 *Let P , Q , and R be non-parallel line segments in \mathbb{R}^2 . Their sum, $P + Q + R$ is a hexagon. Furthermore, when properly chosen, they are exactly the same hexagon as the one pictured in Figure 10, as can be seen in Figure 13.*

Now that we have gained some intuition for the stamping algorithm, we turn to a formal proof of its correctness.

Theorem 2.1 *The stamping algorithm applied to two polytopes P and Q produces their sum, $P + Q$.*

Proof: Recall Lemma 2.3, which states that

$$P + Q = \bigcup_{q \in Q} (P + q)$$

The stamping algorithm exactly performs the union described in that Lemma. In this interpretation, P is the stamp and Q as the template. Assume we choose $\vec{0}$, the origin, to be the handle of P . This will get us that $P + q$ equals the image of P when the handle is placed at q . Each term of the union, $P + q$, corresponds to the act of creating an image of P with the handle of P placed at q . Performing the union on those terms corresponds to taking the union of all the images.

Choosing an arbitrary handle, $h \in \mathbb{R}^n$, is equivalent to adding the point h to $P + Q$. Doing so is just a translation and thus can be ignored. Thus we can choose any handle we want, and the stamping algorithm will still generate a result with the same shape and size as the true Minkowski sum.

Remark: Recall from Lemma 2.2 that $\text{conv}(U + V) = \text{conv}(U) + \text{conv}(V)$. Thus, if we choose a vertex as a handle, we can just stamp the handle onto each vertex of the template and take the convex hull of the result. This method is reflected in the examples of this section, and its running time complexity is analyzed in Section 11.5.

2.4 Scaled Multiples of Polytopes

In this section we present the notion of a scaled polytope and verify some elementary notions about them. That is, we will specify what we really mean when we write something like “ $2P$ for a polytope P ”. We verify that the notationally obvious statement “ $P + P = 2P$ ” really is true under our particular meaning of “+” and “ $2P$ ”. Some readers may wish to simply accept that statement as true and move on to the more important sections of the paper.

Definition 2.7 For any polytope $P \subset \mathbb{R}^n$, we can talk about scaled multiples of P , written kP , for any scalar $k \in \mathbb{R}$. Quite simply,

$$kP = \{kp \mid p \in P\}$$

[Gru67]

Example 2.13 For $k = 0$, the scaled polytope kP is a single point polytope (the origin, $\vec{0}$) independent of P .

Lemma 2.6 Scaled multiples of polytopes are themselves polytopes, as the name suggests.

Proof:

Let P be a polytope. By definition, $P = \text{conv}(V)$ for some set of vectors $V = \{v_1, \dots, v_k\}$. Thus we can write $2P$ as

$$\begin{aligned} 2P &= \{x \in \mathbb{R}^n \mid x = 2p, p = (a_1v_1 + \dots + a_kv_k) \in \text{conv}(V)\} \\ &= \{x \in \mathbb{R}^n \mid x = 2(a_1v_1 + \dots + a_kv_k)\} \\ &= \{x \in \mathbb{R}^n \mid x = (a_1v_1 + \dots + a_kv_k) + (a_1v_1 + \dots + a_kv_k)\} \\ &= \{x \in \mathbb{R}^n \mid x = p + p \in P + P\} \\ &= \{x \in \mathbb{R}^n \mid x \in P + P\} \\ &= P + P \end{aligned}$$

for some scalar coefficients $a_1, \dots, a_k \in \mathbb{R}$.

Remark: Once we have introduced scalar multiples of polytopes, we are obligated to find out if they behave the way scalar multiples do in other domains. That is, is it true that $P + P = 2P$, as is the case for \mathbb{R} and \mathbb{R}^n ?

Notation: By “ $2P$ ”, we mean the dilation of P by a factor of 2. That is,

$$2P = \{p \in \mathbb{R}^n \mid \frac{p}{2} \in P\}$$

Lemma 2.7 If $P \subset \mathbb{R}^n$ is a polytope, then $P + P = 2P$.

Proof:

We will prove that $2P = P + P$ by showing that both $2P \subset P + P$ and $P + P \subset 2P$ hold.

Consider an arbitrary point $x \in P$. By definition

$$\begin{aligned} x + x &\in P + P \\ 2x &\in P + P \end{aligned}$$

and thus $2P \subset P + P$. Now consider a point

$$x + y \in P + P$$

for $x, y \in P$. Let s be the line segment

$$s = \text{conv}(x, y) \in P$$

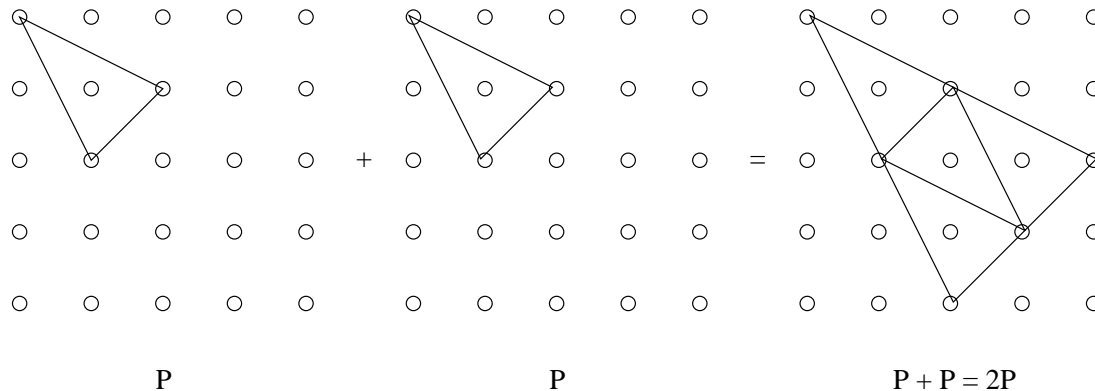


Figure 14: An illustration that $P + P = 2P$.

Since P is convex, $s \subset P$ and midpoint of s is contained in P .

$$\frac{x + y}{2} \in P$$

from which it follows that

$$\frac{x+y}{2} + \frac{x+y}{2} \in P + P$$

$$2\frac{x+y}{2} \in P$$

$$x + y \in P + P$$

and thus $P + P \subset 2P$.

Example 2.14 An illustration of Lemma 2.7 can be found in Figure 14.

Remark: Since Minkowski summation is associative (as we saw in Lemma 2.4), Lemma 2.7 implies the more general result that $\sum_{i=1}^k P = kP$.

2.5 Decompositions of Polytopes

Definition 2.8 A decomposition of a convex polytope P is a set of convex polytopes $S = \{s_1, \dots, s_l\}$ such that $\sum_{i=1}^l s_i = P$. The act of decomposing a polytope refers to finding all such sets which sum to P .

Remark: The sums shown in Examples 11 and 12 and in Examples 10 and 13 demonstrate that Minkowski sum decompositions are not always unique.

Definition 2.9 A polytope P is indecomposable if and only if, for any decomposition $P = Q + R$, the polytopes Q and R are always either scaled versions of P or single points.

Remark: In Section 6, we prove that the indecomposable polygons are triangles, line segments, and single points. In Section 5 we provide an algorithm for finding a decomposition of any polygon into triangles and line segments.

2.6 Hyperplanes, Supporting Hyperplanes, and Faces

In this subsection we present the formal definition of a *hyperplane* in \mathbb{R}^n and the closely related notion of a *face* of a polytope. They are all standard definitions and equivalent ones can be found in almost any text about linear algebra or convex polytopes.

Definition 2.10 A hyperplane in \mathbb{R}^n , H , is a subset of \mathbb{R}^n such that

$$H = \{x \in \mathbb{R}^n \mid \langle x, u \rangle = \alpha\}$$

for some fixed vector $u \in \mathbb{R}^n$ and some fixed value $\alpha \in \mathbb{R}$. Here, u is the normal vector of H , α is the constant term of H , and $\langle \bullet, \bullet \rangle$ represents the standard dot product. [McM71]

This is just a plane in \mathbb{R}^3 or a line in \mathbb{R}^2 , but generalized to \mathbb{R}^n . A hyperplane provides us with a natural way to divide up \mathbb{R}^n into two “halves”; one “above” H and the other “below” H .

Definition 2.11 A hyperplane H in \mathbb{R}^n implicitly defines two half spaces of \mathbb{R}^n , called H^+ and H^- .

$$H^+ = \{x \in \mathbb{R}^n \mid \langle x, u \rangle \geq \alpha\}$$

$$H^- = \{x \in \mathbb{R}^n \mid \langle x, u \rangle \leq \alpha\}$$

[McM71, Gru67]

Notice that the inequalities are not strict. For this paper, we will define both half spaces to include H , since doing so will make later proofs and definitions cleaner.

Remark: We can now observe two elementary facts about half spaces

$$\begin{aligned} H^+ \cap H^- &= H \\ H^+ \cup H^- &= \mathbb{R}^n \end{aligned}$$

Definition 2.12 A hyperplane H is said to be a supporting hyperplane of a polytope P in \mathbb{R}^n if and only if $H \cap P \neq \emptyset$ and either $P \subset H^+$ or $P \subset H^-$. [McM71]

Without loss of generality, we will always assume that P lies in H^+ . We will refer to this simplification as the *positive support assumption*.

Definition 2.13 For any convex polytope P , $F \subset P$ is a face of P if and only if there is some supporting hyperplane of P for which $F = P \cap H$. By convention, P and \emptyset are improper faces of P . [Ewa96, Zie95]

Intuitively, this notion of a face is simply a formal, generalized definition of what we normally think of as “faces” of polytopes in \mathbb{R}^2 and \mathbb{R}^3 . A polygon’s vertices and edges are faces. A polyhedra’s vertices, edges, and surfaces are all faces. The supporting hyperplane for any face is a hyperplane which has been pushed up against that face as far as it can go.

Definition 2.14 The vertices of a polytope P are the zero dimensional faces of P . The edges of P are the one dimensional faces of P .

Notation: The length of an edge E is denoted $|E|$.

Example 2.15 Revisiting the running example of the “house polygon”, some sample supporting hyperplanes and their corresponding faces are shown in Figure 15.

Remark: For any hyperplane H and any polytope P , there exist unique hyperplanes I and J parallel to H such that I supports P with $P \subset I^+$ and J supports P with $P \subset J^-$. That observation motivates the following definition.

Definition 2.15 Let H be a hyperplane in \mathbb{R}^n and F be a face of a polytope $P \subset \mathbb{R}^n$. We say that H positively supports F up to translation if and only if there exists a vector $x \in \mathbb{R}^n$ such that $H \cap P + x = F + x$ and $P + x \subset H^+$.

Remark: For any hyperplane H and any polytope P , both in \mathbb{R}^n , H positively supports exactly one face of P up to translation.

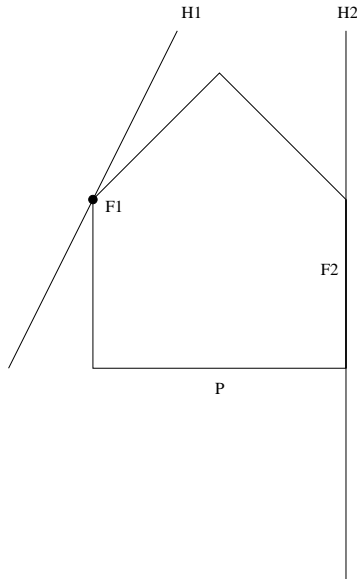


Figure 15: Some examples of supporting hyperplanes of the “house polygon”.

$$\text{edgeset}\left(\begin{array}{|c} \\ \\ \end{array}\right) = \left\{ \begin{array}{|c} \\ \\ \end{array}, \begin{array}{|c} \\ \\ \end{array} \right\}$$

Figure 16: The edge set of a line segment contains two edges, since a line segment is treated as a degenerate rectangle.

2.7 Edge Sets

Here we introduce some definitions to formalize the notion of the *edges* of a polygon. Notice that in this section we are only working in \mathbb{R}^2 and thus are dealing with polygons rather than general polytopes.

Definition 2.16 An edge-vector e of a polygon P is the vector parallel to some edge E of P with magnitude $|E|$. The direction of e is always chosen so that it points counterclockwise around P .

The notion of notion of an edge-vector allows us to have the edges of P “Remember” their orientation in P .

Remark: If a polygon P has two parallel edges of equal length, the corresponding edge-vectors are still distinguishable by their direction.

Definition 2.17 Given a polygon $P = \text{conv}(V)$ with $V = \{v_1, \dots, v_k\}$, the edge set of a polygon P is the set of all the edge-vectors of P .

Notation: Let P be a polygon. The set of all edges of P is denoted $\text{edges}(P)$. The set of all edge-vectors of P is denoted $\text{edgeset}(P)$.

Example 2.16 A single point polygon has an empty edge set.

Example 2.17 Recall that a polygon which is a line segment can be treated as a degenerate rectangle; its edge set contains two edge-vectors which are the same line segment but opposite orientations. This example came up earlier in Figure 3, and is shown again in Figure 16 in the language of edge sets.

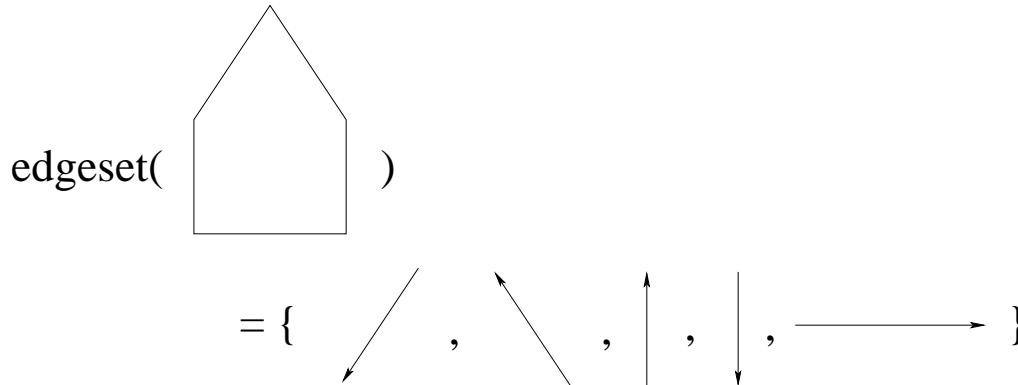


Figure 17: The edge set of the “house” polygon.

Example 2.18 A more complicated example is shown in Figure 17. The “house polygon” shown there is a useful example which reappears throughout this paper.

Definition 2.18 Let S and T be sets of edge-vectors, which is allowed to have duplicate entries. We say that S is a slice of T if and only if for every $s \in S$ there exist $t_1, \dots, t_k \in T$ such that $s = \sum_{i=1}^k t_i$ under Minkowski summation.

Definition 2.19 A sliced edge set of a polygon P is any slice of the edge set of P .

Intuitively, a sliced edge set is the result of taking the edge set of P and breaking up any number of the edge-vectors into smaller edge-vectors. These smaller edges-vectors retain the same orientation as the original ones, and may be duplicated within the set.

Lemma 2.8 Any set of vectors V in \mathbb{R}^2 which sum to $\vec{0} = (0, 0)$ correspond to a sliced edge set of some convex polygon.

Proof:

We offer a constructive proof of the lemma. Place the elements of V tail of each at the tip of the previous. Since the vectors sum to zero, they form a closed shape.

Let E be the set of vectors $\{e_1, \dots, e_n\}$ such that $\sum_{i=1}^k e_i = 0$. Let them be indexed in order by their angle up from standard position. Suppose the shape those vectors form, when taken as oriented edges, is non-convex. It is elementary that there must then be some pair of vectors e_i, e_{i+1} such that the interior angle between them is greater than π radians. That implies that the angle of e_{i+1} is strictly less than the angle of e_i , which is a contradiction. Hence E defines a convex polygon.

Remark: Furthermore, by this construction we see that the polygon determined by V is unique up to translation.

3 Minkowski Sums of Faces

In this section we prove a nice fact which will be useful in later proofs. It is perhaps more of a lemma, but we give it full theorem status due to the length of its proof and value of the result. The proof is closely related the corresponding (but less general) proof for zonotopes given by McMullen in [McM71].

Theorem 3.1 Let P be a convex polytope with a decomposition

$$P = \sum_{i=1}^k P_i$$

Let H be any hyperplane. Let F be the face of P positively supported by H up to translation. Let F_i be the face of P_i positively supported by H up to translation.

Then

$$F = \sum_{i=1}^k F_i$$

Proof:

Let H be the supporting hyperplane of F , meaning $F = H \cap P$ and $P \subset H^+$. For $1 \leq i \leq k$, let H_i be the hyperplane that supports F_i . Notice that H and all H_i are parallel.

