8-1-2024

Enumeration of Level-k Hypertriangulations

Patrick J. Kaylor
The University of Texas Rio Grande Valley

ENUMERATION OF LEVEL-K HYPERTRIANGULATIONS

A Thesis

by

PATRICK J. KAYLOR

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Mathematics

The University of Texas Rio Grande Valley

August 2024

ENUMERATION OF LEVEL-K HYPERTRIANGULATIONS

A Thesis
by
PATRICK J. KAYLOR

COMMITTEE MEMBERS

Alexey Garber, Ph.D.
Chair of Committee

Luigi Ferraro, Ph.D.
Committee Member

Alexey Glazyrin, Ph.D.
Committee Member

Oleg Musin, Ph.D.
Committee Member

August 2024

ABSTRACT

Kaylor, Patrick J., ENUMERATION OF LEVEL-k HYPERTRIANGULATIONS. Master of Science (MS), August, 2024, 30 pp., 1 table, 31 figures, 5 references.

Triangulations are a classical object in discrete and computational geometry that finds it uses in many other fields, including numerous applications. In this thesis we approach the question of enumerating all hypertriangulations of a point set, the family of tilings introduced by Olarte and Santos in 2022 as induced projections of hypersimplices. Unfortunately, the usual approach to enumeration of trinagulations using flips may not work for hypertriangulations as flip-connectivity is only conjectured for that family even in the two-dimensional case. As a result, we develop a DFS-based algorithm and present its Python implementation that enumerates all hypertriangulations of small point sets for all feasible levels k.

DEDICATION

To my wife Saamara, who has been patient through more than one should reasonable ask.

ACKNOWLEDGMENTS

I would like to thank my chair, Alexey Garber, Ph.D. for helping me through this, and never expressing frustration even when he should have.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The concept of triangulations has been an important topic in both applied and pure mathematics since Euclid, and has been more formalized since at least the 19th century. Hypertriangulations are a generalization of triangulations introduced by Olarte and Santos in [3]. For a given point set of n points, a parameter k defines level of each hypetriangulation; we give formal definitions in Chapter III. The purpose of this study is to develop a computer program to find and enumerate level-k hypertriangulations for a particular point configuration, denoted as A. This is the first study to find and enumerate k-level triangulations where $k = 3, 4$ or more. To find and enumerate these hypertriangulations, we developed a tool based on the Depth First Search (DFS) algorithm. We also present some of our results for specific point sets.

Before we address hypertriangulations, however, we shall introduce triangulations and discuss how triangulations are enumerated in the remainder of this chapter and the next. Then we review hypertriangulations. Next, we discuss our algorithm. Finally, we display some of the results from the algorithm.

Definition 1 (Triangulations). A triangulation of a point configuration A in $\mathbb{R}^d$ is a collection of d-simplices. The points of these vertices are in A. The following two properties must be satisfied:

(a) The union of all the simplices must be equal $conv(A)$, the convex hull of A.

(b) Any pair of simplices must intersect in a common face; this common face could be the empty space.

Triangulations have several useful applications. First, they are used in computer graphics, as most 3-Dimensional models are rendered as a mesh of triangles. Second, triangulations can

improve numerical stability and accuracy in numerical analysis and simulations. This is because triangulations allows us to create meshes that can capture our domain of interest. In this instance, triangles can better conform to irregular boundaries than other shapes, providing us with greater accuracy. Next, triangulations are useful in Geographic Information Systems (GIS), since they can create structures such as Triangulated Irregular Networks (TINs) that represent terrain surfaces well. This is important in spatial analyses such as contouring, slope analysis and 3-Dimensional visualization.

Further, enumerating triangulations is a classic topic in discrete geometry, numerical analysis, and computer science [5].

CHAPTER II

TRIANGULATIONS

We now more fully explain a given set's family of triangulations.

Definition 2 (Convex set). A set C is convex if any line segment joining two points in C lie entirely in C.

Definition 3 (Convex Hull). The convex hull of a set of points A is the smallest convex set containing A. We denote the convex hull by $conv(A)$

Simply put, the building blocks of every triangulation are its vertices, edges, and the triangles themselves. For the set $A = \{v_1, \ldots, v_n\}$, the points v_i are the triangles' vertices within this triangulation. Single triangles within a triangulation are spanned by three vertices, v_i, v_j, v_k. We also assume that the edges, e_{ij}, e_{ji}, are not ordered and represent the same edge between v_i and v_j.

A triangulation also has the following requirements [4]. First, no triangle in a triangulation is degenerate; for any (i, j, k) in a set of triple points $P_{\triangle_T}$, a_i, a_j, and a_k cannot be collinear. Next, the interiors of any two triangles in a triangulation cannot intersect, and their boundaries can only intersect at a common edge or vertex. Further, triangulations must be hole-free.

For example, let A be a set of 7 points where v_6 and v_7 are inside the convex hull of the other points. A valid triangulation could exclude v_6, using only v_1, v_2, v_3, v_4, v_5, and v_7, thus illustrating non-intersecting interiors and boundaries, and demonstrating compliance with the hole-free requirement (Figure 1).

As a second example, consider the family of all triangulations of a convex polygon C_n. Let C_n denote a convex polygon with n vertices. These vertices are numbered clockwise from 1 to n. A polygon's number of triangulations is not contingent on the vertices' coordinates. Any $n-3$

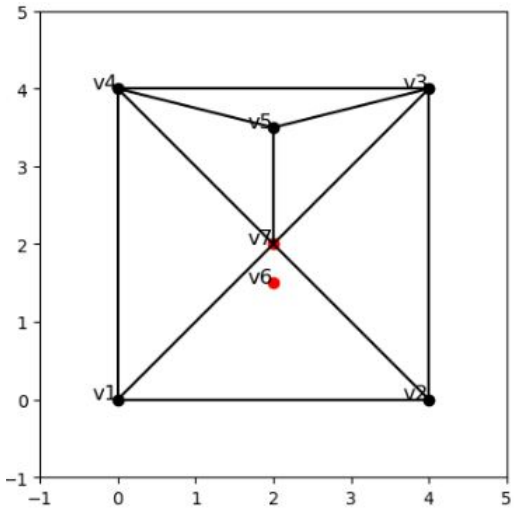


Figure 1: Triangulation of 7 Points Excluding Vertex 6

diagonals not crossing one another will produce a triangulation. Further, for $n = 2$ we use triangles, and for specific case of n points in the convex position, every triangulation can be obtained by choosing $n-3$ non-intersecting diagonals.

Beyond the trivial case, where a triangle has only one triangulation, a triangulation is never unique. This presents complications for enumeration. For example, a quadrilateral has 2 possible triangulations, a pentagon has 5, a hexagon has 14, and a heptagon has 42. This complexity affects the counting of distinct triangulations for larger polygons. The visualization in Figure 2 helps illustrate these differences, where we can see the five different possible trianglulaions in a pentagon.

This leads us to the proposition that a convex n-gon has a specific number of triangulations, denoted t_n, which is the Catalan number. Catalan numbers are a sequence of natural numbers that have applications in various combinatorial problems, one of which is counting the number of ways to divide a polygon into triangles with non-crossing diagonals[1].

The correlation between C_n and t_n substantiates the recursive strategies employed in practical computations of polygon triangulations. This leads us to the following proposition. [1].