By the definition of a hyperplane, we have

$$\begin{aligned} H &= \{x \in \mathbb{R}^n \mid \langle x, u \rangle = \alpha\} \\ H_i &= \{x_i \in \mathbb{R}^n \mid \langle x_i, u \rangle = \beta_i\} \end{aligned}$$

for some fixed $\alpha, \beta_i \in \mathbb{R}$ and some normal vector u . By the definition of a face, we have

$$\begin{aligned} F &= \{x \in P \mid \langle x, u \rangle = \alpha\} \\ F_i &= \{x_i \in P_i \mid \langle x_i, u \rangle = \beta_i\} \end{aligned}$$

Since $P \subset H^+$ and $P_i \subset H_i^+$, we have

$$\begin{aligned} \forall x \in P, \langle x, u \rangle &\geq \alpha \\ \forall x_i \in P_i, \langle x_i, u \rangle &\geq \beta_i \end{aligned}$$

In fact,

$$\begin{aligned} \forall x \in P - F, \langle x, u \rangle &> \alpha \\ \forall x_i \in P_i - F_i, \langle x_i, u \rangle &> \beta_i \end{aligned}$$

where the minus sign signifies set subtraction. We can now make the following sequence of deductions

$$\begin{aligned} \sum_{i=1}^k \beta_i &= \sum_{i=1}^k \langle y_i, u \rangle \\ &= \langle \sum_{i=1}^k y_i, u \rangle \\ &= \langle y, u \rangle \\ &= \alpha \end{aligned}$$

which get us that

$$\sum_{i=1}^k \beta_i = \alpha$$

Now we are ready to prove that

$$F = \sum_{i=1}^k f_i$$

using the classing set theory proof of equality via mutual inclusion. This requires two steps, each of which we will prove by contradiction.

Step I: First I will prove that $F \subset \sum_{i=1}^k f_i$ by contradiction. If this were not the case, then there would be some point $z \in F$ such that

$$z = \sum_{i=1}^k z_i \in P_i$$

where at least one z_i is not in f_i . Without loss of generality, assume that z_1 is not in f_1 . Thus we know that

$$\langle z_1, u \rangle = \gamma > \beta_1$$

and that for all other values of i ,

$$\langle z_i, u \rangle \geq \beta_i$$

Consider:

$$\begin{aligned} \sum_{i=1}^k \langle z_i, u \rangle &= \langle z_1, u \rangle + \sum_{i=2}^k \langle z_i, u \rangle \\ &\geq \gamma + \sum_{i=2}^k \beta_i \\ &> \sum_{i=1}^k \beta_i \\ &> \alpha \end{aligned}$$

But if $z = \sum_{i=1}^k \langle z_i, u \rangle > \alpha$ then z is not in F , which is a contradiction. Thus $F \subset \sum_{i=1}^k F_i$.

Step II: Now we will prove that $F \supset \sum_{i=1}^k F_i$ by a similar contradiction. If this were not the case, then there would be some collection of points $z_i \in P_i$ for $1 \leq i \leq k$ such that

$$z = \sum_{i=1}^k z_i \in P_i$$

where z is not in F . Consider:

$$\begin{aligned} \alpha &= \sum_{i=2}^k \beta_i \\ &= \sum_{i=1}^k \langle z_i, u \rangle \\ &= \langle \sum_{i=1}^k z_i, u \rangle \\ &= \langle z, u \rangle \end{aligned}$$

But that implies that $z \in F$ which is a contradiction. Thus $F \supset \sum_{i=1}^k F_i$.

Together with **Step I**, this implies that

$$F = \sum_{i=1}^k F_i$$

thus concluding the proof.

3.1 Faces of Sums

We are now equipped to answer another salient question. Consider any set of polytopes P_1, \dots, P_k and set of points p_1, \dots, p_k such that $p_i \in F_i$ where F_i is a face of P_i . Does the point

$$p = \sum_{i=1}^k p_i$$

lie on a face of the polytope

$$P = \sum_{i=1}^k P_i$$

and if so, what face? We can answer that question by applying Theorem 3.

Lemma 3.1 *If there exists a hyperplane H such that H positively supports each F_i up to translation, then $p \in F$ where F is the face of P positively supported by H .*

4 Relating Minkowski Summation to Edge Sets

In this section, we consider the task of determining whether or not a given set of polygons S sum to a given polygon $P \subset \mathbb{R}^n$. Note that once again we are restricting ourselves to \mathbb{R}^2 , as this will allow us to prove some more powerful results. If and how these results extend to higher dimensions is an open problem. Section 4.1 shows that we can identify equivalent polygons just by looking at their edge sets. Section 4.2 uses that result and allow us to use edge sets to identify decompositions.

4.1 Edge Sets of Equivalent Polygons

Here we will prove a theorem that will enable us to identify equivalent polygons just by looking at their edge sets. We will achieve that result by proving the equivalence of four properties of pairs of polygons. First, however, we need some additional background.

Definition 4.1 *Two convex polytopes, P and Q , are identical up to translation if and only if $P = Q + x$ for some vector x .*

Definition 4.2 *The corners of a polytope P are the non-empty intersections of edges of P . The set of corners of P is denoted $\text{corners}(P)$.*

Lemma 4.1 *Every polytope is the convex hull of its corners.*

$$P = \text{conv}(\text{corners}(P))$$

Lemma 4.2 *The vertices (zero dimensional faces) of a polytope P are exactly the corners of P .*

I omit the proof, since this result is intuitive in \mathbb{R}^2 , and since the full proof is long and unenlightening. Interested readers can find a formal proof of this fact for general polytopes in [Zie95]. In fact, the vertices, corners, extreme points, and exposed points of a polytope are all equivalent notions, and taking the convex hull of any of them yields P . [Gru67]

Notation: For any set of objects $S \subset \mathbb{R}^n$ and any vector $x \in \mathbb{R}^n$, we use $S + x$ to denote the same set translated by x .

Theorem 4.1 *For any polygons $P, Q \in \mathbb{R}^2$, the following four facts are equivalent.*

- (a) $\text{edgeset}(P) = \text{edgeset}(Q)$
- (b) $\exists x \in \mathbb{R}^2 \mid \text{edges}(P) = \text{edges}(Q) + x$
- (c) $\exists x \in \mathbb{R}^2 \mid \text{corners}(P) = \text{corners}(Q) + x$
- (d) $\exists x \in \mathbb{R}^2 \mid P = Q + x$

These are all equivalent ways of saying that P and Q are equivalent up to translation.

Proof: (a \Leftrightarrow b)

This connection follows directly from Definition 2.16, the definition of an edge-vector. The edge-vectors of a polygon record the same information as the edges, except for translation (which is lost).

Proof: (a \Rightarrow b)

Recall from Lemma 2.8 that any set of vectors which sum to $\vec{0}$ determines form the edge set of some polygon, which is uniquely determined up to translation.

Proof: (b \Leftrightarrow c)

The edges of P can be uniquely determined by walking around the corners of P in counterclockwise order and reading off the convex hulls of consecutive corners. Clearly, any translation of the corners produces the same translation of the edges.

Proof: (b \Rightarrow c)

The corners of P can be uniquely determined by walking around the edges of P in counterclockwise order and reading off their intersections. Clearly, any translation of the edges produces the same translation of the corners.

Proof: (c \Leftarrow d)

The corners of a polytope P are contained in P . Translating P by x means that we are translating every point in P by x , and thus we are also translating the corners by x .

Proof: (c \Rightarrow d)

By Lemma 4.1, a polygon can be uniquely expressed as the convex hull of its corners. Let $C = \{C_1, \dots, C_j\}$ be the set of corner of P , and let $K = \{K_1, \dots, K_j\}$ be the set of corners of Q . By assuming (c), we know that there is some $x \in \mathbb{R}^2$ such that $K_i = C_i + x$ for any $0 \leq i \leq j$.

By the definition of convex hull, any $q \in Q$ can be written as $q = \sum_{i=1}^j a_i K_i$ for $a_i \geq 0$, $\sum_{i=1}^j a_i = 1$. Consider the following manipulations:

$$\begin{aligned} Q &= \sum_{i=1}^j a_i K_i \\ &= \sum_{i=1}^j a_i (C_i + x) \\ &= \sum_{i=1}^j (a_i C_i + a_i x) \\ &= \sum_{i=1}^j a_i C_i + \sum_{i=1}^j a_i x \\ &= \sum_{i=1}^j a_i C_i + x \\ &= P + x \end{aligned}$$

Remark: The accompanying *Mathematica* program “conversion.nb”, which converts among different representations of polygons, relies heavily upon these equivalences. The algorithms for doing so are discussed in detail in Section 11.

Corollary 4.1 *Two polygons $P, Q \subset \mathbb{R}^n$ are identical up to translation if and only if $\text{edgeset}(P) = \text{edgeset}(Q)$.*

4.2 Edge Sets and Decompositions

Theorem 4.2 - Decomposition Theorem:

Let $P \subset \mathbb{R}^2$ be a convex polygon, and let $S = \{P_1, P_2, \dots, P_k\} \subset \mathbb{R}^2$ be a set of convex polygons. Let $Q = \sum_{i=1}^k P_i$. The edge set of S is some sliced edge set of P if and only if $P = Q$ up to translation.

In Section 11, we make use of this result to produce an efficient algorithm for performing Minkowski sums.

Algorithm 4.1 *Given a set of edge-vectors V , we will construct a new set of edge vectors, V' , as follows: Group together all sets of edge-vectors in V with the same orientation. Replace each such group with a new edge-vector which has the same orientation as the old ones and has length equal to the sum of their lengths.*

Remark: Observe that if we perform Algorithm 4.1 on a sliced edge set of a polygon that we produce the (non-sliced) edge set of that polygon. This follows directly from the definition of a sliced edge set, Definition 2.19.

Proof: (\Rightarrow)

Let E_S be some sliced edge set of P which is equal to $\text{edgeset}(S)$. We need to show that $P = Q$.

Construct a new set of edges, A , by performing Algorithm 4.1 on E_S . Similarly construct B from $\text{edgeset}(S)$. Since $E_S = U$ and since Algorithm 4.1 is deterministic, we know that $A = B$. Notice (from the above remark) that A is the (non-sliced) edge set of P and B is the (non-sliced) edge set of Q . Since $A = B$, Corollary 4.1 tells us that $P = Q$.

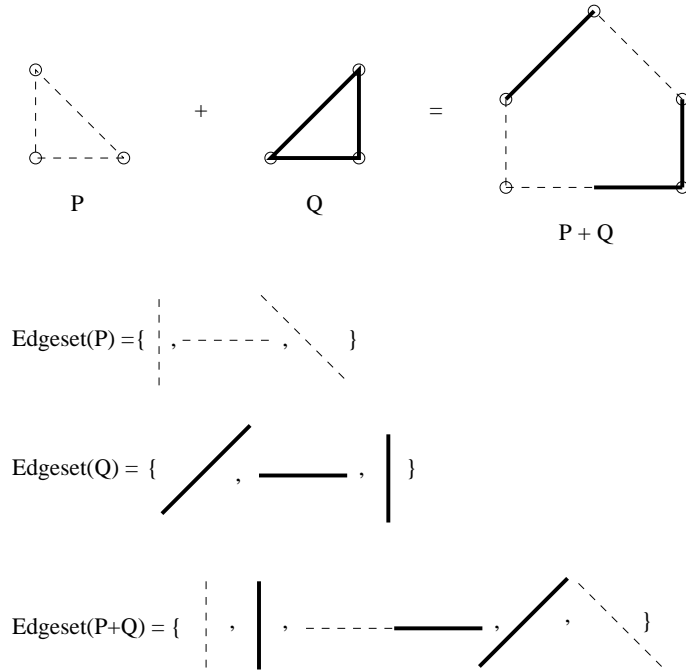


Figure 18: Some sliced edge set of $P+Q$ is the union of the edge sets of P and Q .

Proof: (\Leftarrow)

Given $P = \sum_{i=1}^k P_i$, we need to show that $edgeset(S)$ is some sliced edge set of P . This is equivalent to showing that performing Algorithm 4.1 on $edgeset(S)$ produces $edgeset(P)$. Since edges are faces, this result follows directly from Theorem 3.1.

Example 4.1 Two demonstrations of this result, using our favorite “house” polygon, can be seen in Figures 19 and 18.

Definition 4.3 Two sets of polytopes $S = P_1, \dots, P_s$ and $T = Q_1, \dots, Q_t$ are equivalent under Minkowski summation if and only if

$$\sum_{i=1}^s P_i = \sum_{i=1}^t Q_i$$

Remark: In this manner, Minkowski summation provides equivalence classes for the set of all sets of polytopes.

Corollary 4.2 Two sets of polygons S and T are equivalent under Minkowski summation if and only if there exists a sliced edge set of S which is identical to some sliced edge set of T .

5 Decomposition Algorithm

Using the theorems and techniques that we have developed in the preceding sections, we will now examine an algorithm for actually finding Minkowski sum decompositions of polygons. Note that we are not dealing with general polygons, and are restricting ourselves to \mathbb{R}^2 . We would like to be able to take any given polygon and find a set of indecomposable polygons whose Minkowski sum is the given polygon. Ideally, we would like to find *all* such sets of polygons, but we will see in Section 8 that there are some difficulties in doing that. In this section we present and prove the correctness of an algorithm which finds one decomposition of a polygon into triangles and line segments. Once we have this algorithm, we will be equipped to prove (in Section 6) that the indecomposable polygons are triangles, line segments, and points.

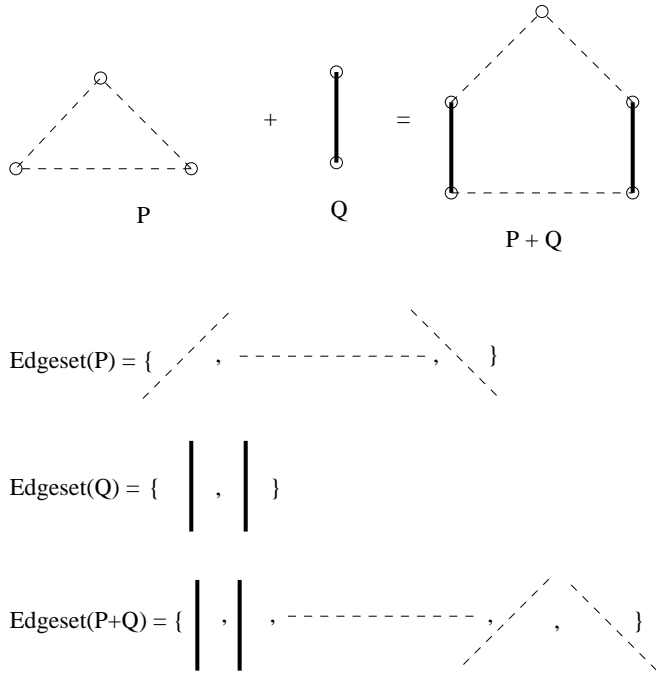


Figure 19: The edge set of $P+Q$ is the union of the edge sets of P and Q .

5.1 Description of the Algorithm

Here we present the algorithm in detail, but only give a high level justification for it. In Section 5.2, we give a low level proof that the algorithm eventually halts and finds a correct decomposition into triangles and line segments.

Algorithm 5.1 *Given a polygon $P \subset \mathbb{R}^n$ that is not a single point, we will find a set S of triangles and line segments such that the Minkowski sum of the elements of S equals P .*

$$\sum_{s_i \in S} s_i = P$$

We will do this through a recursive algorithm. Each time we run the algorithm on a polygon, it will decompose the polygon into two polygons that sum to the original one. By calling the algorithm recursively on both of the generated polygons, we can continue to break it down until we are left with only line segments and polygons. Proving that the algorithm is correct (that it always terminates and returns a valid decomposition) is left to Section 5.2.

Remark: In the description of the algorithm below, P refers to the polygon we are given to decompose, and p is the polygon that has most recently been handed to the algorithm.

Preprocessing:

Before running the recursive portion of the algorithm, we need to perform some preprocessing. In doing so, we will remove parallel edges from P , which is important to the correctness of the recursive algorithm. The edges we remove will show up as summands in the final result.

First identify all pairs of parallel edges - a procedure that will take time proportional to the number of edges of P . For each pair of parallel edges, identify the shorter one, e_0 . Shorten the length of each of those edges by $|e_0|$, the length of e_0 . Call this new polygon P' . Add e_0 to the set of summands. If both of the parallel edges are the same length, then reduce both of their edges to zero, but still only add one of them to the set of summands.

Once the preprocessing is complete, call the below algorithm on P .

Remark: Notice that for zonotopes, preprocessing will entirely decompose the polygon. After removing parallel edges, P will have been reduced to a single point plus some set of line segments. This is not a problem, since the first step of the recursive algorithm will catch that we are done and terminate the procedure.

Recursive Algorithm:

Given a polygon $p \subset \mathbb{R}^n$, do the following:

- (1) If p is a triangle, line segment, or a single point then stop. Add that polygon to the set of summands.
- (2) Choose an “expanding edge” e of p . That is, one such that the two vertices adjacent to e have angles that sum to more than π radians (180 degrees). Let $|e|$ represent the length of e .
- (3) Let H be the supporting hyperplane of e . Since we are working in \mathbb{R}^n , H will be the line that intersects p at e .
- (4) Let H' be a line parallel, but not identical, to H such that the length of $H' \cap p$ is $|e|$. Without loss of generality, we will assume that $H' \subset H^+$.
- (5) The hyperplane H' intersects the edge set of P in two places. If we slice $edgeset(P)$ at those two places (if necessary), then we see that H' partitions that sliced edge set of P into two sets of edge-vectors, s_1 and s_2 . Let s_1 be the partition containing e .
- (6) Let p_1 be the polygon defined by the edge set $s_2 \cup \{e\}$. And let p_2 be the polygon defined by the edge set $s_1 \setminus \{e\}$. We claim that $p_1 + p_2 = p$.
- (7) Run this procedure recursively on p_1 and p_2 . The algorithm will terminate after a finite number of steps, and will produce a valid Minkowski decomposition of P .

Figures 21 through 27 show Algorithm 5.1 at each of the 8 steps as it applied to two different polygons. The two running examples are shown together, but are not actually related in any way.

5.2 Correctness of the Algorithm

In Section 5.1, we gave a high level overview of Algorithm 5.1. Hopefully, its correctness is now intuitively clear. In this section we prove the correctness of the algorithm and give a formal treatment of each of its steps.

It turns out that there is (provably) no universal method for proving the correctness of algorithms, but in many cases it is possible to do so [KS01]. While proofs of theorems are sequences of justified steps, proofs algorithms are sequences of justifications that nothing was overlooked in each step. In order to be more confident that no error was overlooked, proofs of algorithm correctness are often painfully rigorous. In this section we have attempted to explain the need for each proofs, but several may still strike the reader as being unnecessary. We recommend that the reader use this section primarily as a reference for looking up specific parts of the proof that he or she feels need justification.

Definition 5.1 *Let A be an algorithm to calculate the result of some relation F over some domain D . A is correct if and only if, for any input $d \in D$, A executed on d will always halt after a finite amount of time and yield a valid $F(d)$.*

In our case,

A is Algorithm 5.1.

D is the set of all convex polygons.

F is the relation that returns one of the decompositions of P into triangles and line segments.

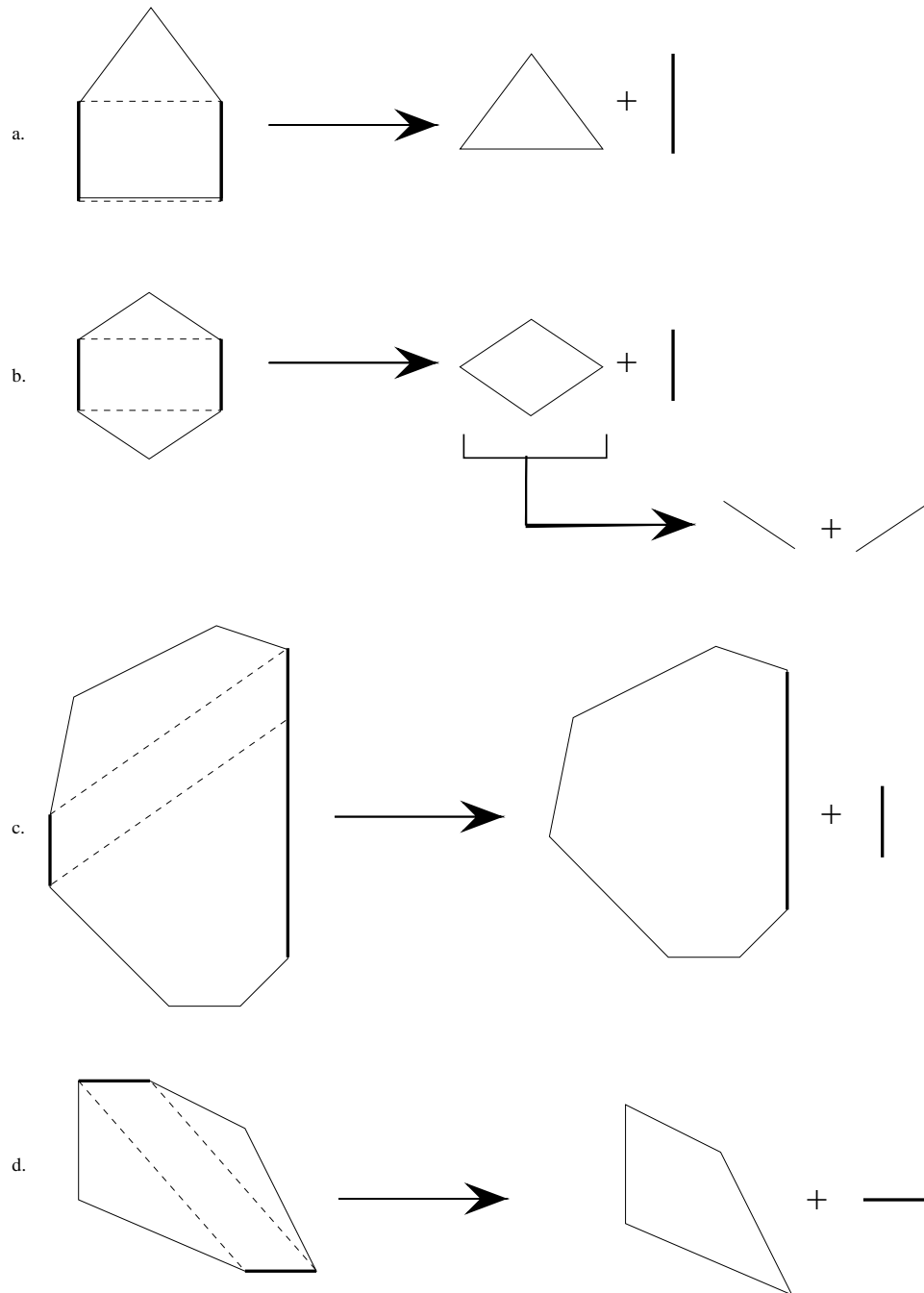


Figure 20: Preprocessing for Algorithm 5.1: removing parallel edges. Diagram (a) demonstrates that the preprocessing stage may completely decompose the polygon. Diagram (b) shows a case in which the preprocessing must be run more than once to remove all instances of parallel edges. Diagram (c) is a case where the parallel edges are not of equal length, and slicing must occur. Diagram (d) shows a case in which the parallel edges are not directly opposite to each other.

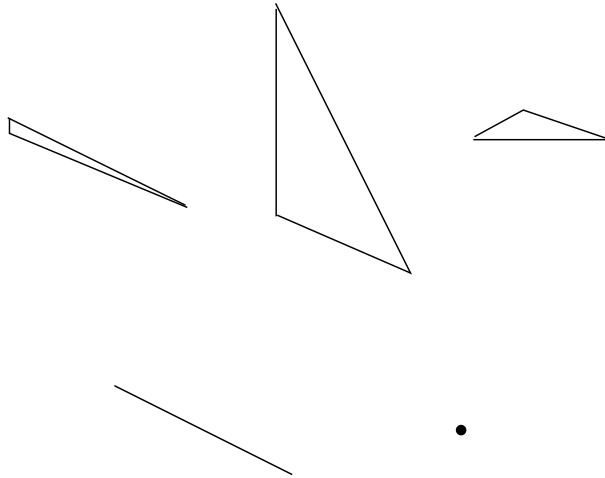


Figure 21: Step 1 of Algorithm 5.1: the base case, when the polygon is a triangle, a line segment, or a single point.

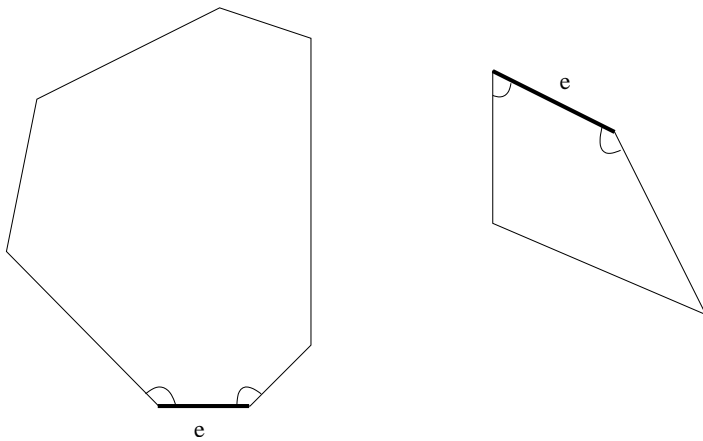


Figure 22: Step 2 of Algorithm 5.1: identifying an “expanding” edge. Note that the top and bottom edges of the polygon on the right are not quite parallel, and were thus not removed by the preprocessing stage.

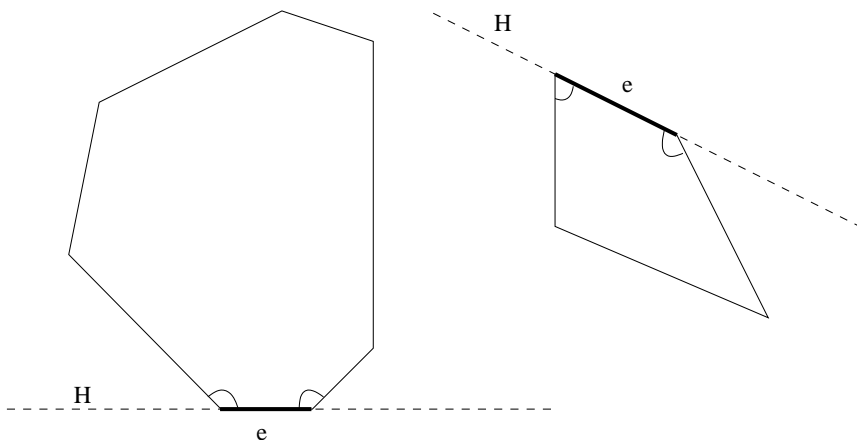


Figure 23: Step 3 of Algorithm 5.1: find the supporting hyperplane (line) for the expanding edge identified in Step 1.

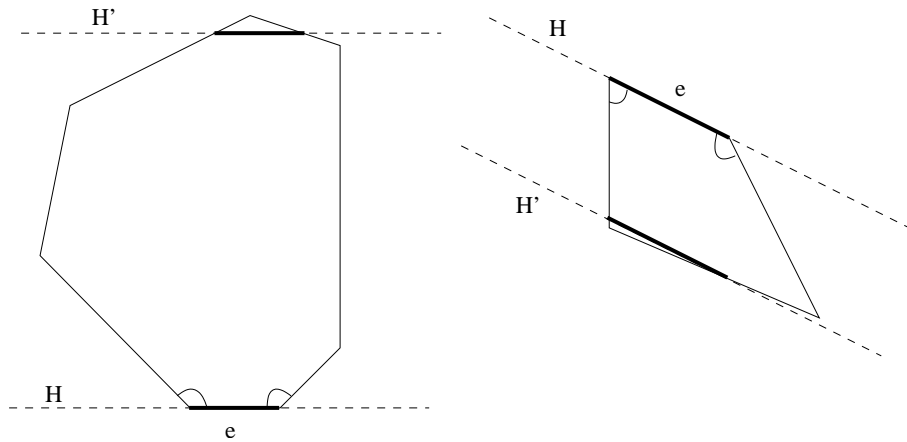


Figure 24: Step 4 of Algorithm 5.1: finding H' based on e and H .

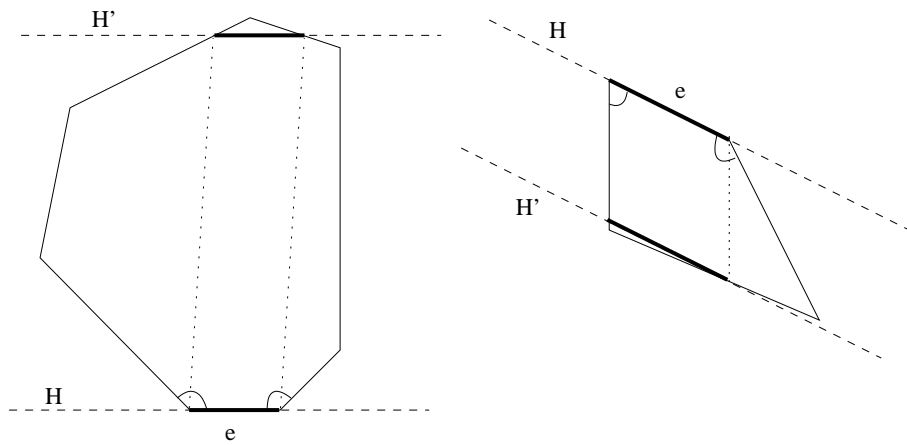


Figure 25: Step 5 of Algorithm 5.1: partitioning a sliced edge set of P based on H' .

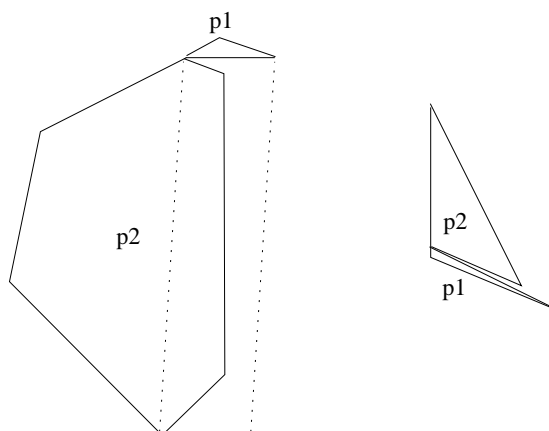


Figure 26: Step 6 of Algorithm 5.1: define p_1 and p_2 by translation based on the partition defined by Step 5.

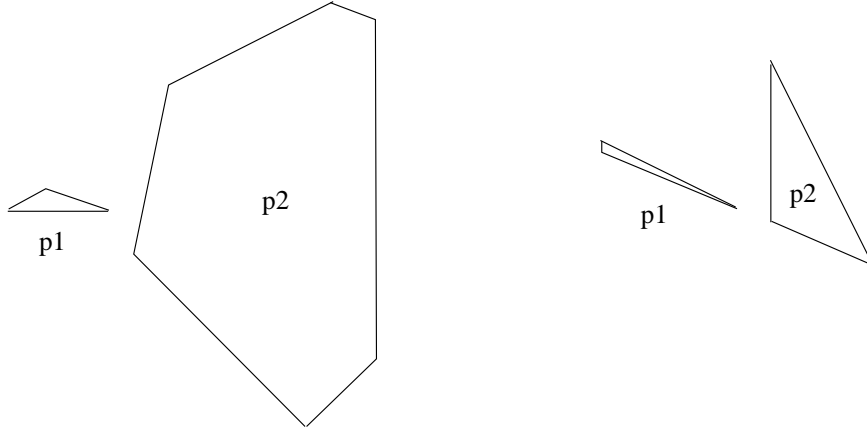


Figure 27: Step 7 of Algorithm 5.1: call the algorithm recursively on each of the summands, p_1 and p_2 .

So the algorithm is correct if it halts on any polygon, and the Minkowski sum of the resulting polygons is P . In order to prove this correctness, we will break it into its 7 steps (plus preprocessing), and treat each as a separate algorithm.

Notation: P refers to the polygon passed to Algorithm 5.1, while p refers to the polygon passed to the recursive portion of the algorithm (steps 1-7).

Preprocessing:

Claim: Let e be the line segment generated by an iteration of the preprocessing. We need to show that each iteration produces a summand of P and that we adjust P correctly. Formally, we must show that

$$P' + e = P$$

Proof:

By Theorem 4, it is sufficient to show

$$\text{edgeset}(P') \cup e = \text{some sliced edgeset}(P)$$

Once state in this form, the proof is easy. By construction, the two edges of e are parallel and of equal length to the section of the edges that were removed from P to create P' .