Proposition II.1. *If we set $t_n = 2$, then the sequence of numbers $t_2, t_3, t_4, \ldots$ satisfies the following recurrence relation: $t_n = t_2 t_{n-1} + t_3 t_{n-2} + \ldots + t_{n-1} t_2$*

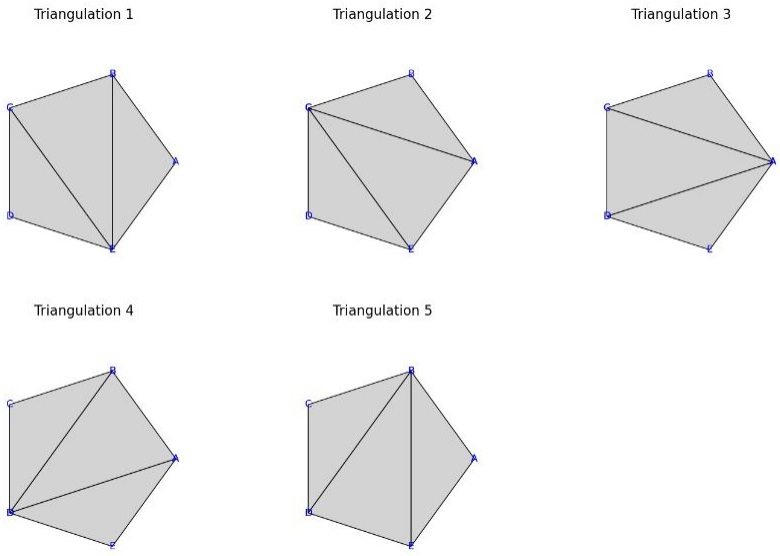


Figure 2: Pentagon Triangulations

Theorem 2.5 is used to calculate these recurrences with triangulations.

Theorem II.2. *The number t_n of triangulations of a convex n-gon equals*

$$\frac{1}{n-1}\binom{2n-4}{n-2}$$

Flips are used to examine how triangulations change, and flip-connectivity provides an efficient method to create algorithms to construct triangulations. Figure 2 provides a clear visualization of what we mean by flips. In this example, we have our original triangles, *ABC* and *BCD*, within a quadrilateral. The triangle begins at *BC*, flips to *AD*, then back to *BC*, and again to *AD*.

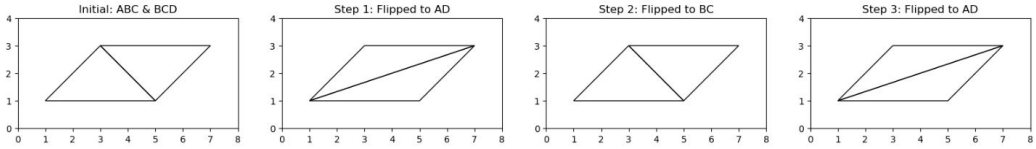


Figure 3: The First Two Quadrilateral Triangulations

For a convex polygon, the number of flips of a triangulation are related to Theorem 2.5 [1]. There are two types of flips. The first type of flip substitutes one diagonal of a convex quadrangle by the other (seen in Figure 3). The second type of flip, an example is shown in Figure 4, subdivides a

triangle into three by adding a vertex or coarsens by removing a degree-3 vertex [2].

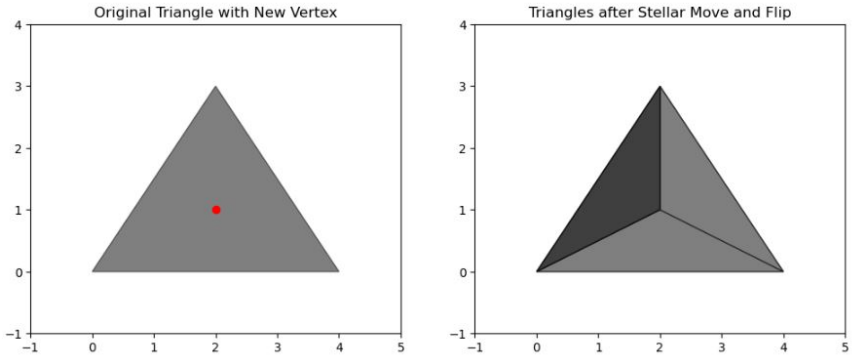


Figure 4: A Flip Subdividing a Triangle

A usual approach to the enumeration of triangulations of a given point set uses flips as a technique to obtain all such triangulations. Hypertriangulations can extend triangulations's theoretical concepts, and it is to them that we now turn. Our goal is thus to enumerate hypertriangulations and we use DFS as flips are not guaranteed to accomplish this.