Claim: The preprocessing halts.

Proof:

P has some finite number of edges, n . Thus there can be at most $\frac{n}{2}$ pairs of parallel edges - a finite number. Each iteration reduces the number of pairs of parallel edges by 1. When there are no more pairs of parallel edges, the preprocessing stops.

Claim: After running the preprocessing, there are no more parallel edges in P .

Proof:

The algorithm repeats itself for each pair of parallel edges, each time remove that parallelism from the polygon. Once the preprocessing halts, it must have removed all parallel edges.

Recursive Portion of Algorithm:

Step (1)

Step 1 halts the algorithm if it is being run on a triangle, line segment, or single point. Identifying whether or not P is a triangle, line segment, or single point is trivial, so this step requires no proof. It is just the base case that makes sure we stop when we have gone far enough.

In Section 6, we will prove Theorem 6.1 which says that triangles and line segments are indecomposable. By doing so, we will have proven that Algorithm 5.1 decomposes polygons into indecomposables. However, for now we will satisfy ourselves with just proving it decomposes P into triangles and line segments.

Step (2)

Step 2 chooses an expanding edge of p . We need to prove that such a choice is always possible.

Lemma 5.1 *If p is not a triangle or a line segment, then it is always possible to choose an edge, e , of p such that the two angles adjacent to e sum to at least π radians.*

Proof:

Let P be an n -gon (a polygon with n edges). Observe that this means that there are also n interior angles (one per vertex).

Fact 1: Choose any interior point x , and connect it to each of the vertices. This construction partitions P into n triangles, each of which shares a face with p . Each triangle has π radians among its angles, so the sum of the angles of all of those triangles is $n\pi$ radians. However, the angles that occur at x form a full circle, and thus constitute 2π radians. Hence the sum of the interior angles of P is $n\pi - 2\pi = (n - 2)\pi$ radians.

Fact 2: If each adjacent pair of angles of p sum to π radians, then the sum of all such pairs is $n\pi$ radians. However, that double counts each angle, and so the actual sum of the angles of p is $\frac{n\pi}{2}$ radians.

Fact 3: Let us consider the case where we cannot find a valid edge for our algorithm. That is, all adjacent pairs of angles sum to less than π radians. We have

$$\begin{aligned} (n - 2)\pi &< \frac{n\pi}{2} \\ (n - 2) &< \frac{n}{2} \\ 2n - 4 &< n \\ n &< 4 \end{aligned}$$

So we see that the only time we can't find an appropriate edge is when we are looking at a polygon with 3 or fewer edges. Since Step 1 ensures us that p has 4 or more edges, Step 2 will always be able to find an expanding edge, e .

Step (3)

Step 3 finds a hyperplane, H , that passes through the edge provided for us by Step 2. By definition, every face of a polytope has a supporting hyperplane.

Step (4)

Step 4 finds the translation of H (called H') where $H' \cap P$ has length equal to the length of e . We need to prove that such a place exists, and that only one such place exists. When the problem is properly phrased, both of these proof follow from the intermediate value theorem.

Note that the correctness of preprocessing guarantees us that parallel edges have been removed. Thus we are looking at the generic case in which p has no parallel edges.

Consider a function $F : \mathbb{R} \rightarrow \mathbb{R}$ such that

$$F(x) = |p \cap (H_x)|$$

where H_x denotes the hyperplane which is identical to H except that x has been added to the constant term.

We will say that F is undefined over the empty set. Intuitively, increasing x moves a parallel copy of (H) along p . As it does so, $F(x)$ measures the width of that cross section of p .

Lemma 5.2 *F is continuous where defined.*

Proof:

This proof becomes clear if we look at F as a piecewise function, for which the function changes at each value of x where H_x intersects a vertex of p . In between those values of x , F is just the distance function between two lines. In order to show that a piecewise function is continuous, we need to show that each of the functions is continuous, and that they are equal where both are defined.

For any value of x for which H_x does not intersect any of the vertices of P , we know that F is continuous. That is simply the distance between two linear functions, and thus the functions in the piecewise F are continuous.

Now consider the values of x for which H_x *does* cross a vertex of P . In this case, the distance between them, $F(x)$, can be calculated using either of the two edges intersection to form that vertex. Both calculations will yield the same result since the vertex is contained in both edges. Hence the functions in the piecewise F are equal where they overlap.

Remark: Observe that $F(0) = |e|$. Since e was chosen by Step 4, it must be an “expanding edge”; its adjacent angles must sum to at least π radians. Since parallel edges have been removed, we also know that they cannot sum to exactly π radians. So, as we initially increase x , $F(x)$ strictly increases. For some $\epsilon > 0$, we have $F(\epsilon) > F(0)$.

Having no parallel edges also means that there is no parallel face opposite to e in p . That is, as we increase x , $F(x)$ will eventually be measuring the width of the opposite vertex and thus return 0. This will occur for some $x = y \in \mathbb{R}$, where $y > \epsilon > 0$.

Since F is continuous where defined, we can apply the intermediate value theorem to it over that domain. For any $z' \in \mathbb{R}$ between $F(y)$ and $F(\epsilon)$, there is some $z \in \mathbb{R}$ between y and ϵ such that $F(z) = z'$. [MH]

By letting $z' = |e|$, we are guaranteed to be able to find a z such that $F(z) = |e|$ for $z > 0$. Since $z > \epsilon > 0$, we are assured that we are finding a line segment of equal length to e , which which is not the same as e . So by letting $H' = H + z$, we get $|H' \cap p| = |e|$, which is just what we wanted.

Step (5)

There is nothing to prove for Step 5, since all we do there is defined a partition.

Step (6)

Step 6 constructs two polygons, p_1 and p_2 and claims that $p_1 + p_2 = p$.

By the construction of H' , both $s_2 \cup \{e\}$ and $s_1 \setminus \{e\}$ sum to the zero vector. Lemma 2.8 ensures us that sets of vectors which sum to the zero vector define unique polygons (which we are calling p_1 and p_2). Since the union of the edge sets of p_1 and p_2 is (by construction) a sliced edge set of p , we know that $p_1 + p_2 = p$.

Step (7)

Step 7 recursively calls the recursive portion of the algorithm on p_1 and p_2 . To justify this recursion, we will need to show that it implies that

- (1) We do not need to run the preprocessing on p_1 and p_2 again.
- (2) The algorithm halts after a finite number of steps.
- (3) When the algorithm does halt, it produces a decomposition of p into triangles and line segments.

Proof (1):

The first call to Steps 1-7 comes immediately after preprocessing, so there are no parallel edges in the edge set of p . Steps 1-7 do not add anything to the edge set of p , and the edge sets of p_1 and p_2 are subsets of the edge sets of p (by construction). Thus no new parallel edges have developed, so the subsequent calls to Steps 1-7 will pass polygons without parallel edges. Inductively, there will never be parallel edges in any call to Steps 1-7, and thus preprocessing won't be necessary again.

Proof (2):

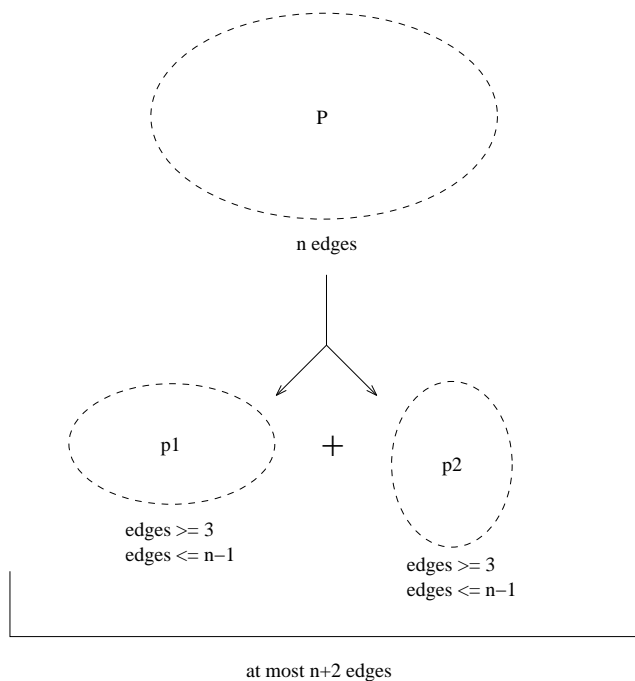


Figure 28: The recursion in Step 7 will make progress towards the base case (Step 1) and thus the algorithm will eventually halt.

Say p begins with n edges. After running Steps 1-7 on p , we may have up to $n + 2$ edges, since up to two edges could have been broken apart by the partition of Step 5. However, since P has no parallel edges, we know that the summands we produced must have at least 3 sides each. Thus each of them has at most $n + 2 - 3 = n - 1$ edges. The number of edges of the polygon decreased in size on each recursive call, so the polygon must eventually become small enough to be caught by Step 1 and halt that recursive call. Since chain of calls halts, the entire procedure halts. Figure 28 shows a visual representation of this argument.

Proof (3):

By construction, the union of the edge sets of p_1 and p_2 is a sliced edge set of p . By Theorem 4.2, we know that $p_1 + p_2 = p$.

All of the sub-proofs check out, so the algorithm will halt and yield a valid decomposition of p into triangles and line segments. Since the preprocessing is also correct, Algorithm 5.1 as a whole is correct.

In the following section we prove that triangles and line segments are indecomposable, thus showing that Algorithm 5.1 actually produced a decomposition of P into indecomposable polygons.

Remark: Step 2 involves a choice - the correctness of the algorithm does not depend upon which expanding edge is chosen. Sometimes making a different choice will produce a different (but also correct) decomposition of P . Other times, the final result of the algorithm will be the same. However, running the algorithm several time and making all possible choices for Step 2 does *not* always find *all* decompositions of P . Enthusiastic readers should examine the case of a hexagon in general position.

6 Indecomposability

In this section we examine indecomposability of polygons more closely. These results were given, without proof, by Gale in 1954 in [Gal54] and later formalized by Meyer in [Mey74]. Here we give a more different proof, which makes use of Algorithm 4.1 defined in Section 4.

Theorem 6.1 *Triangles, line segments, and points are indecomposable. No other polygons are indecomposable.*

Proof:

By using Algorithm 5.1 we can decompose any non-triangle, non-line segment, non-point polygon into triangle and line segments. Thus to prove that triangle and line segments are indecomposable, it is sufficient to show that they cannot be decomposed into triangles and line segments.

Triangles

Suppose a triangle T were decomposable into triangles and line segments. If t is a triangular summand of T , then Theorem 4.2 tells us that the edge set of t is a subset of some sliced edge set of T . Since both T and t are triangles, they each have only three edges. Thus we know that each of the edges of t is parallel to a different edge of T . This means that t and T are just scaled multiples of each other, and hence t will not be part of a non-trivial decomposition.

$$\exists k \in \mathbb{R} \mid T = kt$$

In summary, any triangles that T decomposes are just scalar multiples of T and thus do not count towards finding a non-trivial decomposition. So if a triangle is going to decompose, it will have to be into just line segments. However, that would imply that triangles are zonotopes, which they are not [She74, Ewa96]. Hence triangles are indecomposable.

Line Segments

Now lets consider decompositions of an arbitrary line segment L . Line segments cannot decompose into triangles, because triangles have three different edge-vectors (more than the line segment!). If L decomposes into line segments, then they must be parallel to L by Theorem 4.2. However, all parallel line segments are scalar multiples of each other, such a decomposition would be trivial. Hence line segments are indecomposable.

Points

A single point polytope is trivially indecomposable. It could only ever be a sum of polytopes with empty edge sets. Thus points only decompose into sums of points, which is a trivial decomposition.

7 Symmetries

Here we examine some nice properties of the effect of Minkowski summation on symmetries of the summands. These results will prove useful in Section 8 and are interesting in their own right.

Lemma 7.1 *Given a polytope $P \subset \mathbb{R}^n$ with a Minkowski decomposition*

$$P = \sum_{i=1}^k P_i$$

and any linear transformation $\gamma : \mathbb{R}^n \rightarrow \mathbb{R}^n$, then

$$\tau P = \sum_{i=1}^k \tau(P_i)$$

Proof:

Since Minkowski summation is associative, it will suffice to show the lemma for a 2 part decomposition, $P = Q + R$.

$$P = Q + R = \{q + r \mid q \in Q, r \in R\}$$

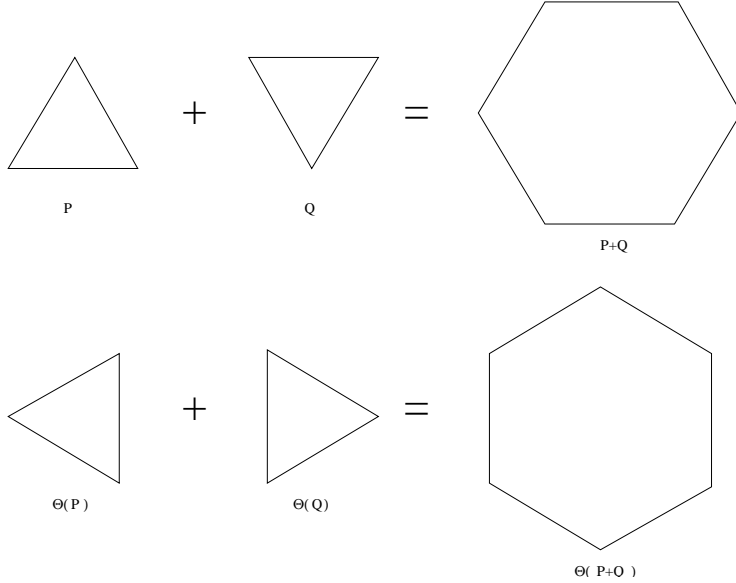


Figure 29: Rotating each of a set of polytopes clockwise by $\theta = \frac{\pi}{6}$ rotates their Minkowski sum by the same amount.

hence

$$\begin{aligned}
 \tau P &= \tau(Q + R) \\
 &= \tau(\{q + r \mid q \in Q, r \in R\}) \\
 &= \{\tau(q + r) \mid q \in Q, r \in R\} \\
 &= \{\tau(q) + \tau(r) \mid q \in Q, r \in R\} \\
 &= \tau(Q) + \tau(R)
 \end{aligned}$$

Example 7.1 A good way to visualize Lemma 7.1 is by looking at the linear transformation of rotation. That is,

$$\theta(P + Q) = \theta(P) + \theta(Q)$$

where θ is a clockwise rotation by $\frac{\pi}{6}$ radians. Figure 29 shows the case where $P + Q$ is the regular hexagon, with P and Q being equilateral triangles.

Remark: For any polygon P , the edge set of $\tau(P)$ is the same as the edge set of P , except that each of the edge-vectors has been transformed by τ .

$$\text{edgeset}(\tau(P)) = \tau(\text{edgeset}(P))$$

Theorem 7.1 Given any polytope $P \subset \mathbb{R}^n$ and two of its Minkowski decompositions

$$\begin{aligned}
 P &= \sum_{i=1}^k P_i \\
 P &= \sum_{i=1}^k P'_i
 \end{aligned}$$

and any linear transformation τ from \mathbb{R}^n to \mathbb{R}^n , if

$$\{P_1, \dots, P_k\} = \{\tau(P'_1), \dots, \tau(P'_k)\}$$

then

$$P = \tau(P)$$

Proof:

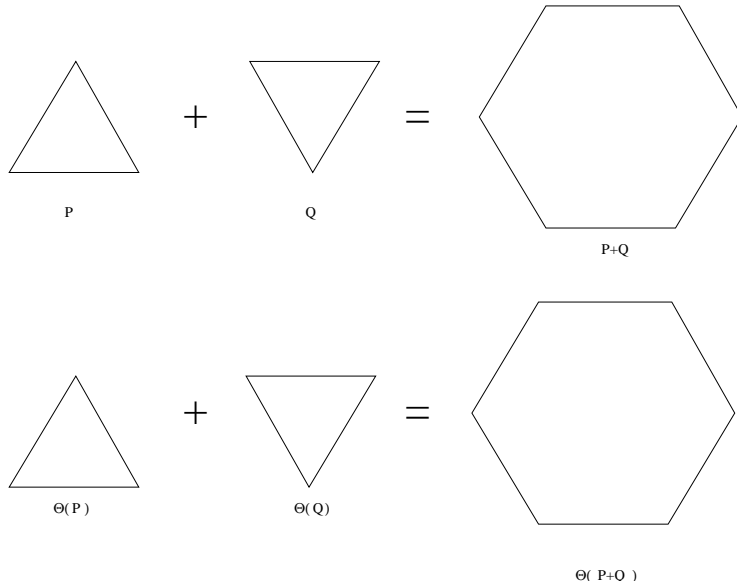


Figure 30: Applying the clockwise rotation $\theta = \frac{2\pi}{3}$ to the summands of a regular hexagon fixes each of them. Hence their sum is (trivially) fixed under the same rotation.

We are given that

$$\begin{aligned} P &= \sum_{i=1}^k P_i \\ &= \sum_{i=1}^k \tau(P'_i) \end{aligned}$$

and Lemma 7.1 tells us that

$$\sum_{i=1}^k \tau(P'_i) = \tau(P)$$

Thus $P = \tau(P)$.

Remark: There are two special case applications of Theorem 7.1.

- (1) A polytope remains fixed under any linear transformation that fixes all of the summands in any of its decompositions. See Example 7.2.
- (2) A polytope remains fixed under any linear transformation that fixes any of its decompositions (although it need not fix each of the summands in that decomposition). See Example 7.3.

Example 7.2 We revisit our earlier examination of the regular hexagon decomposed into two equilateral triangles. This time, consider the transformation of rotation by $\theta = \frac{2\pi}{3}$. We can see in Figure 30 that each of the summands is fixed under rotation by $\frac{2\pi}{3}$, and thus so is their sum.

Example 7.3 Now consider the same situation, but choose θ to be a rotation by $\frac{\pi}{3}$. Figure 31 shows this versions of the example. Not only do we see Lemma 7.1 at work, but we also observe that the hexagon $P + Q = \theta(P + Q)$. Even though neither summand has 6-way radial symmetry, their sum does.

Corollary 7.1 Given a polytope P with a decomposition $D = \{d_1, \dots, d_k\}$, Theorem 7.1 tells us that λP has a decomposition $\lambda D = \{\lambda d_1, \dots, \lambda d_k\}$ for $\lambda \in \mathbb{R}_{\geq 0}$.

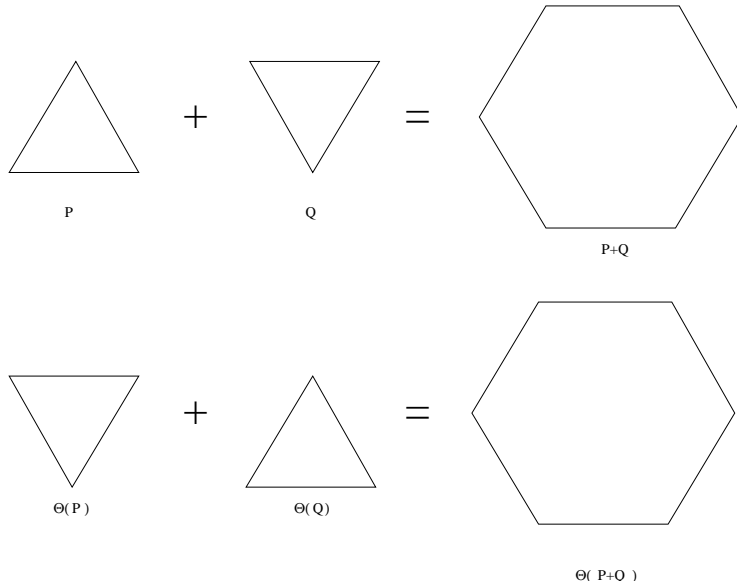


Figure 31: Here we choose $\theta = \frac{\pi}{3}$ to the summands of a regular hexagon results in the same set of summands, up to reordering. The sum inherits a symmetry that none of the summands have, but which the set of summands had as a whole.

8 Finiteness and Decomposition

There are an infinite number of decompositions for any given polytope, if we allow trivial ones such as $P = \frac{1}{2}P + \frac{1}{2}P$. As we will see, many polytopes also have an infinite number of non-trivial decompositions into indecomposable summands. However, as we've seen in numerous examples so far, polygons appear to have only a finite number of “interesting” decompositions. In this section, we identify and formalize the finite aspect of Minkowski sum decomposition of polygons.

8.1 Infinite Aspects

First we will examine the infinite aspects of decomposing polytopes.

Lemma 8.1 *For any polygon P , convex combinations of decompositions of P are themselves decompositions of P .*

Proof:

Let P be a polytope with decompositions into indecomposable summand, $S_Q = Q_1, \dots, Q_{k_1}$ and $S_R = R_1, \dots, R_{k_2}$ s. We need to show that

$$S = \sum_{i=1}^{k_1} \lambda Q_i + \sum_{i=1}^{k_2} (1 - \lambda) R_i$$

is a decomposition of P , for $0 \leq \lambda \leq 1$.

Observe that Corollary 7.1 tells us the following two important facts:

- (a) λS_Q is a decomposition of λP , and
- (b) $(1 - \lambda) S_R$ is a decomposition of $(1 - \lambda) P$.

By Lemma 2.7, $P = (1 - \lambda)P + \lambda P$. Thus $S = (\lambda S_Q \cup (1 - \lambda) S_R)$ is a decomposition of P .

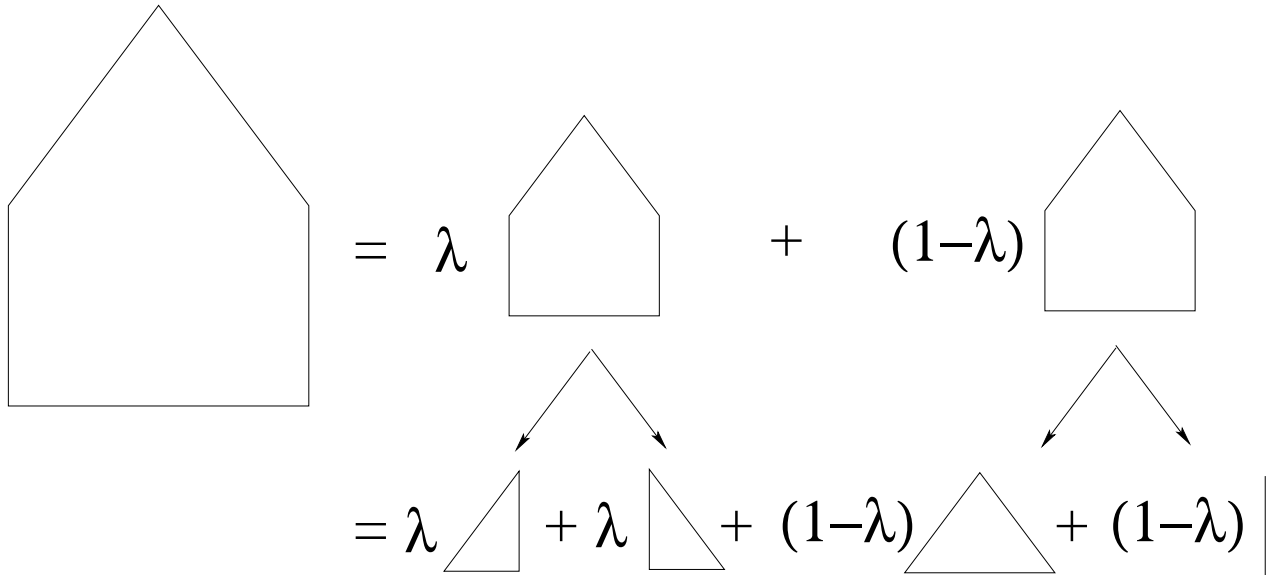


Figure 32: There are an infinite number of ways to decompose the house polygon into indecomposables.

Notation: The set of all decompositions of a polytope P denoted D_P .

Remark: The immediate implication of Lemma 8.1 is that D_P is a convex set. Of course this observation begs the question of where this convex hull resides. We will answer that question in Section 8.3. For now, we need some additional background.

Lemma 8.2 *If a polytope has two distinct non-trivial decompositions into indecomposables, then it has an infinite number of distinct non-trivial decompositions into indecomposables.*

Recall that we consider any decomposition of the form $P = \lambda P + (1 - \lambda)P$ for $0 \leq \lambda \leq 1$ to be trivial.

Proof:

Let P be a polytope with at least two distinct decompositions into indecomposable summand, $S_1 = Q_1 + \dots + Q_{k_1}$ and $S_2 = R_1 + \dots + R_{k_2}$ s. Consider the trivial decomposition

$$P = \lambda P + (1 - \lambda)P$$

for $0 < \lambda < 1$. Now decompose the two summands in two different ways. We get

$$\begin{aligned} P &= \lambda \sum S_1 + (1 - \lambda) \sum S_2 \\ &= \lambda(Q_1 + \dots + Q_{k_1}) + (1 - \lambda)(R_1 + \dots + R_{k_2}) \\ &= \lambda Q_1 + \dots + \lambda Q_{k_1} + (1 - \lambda)R_1 + \dots + (1 - \lambda)R_{k_2} \end{aligned}$$

which is a decomposition of P into indecomposable polytopes. For different choices of λ , we will get different decompositions of P . In this manner, we can produce an uncountably infinite number of decompositions of P into indecomposables.

Example 8.1 *Consider the, by now familiar, example of the house polygon. Figure 32 shows how we can construct an infinite number of decompositions of that polygon into indecomposables.*

8.2 Some Notation for Decompositions and Summands

In [Mey74], Meyer proves that the number of distinct indecomposable summands of any polytope is finite (up to translation and scaling). However, in our setting we do not want to ignore scaling. That is, if P_0

is an indecomposable summand of a polygon P , then so is $\frac{1}{2}P_0$. In order to maintain a finite number of indecomposable summands, we will only count each set of scaled summands once. Specifically, we choose the largest scaling λP_0 such that no edge of λP_0 is longer than the corresponding edge of P . Without loss of generality (and for the sake of clarity), we will refer to the largest scaling of P_0 simply as P_0 .

Notation:

Let P be a polytope with distinct indecomposable summands P_1, \dots, P_k . Every decomposition of P is of the form

$$P = a_1 P_1 + \dots + a_k P_k$$

for $a_i \in \mathbb{R}_{\geq 0}$. Of course, often most of the a_i will be zeroes. We will use *weight-vector* notation, in which we represent such a decomposition as the vector (a_1, \dots, a_k) . Any summand of P (not just indecomposable ones) can also be expressed as a sum

$$P = a_1 P_1 + \dots + a_k P_k$$

for $a_i \in \mathbb{R}_{\geq 0}$. Thus, summands of P can also be represented in *weight-vector* notation.

Remark: Not every weight vector corresponds to a decomposition or a summands, but every decomposition and summand can be expressed as a weight vectors. Furthermore, observe that every weight-vector corresponds to at most one decompositions and to at most one summand. However, a given summand may have many different weight-vectors which define it. Specifically, every decomposition of that summand has a (different) weight-vector representation which defines the summand. Once we have develop more terminology, we will better be able to understand exactly which points correspond to each polytope.

Example 8.2 Let P be the unit square. Figure 34 shows the weight vectors for some of the summands and decompositions of P .

Example 8.3 Let Q be the house polygon. Figure 35 shows the weight vectors for some of the summands and decompositions of Q .

We will build on each of these examples and add to them throughout this portion of the paper.

Remark: A natural question now arises: for a given polytope P , which k -vectors are decompositions of P ? We will answer this question for polygons, but first we will need one more piece of notation.

Notation:

For any polygon P , the *summand-edge weight matrix* of P is an $k \times n$ matrix denoted M_P ; k is the number of distinct indecomposable summands of P , and n is the number of edges of P . The entry (i, j) in M_P is the ratio of the length of the j^{th} edge of P to the corresponding edge of the i^{th} summand. Hence all the entries of M_P range from 0 to 1 (recall the discussion of the finiteness of distinct indecomposable summands given at the beginning of this section). Each row of M_P gives us the weights on the edges of one of the indecomposable summands of P .

Lemma 8.3 A weight-vector $W = (W_1, \dots, W_m)$ corresponds to a decomposition of a polygon P if and only if

$$WM_P = \vec{1}$$

where $\vec{1}$ is the length n vector of all ones.

That is, the sums of the weights contributed by the summands to each edge in P must be exactly 1.

Proof: (\Rightarrow)

Let P be a polygon with indecomposable summands $\{P_1, \dots, P_k\}$. Let $W = \{w_1, \dots, w_k\}$ be a vector of weights for the summands of P that defines a decomposition of P . Let e_j be an arbitrary edge of P with length $|e_j|$ (for $1 \leq j \leq k$).

Recall that the entries of W represent ratios of lengths of edges of P_i to the corresponding edges of P . Consider the length of e_j in the edge set of each summands of P ; these values are recorded in the j^{th} column

of WM_P). By Theorem 4.2, we know that the union of the edge sets of the summands is some sliced edge set of P . In order for that to hold, the total length of all copies of e_j in the edge sets of the summands must equal the length of e_j in P . That is, the sum of the ratios of each of those lengths to $|e_j|$ must equal one. Since this must hold for any e_j , we must have that $WM_P = \vec{1}$.

Proof: (\Leftarrow)

Let P be a polygon with indecomposable summands $\{P_1, \dots, P_k\}$. Let $W = \{w_1, \dots, w_k\}$ be a vector of weights for the summands of P that does not form a decomposition of P .

By Theorem 4.2, the edge sets of the summands must not match any sliced edge set of P . Since the proposed decomposition is a set of weights applied to the actual summands of P , the edges of P are all present in the summands. If the edge sets don't match up, then the difference must be in their lengths; the lengths of some edge at it appears in the weighted summands must not equal $|e_j|$. Let e_j be such an edge. This means that the sum on the entries in the j^{th} column must not equal one. Hence $WM_P \neq \vec{1}$.

Example 8.4 Let P be the unit square with edges $\{e_1, e_2, e_3, e_4\}$. Figure 34 shows the edge-summand weight matrix, M_P of P .

Example 8.5 Let P' be any rectangle. Observe that, since $M_{P'}$ records ratios of the summands' edges to the polygon's edges, it is the same matrix as M_P from Example 8.4.

Example 8.6 Let Q be the house polygon with edges $\{e_1, e_2, e_3, e_4, e_5\}$. Figure 35 shows the weight vectors for some of the summands and decompositions of Q .

8.3 The Decomposition Polytope

We will now examine more closely the set of all decompositions of a polygon.

Theorem 8.1 For any polygon $P \subset \mathbb{R}^2$, the set of all decompositions of P , denoted D_P , forms a convex polytope in \mathbb{R}^k .

Proof:

By Lemma 8.3, the set of all decompositions of an n -gon is completely defined by the linear equalities given by $\vec{x}M_P = \vec{1}$ with the restriction $\vec{x} \geq \vec{0}$.

The intersection of the hyperplanes given by $\vec{x}M_P = \vec{1}$ defines a hyperplane H of some (lower) dimension. By construction, the entries in M_P are all non-negative. Combined with the restriction that $\vec{x} \geq \vec{0}$, we know that the solutions form a bounded subset of H . Specifically, each entry a_i of \vec{x} is bounded below by 0 and above by $\frac{1}{a_i}$.

There is an important theorem in combinatorics which states that a region is bounded and defined by the intersection of a finite number of half spaces if and only if that region is the convex hull of a finite number of points. These are two equivalent definitions of a polytope, however proving their equivalence is surprisingly difficult and long. We omit the proof, but encourage interested readers to examine the proof given in [Zie95].

Both $\vec{x}M_P = \vec{1}$ and $\vec{x} \geq \vec{0}$ can be expressed as linear inequalities, and thus the set of solutions satisfying both conditions is an intersection of a finite number of half spaces. Thus D_P is a polytope. Specifically, this polytope resides in the positive orthant of \mathbb{R}^k , where k is the number of indecomposable summands of P .

Remark: Since D_P is a polytope, it has a finite number of extreme points. These extreme points are the finite number of "interesting" decompositions of P . It is an open question as to whether or not D_P is a polytope when P is a polytope of dimension greater than two.

Example 8.7 Let P be the unit square. Figure 34 shows the decomposition polytope for P . By looking at the rank of M_P , we can observe that D_P must be zero dimensional.

Example 8.8 Let Q be the house polygon. Figure 35 shows the decomposition polytope for Q . By looking at the rank of M_Q , we can observe that D_Q must be one dimensional. We can finally confirm that the two decompositions of the house polygon that have shown up throughout this paper are in fact the only "interesting" ones: they are the only extreme points of D_Q .

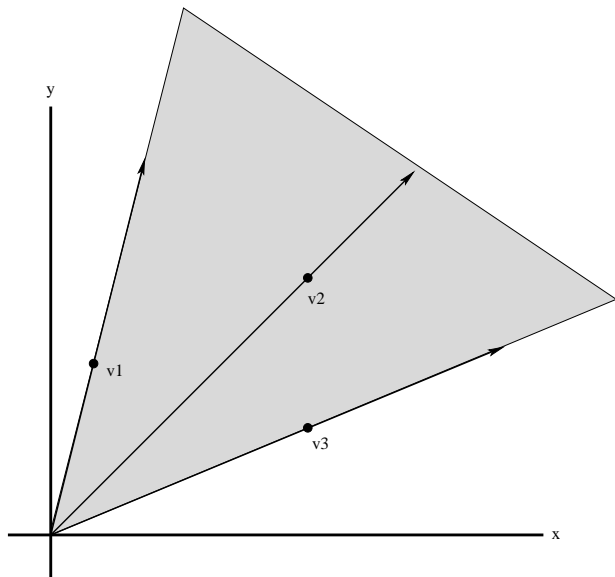


Figure 33: The conical hull of $\{v_1, v_2, v_3\}$.

Lemma 8.4 For any polygon $P \subset \mathbb{R}^2$, The dimension of D_P is the rank of M_P .

Proof:

Let P be a polygon with n edges and k distinct indecomposable summands. Lemma 8.3 lets us completely describe D_P as the set of vectors W which satisfy $WM_P = \vec{1}$. The dimension of D_P is the dimension of the row space of M_P , which is the rank of M_P . [AR94]

Definition 8.1 For any polytope P , $D_{\lambda P}$ is the union of all decompositions of all positive scalings of P .

8.4 The Set of All Summands

Here we discuss the relation between D_P and some similar results by Meyer in [Mey74]. First we will need some additional background terminology. Then we will describe some important implications of Meyer's work, and prove some lemmas about how it fits into our own results. Lastly we will summarize the relations between several important objects associated with the decompositions of a polytope.

Definition 8.2 Let $V = \{V_1, \dots, V_k\}$ be a set of vectors in \mathbb{R}^n . The conical hull of V is given by

$$\text{cone}(V) = \{x \mid x = a_1V_1 + \dots + a_kV_k, \forall a_i \geq 0\}$$

We call such x the conical combinations of V .

[Zie95]

Definition 8.3 A cone is the conical hull of a finite number of vectors in \mathbb{R}^n . Notice that a cone always includes the origin.

Example 8.9 Figure 33 shows the conical hull of three vectors in \mathbb{R}^2 .

Definition 8.4 A face of a cone C is the intersection of C with a supporting hyperplane of C . One dimensional faces of a cone are called extreme rays.

A more detailed treatment of cones is given in [Zie95].

Definition 8.5 The positive orthant of \mathbb{R}^n is the conical hull of $\{e_1, \dots, e_n\}$, where e_i is the zero vector with the i^{th} entry replaced by a 1.

Definition 8.6 For any polytope P , S_P is the set of all summands of P .

Definition 8.7 For any polytope P , $S_{\lambda P}$ is the union of the sets of all summands of all positive scalings of P .

Lemma 8.5 If P be a polytope with k indecomposable summands, then $S_{\lambda P}$ is identical to the positive orthant of \mathbb{R}^k .

Proof:

From the definitions, we see that $S_{\lambda P}$ is an conical combination of the indecomposable summands of P . However, we need to make sure that there are only a finite number of such summands. In [Mey74], Meyer examines $S_{\lambda P}$ and shows that every polytope has a finite number of indecomposable summands. Thus $S_{\lambda P}$ is equivalent to the positive orthant of \mathbb{R}^n .

We will now move on to relating the set of all summands to the set of all decompositions.

Notation:

Given two weight-vectors

$$\begin{aligned} V &= \{v_1, \dots, v_k\} \\ U &= \{u_1, \dots, u_k\} \end{aligned}$$

we say that $V \leq U$ if and only if $v_i \leq u_i$ for $1 \leq i \leq k$

Remark: This definition of “ \leq ” imposes a partial ordering on vectors in \mathbb{R}^k .

We will now introduce some new terminology which will allow us to describe some important relations among decompositions and summands of a polygon.

Definition 8.8 The umbra of a vector V in summand space $S_{\lambda P}$ of P is given by

$$\text{umbra}(V) = \{U \in S_P \mid U \leq V\}$$

The umbra of a set of vectors is the union of the umbras of those vectors.

By this definition, every vector is in its own umbra.

Remark: Any umbra in \mathbb{R}^n is an unbounded region. In fact, the umbra of a single point is a cone originating from that point. However, any umbra in the summand space of a polytope is bounded, since we are restricted to the positive orthant.

Lemma 8.6 Every summand of P is in the umbra of some decomposition of P .

Proof:

We will prove this lemma by contradiction. Let \vec{d} be a decomposition of P , and let \vec{s} be a summand of P . Suppose it were not the case that $\vec{s} \leq \vec{d}$. Then for some i , we have $s_i > d_i$.

By Lemma 8.3, \vec{d} satisfies $\vec{d}M_P = \vec{1}$ and thus the i^{th} column of $\vec{d}M_P$ sums to 1. It must therefore be that $\vec{s}M_P \not\leq \vec{1}$, due to it's i^{th} column.

By the definition of \vec{s} being summand, there must be some summand \vec{s}' such that, $(\vec{s} + \vec{s}')M_P = \vec{1}$. Since all weight-vectors are positive by construction, the i^{th} element of $\vec{s} + \vec{s}'$ must be strictly greater than 1. Hence $(\vec{s} + \vec{s}')M_P = \vec{1}$ cannot be satisfied, and we have a contradiction.

Lemma 8.7 No decomposition of P is in the umbra of another (distinct) decomposition of P .

Proof:

Suppose there were two decompositions of P , \vec{d}_1 and \vec{d}_2 , such that $\vec{d}_1 < \vec{d}_2$. By Lemma 8.3, \vec{d}_2 satisfies $\vec{d}_2 M_P = \vec{1}$. However, it must be the case that

$$\vec{d}_1 M_P < \vec{d}_2 M_P$$

which means that it cannot hold that $\vec{d}_1 M_P = \vec{1}$. This result contradicts Lemma 8.3, and so the supposition must have been false.

Lemma 8.8 *For any polygon P , D_P is a face of S_P .*

Proof:

Observe that the weight-vector for any decomposition of P is the weight-vector P treated as a trivial summand of itself. Thus $D_P \subset S_P$.

Recall that D_P for a polygon is a bounded subset of some hyperplane H . Recall from Lemma 8.6 that S_P is in the umbra of some element of D_P . Thus, $D_P = S_P \cap H \neq \emptyset$ and $S_P \subset H^-$, meaning D_P satisfies the definition of a face of S_P .

8.5 Summary and Connections

For any polytope P with k indecomposable summands, we are concerned with the following four objects in \mathbb{R}^k . Here we give a summary of their properties and some observations about the relations between them.

- (1) $S_{\lambda P}$: The set of all summands of λP , for $\lambda \in \mathbb{R}_{\geq 0}$.

$S_{\lambda P}$ is equivalent to the positive orthant on \mathbb{R}^k . The finite number of extreme rays that Meyer observed are the positive axes of \mathbb{R}^k , and correspond to the positive scalings of the indecomposable summands of P .

- (2) D_P : The set of all decompositions of P .

D_P is a convex subset of $S_{\lambda P}$. If $P \subset \mathbb{R}^2$, then D_P is a polytope with dimension equal to the rank of M_P . Typically, $\dim(D_P) < k$.

- (3) S_P : The set of all summands of P .

$S_P \subset \text{umbra}(D_P)$ and forms a bounded subset of $S_{\lambda P}$. D_P is a face of S_P .

- (4) $D_{\lambda P}$: The set of all decompositions of λP , for $\lambda \in \mathbb{R}^{\geq 0}$.

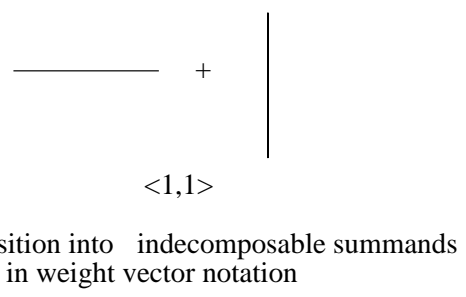
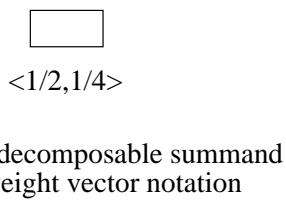
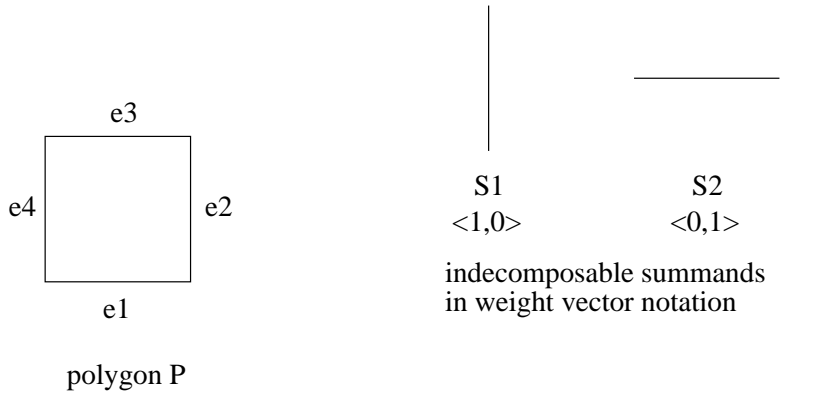
Decompositions of λP for $\lambda \neq 1$ correspond to conical scalings of D_P . These different scalings are parallel to each other, and their union forms a cone in \mathbb{R}^k .

9 Conclusion and Future Work

In this paper, we have provided a thorough background for understanding and working with decompositions of convex polytopes into Minkowski summands. We provide a number of theorems, algorithms, and terminology which serve as powerful tools for examining decompositions, especially decompositions of polygons. The accompanying *Mathematica* notebook provides an easy means to gain intuition for the subject. A description of that notebook is given in Section 11.

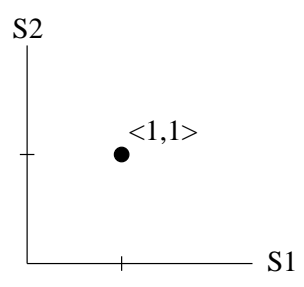
In the future, we hope to develop a general method for decomposing n -dimension polytopes and producing their decomposition polytope. We hope to generalize many of the theorems of this paper to work in higher dimensions. We would like to extend the *Mathematica* notebook to perform more of the algorithms described in the paper, and provide an even easier interface for it. We would also like to refine the **VMS** algorithm so that it is compatible with the rest of the package.

The remainder of this paper is a collection of supplements. It includes a description of time complexity, an analysis of the accompanying *Mathematica* program, as well as some results about lattice volumes.



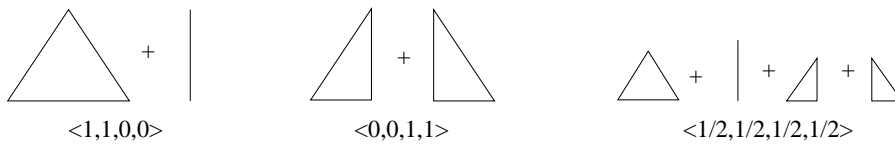
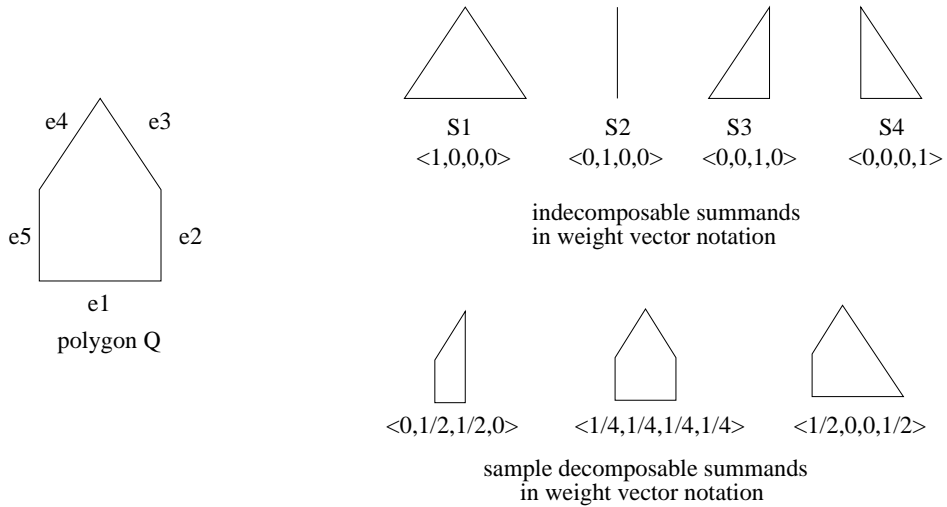
	e_1	e_2	e_3	e_4
S_1	0	1	0	1
S_2	1	0	1	0

M_P : the edge-summand weight matrix of P



D_P : the decomposition polytope of P

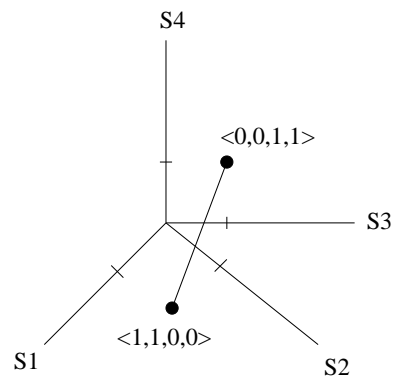
Figure 34: The breakdown of the unit square.



decompositions into indecomposable summands in weight vector notation

	e1	e2	e3	e4	e5
S1	1	0	1	1	0
S2	0	1	0	0	1
S3	1/2	1	0	1	0
S4	1/2	0	1	0	1

M_Q : the edge-summand weight matrix of Q



D_Q : the decomposition polytope of Q

Figure 35: The breakdown of the house polygon.

10 An Introduction to Time Complexity

Time complexity is a notion that allows us to classify how quickly different functions grow as their input size gets arbitrarily large. A detailed examination of complexity analysis is a very large topic, so we will only give a brief introduction. A more detailed discussion of time complexity can be found in almost any introductory computer science textbook. We particularly recommend the classic algorithms book [CLR00], as it is both thorough and readable.

Definition 10.1 A function $f(x)$ is said to be $O(g(x))$ if and only if

$$\begin{aligned} \forall N \in \mathbb{N}, \exists c \in \mathbb{N} \text{ s.t.} \\ \forall n \geq N, cf(n) \leq g(n) \end{aligned}$$

[CLR00] That is, g is eventually larger than f after some scaling.

Remark: Since the constant c can be set arbitrarily low, a multiplying a function by a constant does not change its complexity class.

There are 6 major complexity classes, which are listed here in order from fastest to slowest.

- Constant Time, $O(1)$
- Logarithmic Time, $O(\log n)$
- Linear Time, $O(n)$
- N-LOG-N Time, $O(n \log n)$
- Polynomial Time, $O(n^k)$ for some $k \in \mathbb{R}$
- Exponential Time, $O(2^n)$

In general, exponential time algorithms are unacceptable, as a small increase in input size can double the amount of work done. No matter how much processing power one has, an exponential algorithm can be made inhibitive slow with only a small increase in the problem size. Polynomial time algorithms are much better than exponential, but for large problems are still often inhibitive.

Remark: Under normal circumstances, sorting cannot be done faster than $O(n \log n)$. As a result, many algorithms are prevented from being faster. No matter how efficient an algorithm is, if it has to sort its input before running, then it cannot improve past $O(n \log n)$.

Equipped with this terminology, we are now ready to examine and analyze the Mathematica notebook that accompanies this paper.

11 The *Mathematica* Notebook: Usage Guide and Analysis

In this section, we will give an introduction to the *Mathematica* tool that accompanies this document. *Mathematica* is a powerful mathematical analysis program produced by Wolfram. All code included in that notebook should be considered open source.

We provide a set of functions that allow for a convenient, visual manipulation of Minkowski sums, as well as implementations for some of the algorithms discussed in the paper. First we will give a summary of the functions and their syntax, as well as giving a few examples. In addition, we will give a brief description of internal workings of the functions and analyze their time complexity.

11.1 Functions and Syntax

We have written four *Mathematica* files dealing with Minkowski sums, plus an additional one written by Fukuda and Mizukoshi [FM] that we make use of internally. These files define three basic sets of commands.

- The file *conversion.nb* contains commands of the form $\{\mathbf{UC}, \mathbf{OC}, \mathbf{UE}, \mathbf{OE}, \mathbf{UV}, \mathbf{OV}\}$ to $\{\mathbf{OC}, \mathbf{UE}, \mathbf{OV}\}$, which convert between different representations of polygons.
- The file *mink_sum.nb* contains the function **RMS**, which recursively finds the Minkowski sum of a list of polygons in the **OC** (ordered corner) representation.
- The file *mink_sum_vertices.nb* contains the function **VMS**, which performs the Minkowski sum of two polytopes in **UC** representation, but does so by adding together their vertices. As we will see in Section 11.5, this algorithm has a worse time complexity than **RMS**, but can handle polytopes of arbitrary dimension. **VMS** is included as a demonstration of an alternative, but is not itself very refined yet. It does not currently clean its vertices, and thus produced output which cannot be inputted into **DePo**. However, its simplicity allows us to be sure of its correctness in all cases, and we hope to improve its output in future versions.
- The file *decom_poly.nb* contains the function **DePo**, which takes a polytope in **OC** format and produces the extreme points of its decomposition polytope.

In general, lattice polygons and rational polygons work the best, since they do not incur problems with roundoff errors and precision mismatching.

Syntax and examples are described thoroughly in the notebooks themselves. We do the same below, as well as giving some proofs of accuracy and complexity.

11.2 Type Conversion Functions

There are six types of representations of polygons which the user may wish to convert among.

UC - UnorderedCorners

OC - Ordered Corners

UE - Unordered Edges

OE - Ordered Edges

UV - Unordered Vectors

OV - Ordered Vectors

Unordered means that they are in no particular order and no particular initial element.

Ordered means that the list is in counter-clockwise order. The first element is the southwest most corner - a notion which is formally defined at the beginning of the proof of Theorem 12.1. Intuitively, we want to start the list with southern most corner, breaking ties by taking the western most corner.

Corner representations define the polygon as a list of its vertices. We use the term “corner” rather than “Vertex” so that the abbreviation is unambiguous with the Vector representation (described below). Recall that vertices and corners are equivalent notions [Zie95].

Edge representations define the polygon as a list of its edges. Each edge is represented as an ordered pair of vectors; the first of which is the initial point, and the second of which is the ending point. Within each edge, those two points must be ordered so as to reflect the counterclockwise orientation of P (even when the edges are unordered).

Vector representations define the polygon as a list of its edge vectors. These vectors have all been translated to the origin, so they are stored simply as ordered pairs (unlike the Edge representations). Each vector is directed so as to reflect the counterclockwise orientation of P , even when the vectors are unordered. Notice that, unlike the other two representations, vector representations do not record translation. That is, converting to **OV** representation and then back again *loses* information.

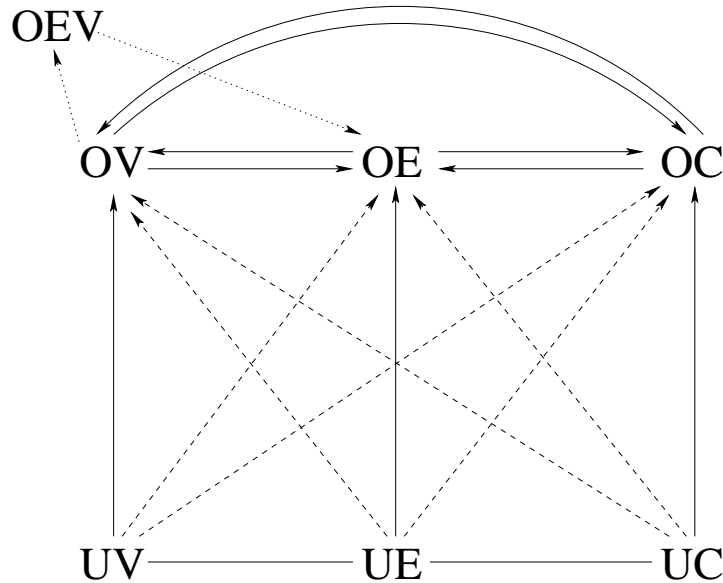


Figure 36: A diagram showing the conversions between different polygon representations that are provided in the accompanying *Mathematica* notebook. The dotted lines indicate conversions that were written simply by composing other conversion functions. **OEV** is an internal intermediate representation that is included for the sake of readers who want to understand the actual workings of the code.

Example 11.1 *A simple unit square can be represented in any of these ways.*

OC: $\{\{0, 0\},$
 $\{1, 0\},$
 $\{1, 1\},$
 $\{0, 1\}\}$

OE: $\{\{\{0, 0\}, \{1, 0\}\},$
 $\{\{1, 0\}, \{1, 1\}\},$
 $\{\{1, 1\}, \{0, 1\}\},$
 $\{\{0, 1\}, \{0, 0\}\}\}$

OV: $\{\{1, 0\},$
 $\{0, 1\},$
 $\{-1, 0\},$
 $\{0, -1\}\}$

The actual functions are all of the form “ $\{\{UC, OC, UE, OE, UV, OV\}$ to $\{OE, UC, OC\}$ ”. All possible conversions are shown in Figure 36.

For example, **UEtoOC** takes a polygon in Unordered Edge representation and returns the same polygon in Ordered Corner representation.

11.3 Internal Workings of RMS (Recursive Minkowski Sum)

We will now take a high level look at the internal workings of the functions described in the previous section. Further descriptions can be found in the program file itself. **RMS[P]** expects a Polygon in **OC** (Ordered Corner) representation.

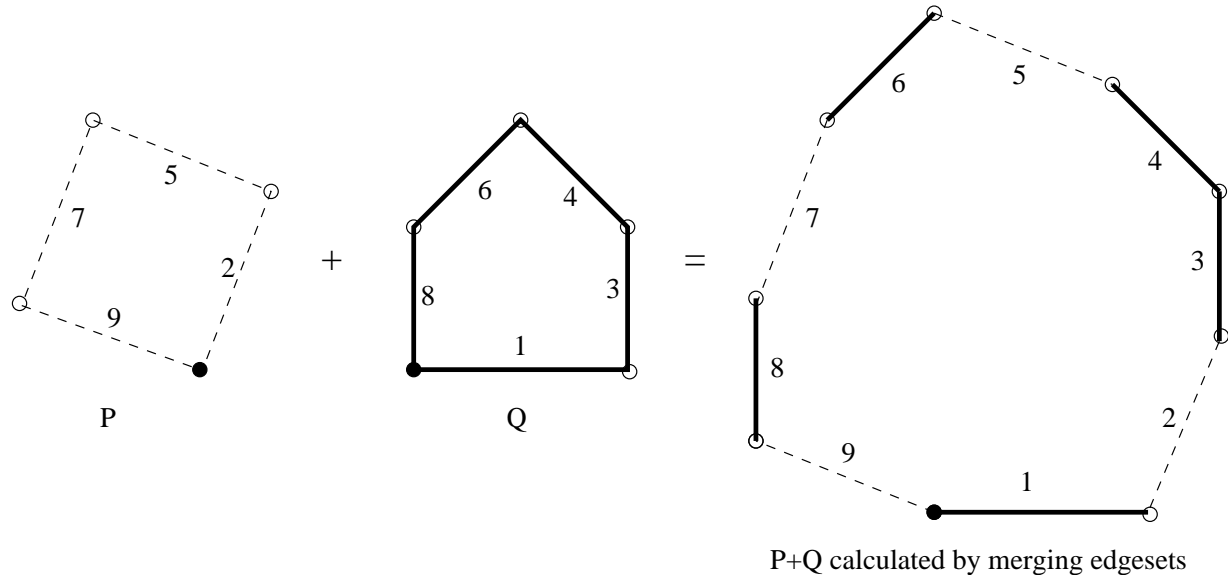


Figure 37: An illustration of how the Minkowski sum of two polygons can be found by performing merge sort on the edges of the summands, and sorting by angle.

On a technical note, the first thing the function does is take a copy of the first corner of the list and add it to the end of the list, using the **MakeCyclic** function. Thus while RMS takes standard **OC** representation as its parameter, the analysis below will assume that the representation has already been augmented by **Makecyclic**.

The **RMS** function calculates the Minkowski sum of a list of polygons recursively, by looking at them in pairs. It sums the last two in the list, then takes that result and adds it to the third to last element of the list, and so on. In order to do the actual sums, **RMS** call on the **ChooseMS** function. The function **ChooseMS** determines if either of the two polygons are single points. If so, it calls **PointMS** to perform the calculation. If not, it calls **PolyMS** to perform the sum.

The **PointMS** function does the obvious; it take the non-point polygon and add the point polygon to all of the vertices. The **PolyMS** function is more subtle, and is based on Section 4. Theorem 4.2 tells us that the Minkowski sum of two polygons will have the same edge set as the union of the edge sets of the summands, up to slicing. Thus, we know what the edges will look like (by comparing adjacent vertices in the summands), so we only have to put them in the correct order. By Lemma 2.8, we know that they define a unique polygon (up to translation). We can put the vectors in the correct order by comparing their angles up from standard position.

If you are given a set of edges (pairs of vertices), you can figure out how to make a convex polygon out of them by just looking at the angles they make (how much they are rotated up from standard position). The angles must be increasing (non-strictly) or else the shape will be concave. However, since we know that the two summands already satisfy this property (that their vertices are sorted by angle) we can just “merge” their edges together. We do this in a way analogous to merge sort; “walk” around both summands simultaneously and and compare the current edges. Take the one with a lower angle (up from standard position) and then advance your position on that summand by one edge. In the case of a tie, either choice is correct. The result will be a sorted list of all of the edges in both summands, and thus be the Minkowski sum of those two polygons (by Theorem 4.2). This merging procedure is exactly what **PolyMS** does, and is illustrated in Figure 37.

Notice that in the case of a tie during merging, there will end up being a stray vertex in the middle of an edge of the sum. This phenomenon comes from the fact that the edge sets of the summands are equal to the *sliced* edge set of the sum, and occurs when the summands have some edges that are parallel. The stray vertex corresponds to a place where that edge would have to be sliced in order to get the edge sets of the two summands. However, it is technically an error and clutters up the representation of that polygons.

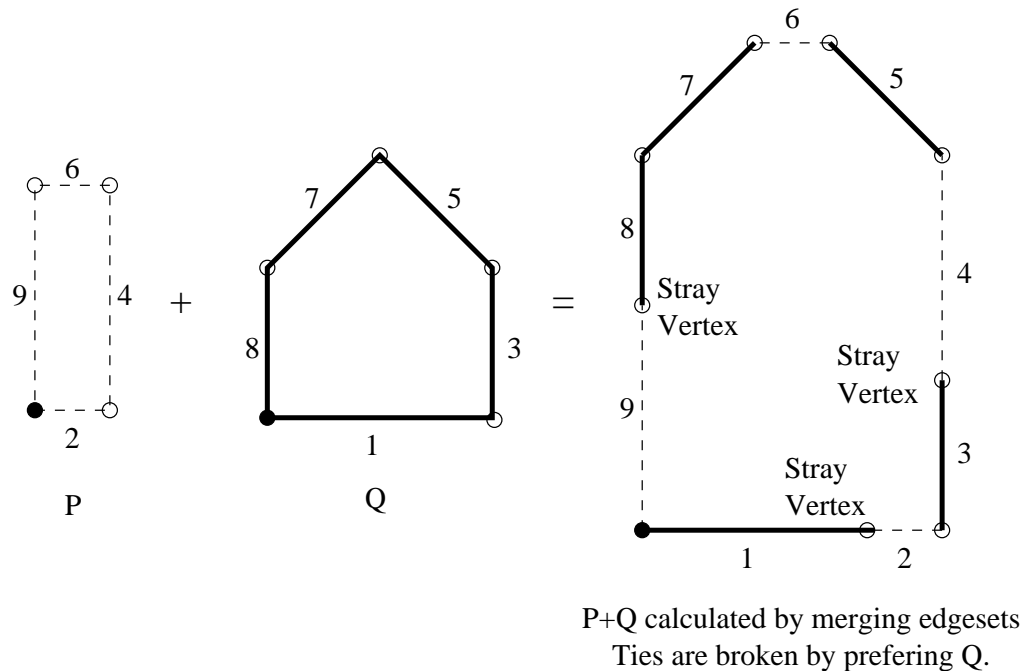


Figure 38: Another example of Minkowski sums calculated using merge sort. In this case, some stray vertices are produced along the edges of the result.

The **CleanEdges** function detects and removed such vertices from a give polygon. An example of this phenomenon is shown in Figure 38.

11.4 Complexity Analysis of RMS

Merging two sorted lists is a linear in terms of the total length of both lists. In this case, this means that the *RMS* function is $O(n)$ where n is the total number of edges among all the summands.

This complexity assumes that we are using a sorted-vertex representation of polygons. If we are given polygons in another form, then there will be some additional preprocessing. Specifically, we may need to sort the edges, which would raise the complexity to $O(n \log n)$. This increase is acceptable, since the algorithm is still faster than polynomial time.

Remark: This analysis assumes that simple operations provided by *Mathematica* are constant time. That is, we assume that addition, multiplication, and other simple operations require constant or essentially constant time to perform. This simplification is a common one and is a fairly safe assumption to make.

11.5 Alternate Methods for Computing Minkowski Sums

In order to give the reader an idea of how the efficiency of the the *RMS* algorithm compares to other algorithms for computing Minkowski sums, we will briefly present and analyze some alternative algorithms.

One salient way to write an algorithm is to have it follow directly from the mathematical definition. As any programmer will tell you, this approach often fails and even more often is inefficient. In our case, consider trying to directly compute the Minkowski sum:

$$P + Q = \{p + q \mid p \in P, q \in Q\}$$

For non-trivial cases, there are an uncountably infinite number of valid p and q vectors to consider. Hence the naive algorithm will never even terminate.

However, there is a mildly clever modification that can be made to the naive algorithm to make it terminate: just consider the vertices. One might observe (correctly) the following:

Lemma 11.1

$$\text{conv}(V_P) + \text{conv}(V_Q) = \text{conv}(V_P + V_Q)$$

where V_P and V_Q are vertices of polygons.

That is to say, we only need to add each of the *vertices* of P to each of the *vertices* of Q , and then take the convex hull of the result. This follows from Lemma 2.2 and from the fact that the vertices of $\text{conv}(V)$ are always elements of V [Gru67]. This approach only involves a finite number of sums, and thus will terminate nicely.

However, consider the complexity of the vertex-sum approach. Let n be the number of vertices of P and m be the number of vertices of Q . Without loss of generality, let $n \geq m$. The procedure will add each of n vectors to each of m other vectors, thus taking a total of nm operations. In general, this method will take us $O(n^2)$ time, where n is the number of edges of the summands (which is the same as the number of vertices of the summands). While a polynomial algorithm is reasonable efficient, the *RMS* function beats it, and will be considerably faster at computing sums of large polygons.

On the positive size, the vertex-sum approach can sum polytopes of any dimension, and are not restricted to polygons. This is a classic tradeoff between efficiency and generality.

We provide **VMS**, an implementation of this approach, in the accompanying *Mathematica* notebook. It has not been refined enough to clean stray vertices from its interior, and thus is not compatible with the other functions in that notebook. We hope to improve it in future versions of the notebook.

11.6 The DePo Function (Decomposition Polytope)

Recall from Section 8 that the set of all decompositions of a polygon forms a polytope in high decomposition space. The **DePo** function generates finds the extreme points if D_P for a given polygon P . To just find the edge-summand matrix, use the **EdgeSummand** function.

The **DePo** function operates through brute force. It first identifies all size 2 and size 3 subsets of P , and then determines which of the size 3 subsets are able to form triangles. It constructs each such triangle, measures the side lengths with the Law of Sines, and generates the edge-length ratios. Lastly, it builds the edge-summand matrix and feeds it into `Vertex_Enum`, a *Mathematica* package written by Fukuda and Mizukoshi. [FM]

12 Bounds on Lattice Volume

When we take a Minkowski sum $P + Q = R$, observe that the *lattice volume* of S will be greater than or equal to that of each summand. Here we define the lattice volume of a polytope as the number of integer lattice points (\mathbb{Z}^n) inside of that polytope. In this section, we will find (and prove) tight bounds on the lattice volume of a polytope as a function of the lattice volume of its Minkowski summands.

12.1 Background

Definition 12.1 The lattice volume of a polytope $P \subset \mathbb{R}^n$ is the size of the set

$$\{z \in \mathbb{Z}^n \mid z \in P\}$$

We will denote this $L(P)$.

Remark: We will only be dealing with the standard lattice \mathbb{Z}^n , however the theorems in this section can be generalized to arbitrary lattices.

12.2 Lattice Polygons

First we will examine lattice polygons.

Definition 12.2 A polytope $P \subset \mathbb{R}^n$ is a lattice polytope if and only if all of its vertices are elements of \mathbb{Z}^n . That is, if and only if

$$P = \text{conv}(V) \text{ with } V \subset \mathbb{Z}^n$$

Remark: Since polytopes are required to be non-empty, this definition means that every lattice polytope contains at least one integer lattice point.

Lemma 12.1 A Minkowski sum of lattice polytopes is itself a lattice polytope.

Proof:

Let P and Q be lattice polygons and $R = P + Q$. We need to show that R is a lattice polygon, which means we need to show that the vertices of R are in \mathbb{Z}^n . Recall from Theorem 3 that $\text{conv}(A + B) = \text{conv}(A) + \text{conv}(B)$; that is, the sums of the vertices of two polytopes completely determines their Minkowski sum. Since sums of lattice points are lattice points, the $R = P + Q$ must have vertices that are lattice points.

Theorem 12.1 Let P and Q be lattice polytopes in \mathbb{R}^n . The bounds

$$L(P) + L(Q) - 1 \leq L(P + Q) < \infty$$

are correct and tight (no stronger bounds exist).

Remark: For this proof, we will impose an ordering on \mathbb{Z}^n . Given

$$\begin{aligned} \vec{x} &= \{x_1, \dots, x_n\} \in \mathbb{Z}^n \\ \vec{y} &= \{y_1, \dots, y_n\} \in \mathbb{Z}^n \end{aligned}$$

we will say that $\vec{x} \leq \vec{y}$ if and only if either $\vec{x} = \vec{y}$ or $x_i \leq y_i$ where i is the first entry at which the two vectors differ. For polygons, we are working in \mathbb{Z}^2 and we say that the smaller of two vectors is the southwestern most one (where south has priority over west).

Proof:

In order to prove this theorem, we need to make sure that the upper and lower bounds are both correct and tight.

(1) The Lower Bound is Correct

We will use the stamping algorithm to prove this portion of the theorem. Without loss of generality, let P be the template and Q be the stamp. Let the handle h be at the smallest vertex of Q (using our imposed ordering).

Unlike the usual stamping algorithm, we will place the images of the stamp in a very precise manner. We will place h on each of the integer points of the template (P), starting with the largest one. Each subsequent stamped image will be made by placing h at the next smaller integer point in P . After stamping h at all of the lattice points, we can go back and stamp the rest of the points in any order.

The first image placed will have $L(Q)$ integer points in it. Each additional placement of the template will add at least one new (non-redundant) integer point to the result. We can be assured of this by our ordering. At no time will a placement of h create an image of the stamp that covers any integer points that h has not already been placed at. Thus, if nothing else, the integer point in P where h was placed is new.

Hence we get the lower bound $L(P) + L(Q) - 1 \leq L(P + Q)$.

Example 12.1 Figure 39 shows an example of two lattice polygons. Notice that we chose the handle to be the southwest most corner of the stamp (Q). The handle is placed on each of the lattice points in the template (P) before being placed anywhere else. Furthermore, we place them in the order indicated by the arrows, starting at the top-most vertex.

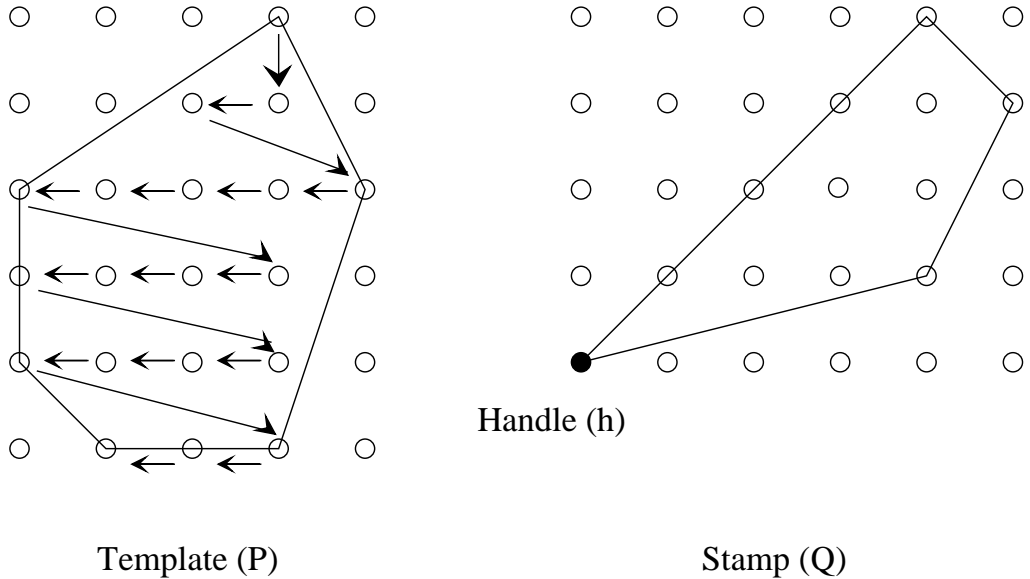


Figure 39: A two dimensional illustration of the proof of the lower bound given in of Theorem 12.1.

(2) The Lower Bound is Tight

Once we know that the lower bound is correct, we can show that it is tight by finding an example for which the bound is exact.

Example 12.2 Let $P = Q$ both be the line segment $\text{conv}(\{(0,0), (0,1)\})$. Their sum is the line segment $\text{conv}(\{(0,0), (0,2)\})$.

$$\begin{aligned} L(P) &= 2 \\ L(Q) &= 2 \\ L(P + Q) &= 3 \\ L(P) + L(Q) - 1 &= 3 \end{aligned}$$

And we see that $L(P + Q) = L(P) + L(Q) - 1$.

The lower bound given in the lemma is exactly correct in this case, meaning there can be no larger lower bound. Of course, there may be a better lower bound if we allow ourselves to have more information about the summands than just their lattice volumes. We will examine that idea later on in this section.

(3) The Upper Bound is Correct

Polytopes are bounded and integer lattice points are not dense within the reals. Thus there can only be a finite number of integer points within any given polytope. [MH]

(4) The Upper Bound is Tight

Suppose there were a better (smaller) upper bound, $L(P + Q) \leq M$. Then we could construct a counterexample in the following manner.

Let P be the line segment $\text{conv}(\{(0, M), (1, 0)\})$, and let Q be the line segment $\text{conv}(\{(0, 0), (1, 0)\})$. By making Q a summand, the resulting polygon will necessarily have a width of at least 1 unit at all heights. Since the summand P has a height of M , the resulting polygon must have a height of at least M . Thus, there must be at least M integer points contained in $P + Q$. Furthermore, at the top and bottom of $P + Q$, there will be 2 integer points rather than just 1. Thus we have

$$\begin{aligned} L(P) &= 2 \\ L(Q) &= 2 \\ L(P + Q) &= M + 2 \end{aligned}$$

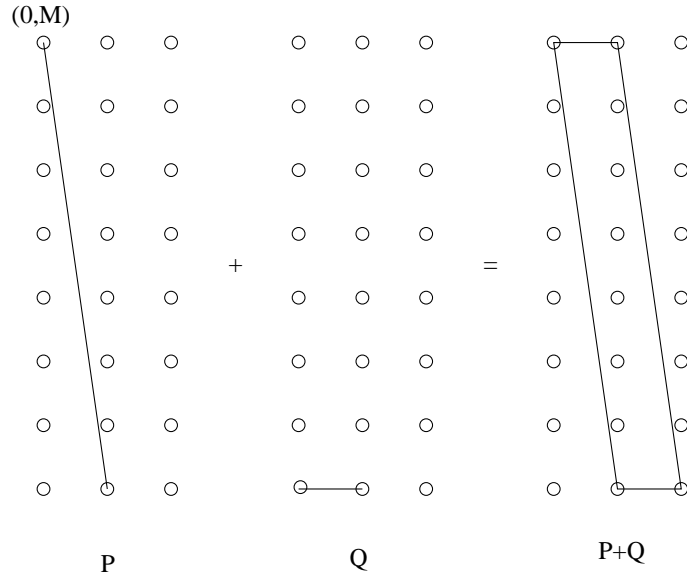


Figure 40: An illustration of the counterexample given in part 4 of Theorem 12.1, with $M = 7$.

which contradicts the attempted upper bound. This example is illustrated in Figure 40 for $M = 7$.

Knowing the integer volumes of the Minkowski summands is not enough information to put any finite upper bound on the lattice volume of the Minkowski sum.

Conjecture:

The lower bound given in Theorem 12.1

$$L(P + Q) \leq L(P) + L(Q) - 1$$

is an equality only if either (a) P and Q are parallel line segments, or (b) at least one of the summands is a single point.

12.3 General Polygons

Now we will examine arbitrary polygons and allow non-lattice summands.

Lemma 12.2 *Any lower bound on the lattice volume of a lattice polytope is also a lower bound for general polytopes, although not necessarily a tight one.*

Proof:

It is equivalent to say that the convex hull of all the lattice points of P is contained in P , which follows from the fact that the convex hull of any subset of P is contained in P . Hence any lower bound for the volume of that subset will also apply to the entire polygon.

Theorem 12.2 *Let P and Q be polytopes in \mathbb{R}^n (not necessarily lattice polytopes). The bounds*

$$L(P) + L(Q) - 1 \leq L(P + Q) < \infty$$

are correct and tight.

Proof:

Knowing Lemma 12.2, the proof exactly follows the proof of Theorem 12.1.

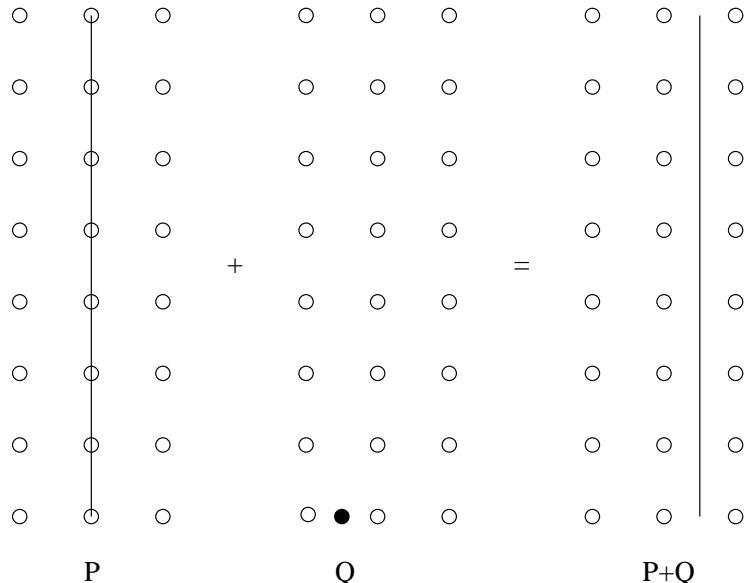


Figure 41: An illustration of the general counterexample given Theorem 12.3, with $M = 7$.

Theorem 12.3 *Let P and Q be polytopes in \mathbb{R}^n . The trivial bounds*

$$0 \leq L(P + Q) < \infty$$

are the tightest (best) bounds that can be found as a function of the value of $L(P) + L(Q)$.

That is to say, if we only know the total number of integer lattice points in the two summands, then we cannot make any non-trivial claims about the number of lattice points in the Minkowski sum. This result may be counter to intuition, but the proof is illuminating.

Proof:

The proof given of Part 4 of Theorem 12.1 also applies here, and shows that there is no better upper bound. Thus our task is to show that there is no better lower bound. We can see this by constructing a general counterexample which takes summands with arbitrarily high total lattice volume and produce a sum with zero lattice volume.

Example 12.3 *Let $P = \text{conv}((0, 1), (0, M))$ for any arbitrarily large M , and let $Q = (\frac{1}{2}, 0)$. We get*

$$\begin{aligned} L(P) &= M \\ L(Q) &= 1 \\ L(P + Q) &= 0 \end{aligned}$$

which serves as a counterexample.

This example is illustrated in Figure 41.

13 Acknowledgments

This thesis is presented for completion of the Haverford College Undergraduate Mathematics Major. It was produced under the advising of Dr. Curtis Greene. Additional thanks go to Johnicholas Hines and Dr. Stephanie Frank Singer for insights and advice. The document was generated using a Linux version of L^AT_EX.

References

- [Alt97] Klaus Altmann. The versal deformation of an isolated toric gorenstein singularity. *Inventiones Mathematicae*, 128:443–479, 1997.
- [AR94] Howard Anton and Chris Rorres. *Elementary Linear Algebra*. Anton Textbooks, seventh edition, 1994.
- [Bar] Alexander Barvinok. Lattice points and lattice polytopes.
- [BGK96] Thomas Burger, Peter Gritzmann, and Victor Klee. Polytope projection and projection polytopes. *American Mathematical Monthly*, 103(9):742–755, November 1996.
- [CLR00] Thomas H. Cormen, Charles E. Lierserson, and Ronald L. Rivest. *Introduction to Algorithms*. McGraw-Hill Book Company, New York, 2000.
- [Ewa96] Gunter Ewald. *Combinatorial Convexity and Algebraic Geometry*. Graduate Texts in Mathematics. Springer-Verlag, New York, 1996.
- [FM] Komei Fukuda and Ichiro Mizukoshi. Vertex enumeration of polytopes. Web Page. Version 0.41 Beta.
- [Gal54] David Gale. Irreducible convex sets. 1954.
- [GL] Shuhong Gao and Alan Lauder. Decompositions of polytopes and polynomials.
- [Gru67] Branko Grunbaum. *Convex Polytopes*. Interscience Publishers, New York, 1967.
- [Kar01] Yarl Karshun. Private Communication, October 2001. Follow up to talk given at University of Pennsylvania Mathematics Department.
- [KS01] Efim Kinber and Carl Smith. *Theory of Computing: A Gentle Introduction*. Prentice Hall, Upper Saddle River, New Jersey, 2001.
- [McM71] P. McMullen. On zonotopes. *American Mathematical Association*, 159, September 1971.
- [Mey74] Walter Meyer. Indecomposable polytopes. *American Mathematical Society*, 190, 1974.
- [MH] Jerrold Marsden and Michael Hoffman. *Elementary Complex Analysis*.
- [PS85] Franco P. Preparato and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.
- [She74] G. C. Shephard. Combinatorial properties of associated zonotopes. *Can. J. Math.*, 26(2):302–321, 1974.
- [Str89] Strayer. *Linear Programming and Its Applications*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1989.
- [Zie95] Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer-Verlag, New York, 1995.

Force references to be listed, even if not cited:

[KS01] [FM] [Ewa96] [Alt97] [Bar] [BGK96] [PS85] [Gal54] [GL] [Gru67] [Kar01] [MH] [McM71] [Mey74]
[She74] [Str89] [Zie95] [CLR00]

Homework #Rob Seater
May 1, 2002

Problem #1:

Choose *one* of the following two statements, and prove it using the core definitions given in lecture. Both are easy.

- (a) The convex hull of a set of vectors is convex.
- (b) Minkowski summation is associative.

Problem #2:

Let

$$Q = \text{conv}(\{(0, 0), (2, 0), (1, 1)\})$$
$$R = \text{conv}(\{(0, 0), (1, 0), (2, 1), (0, 1)\})$$

- (a) Find and label the vertices of the polygon $P = Q + R$.
- (b) Verify that the edge sets of the summands match some sliced edge set of the sum.
- (c) Find a decomposition of P into indecomposables, which does not include Q .

Problem #3:

Use the decomposition algorithm described in class to find the decomposition of the following polygon:

$$W = \text{conv}(\{(0, 0), (2, 0), (2, 2), (1, 2), (0, 1)\})$$

Extra Credit: (*This problem is worth more than required ones, but still probably is not worth your time*)

Find the edge-summand weight matrix for W as defined in Problem #3.

Collaboration:

Name your collaborators. Name all collaborators who verbally berate you and all collaborators whom you verbally berated.

I work in the Math Question Center on Tuesdays. My extension is x6191 (different than stated in the student listing). I can usually be found in the Stokes basement computer lab (008). I sometimes go back to my room (13b Comfort) when my professors aren't keeping me busy enough. If you come to ask me questions, I will try to get you to play xpilot with me, or any other computer game that isn't my thesis. If you are persistent enough, I will give up and actually help you.