CHAPTER III

HYPERTRIANGULATIONS

We now move to the main topic for our project: hypertriangulations. First, however, we need to introduce several preliminary definitions. Define the index set $[n] = \{1, 2, \ldots, n\}$. An integer k, referred to as the level, is fixed between 1 and $n-1$. Let $A = \{a_1, a_2, \ldots, a_n\}$ represent a set of n distinct points in $\mathbb{R}^2$. For any subset $I \subseteq [n]$, let $a_I = \sum_{i \in I} a_i$ denote the vector sum of the points indexed by I.

Next, we write $\Delta_n = \text{conv}\{e_1, e_2, \ldots, e_n\} \subseteq \mathbb{R}^n$ for the standard $(n-1)$-simplex where $e_1, e_2, \ldots, e_n$ are vectors of the standard basis in $\mathbb{R}^n$. Then, more generally, $\Delta_n^{(k)} = \text{conv}\{e_I \mid I \subseteq [n], |I| = k\}$ for the k-th standard $(n-1)$-dimensional hypersimplex. Fix a parameter $k \in [n-1]$, and define $A^{(k)} = \{a_I \mid I \subseteq [n], |I| = k\}$ as the set of k-fold sums.

Consider all partial triangulations of $A^{(k)}$, defined as the decompositions of the convex hull of $A^{(k)}$ into triangles, where each triangle is formed by connecting subsets of points with straight edges. Assume temporarily that no three points in $A^{(k)}$ are collinear. $A^{(k)}$ references the set of k-fold sums of the points in A.

The projection of Δ_n on A can be extended to a projection of $\Delta_n^{(k)}$ that gives rise to hypertriangulations by considering appropriate faces of the hypersimplex. We study the hypertriangulations of a finite set of n points in the plane that are induced by the projections of an $(n-1)$-dimensional hypersimplex to the plane [2]. We shall more formally define a hypertriangulation in a simpler way below using specific edges and triangles that are available.

Definition 4. A level-k hypertriangulation of A is a partial triangulation of $A^{(k)}$ such that:

(V) every vertex is of the form a_I, where $|I| = k$,

(E) every edge connects two vertices, a_I and a_J, with $|I \cap J| = k-1$.

From this definition, we can further define *white triangles* and *black triangles*, as the condition on the endpoints of every edge applies to the vertices of every triangle:

Definition 5. Let $\Delta = a_I a_J a_K$ be a triangle whose vertices and edges satisfy conditions (V) and (E). Then:

If $|I \cap J \cap K| = k-1$, Δ is called a *white triangle*,

If $|I \cap J \cap K| = k-2$, Δ is called a *black triangle*.

White triangles are possible when $1 \leq k \leq n-2$, and black triangles are possible when $2 \leq k \leq n-1$. For a triangulation T, denote $W(T)$ and $B(T)$ as the sets of white and black triangles, respectively.

As an example, for $n = 4$ points in $\mathbb{R}^2$, we analyze level-k hypertriangulations for $k = 1, 2, 3$. In the general case, there are only two combinatorially distinct configurations of four points:

- Vertices of a convex quadrangle,
- Vertices of a triangle with the fourth point positioned internally.

We label these as the convex configuration and the non-convex configuration, respectively.

We can visualize this in Figure 5, where we have the two configurations of four points and their hypertriangulations. This includes the convex configuration in the top row and the non-convex configuration in the bottom row. From left to right we have the two level-1, level-2, and level-3 hypertriangulations for each configuration. Notice that the squares in the upper-middle can be more general parallelograms so the respective central fifth vertices are not at the same geometric location. The convex hexagons in the lower middle are not necessarily regular either, but they are centrally symmetric [2].

Moreover, we can see the following with each level.

Level $k = 1$: The vertices coincide with the original points, and all triangles formed are white.

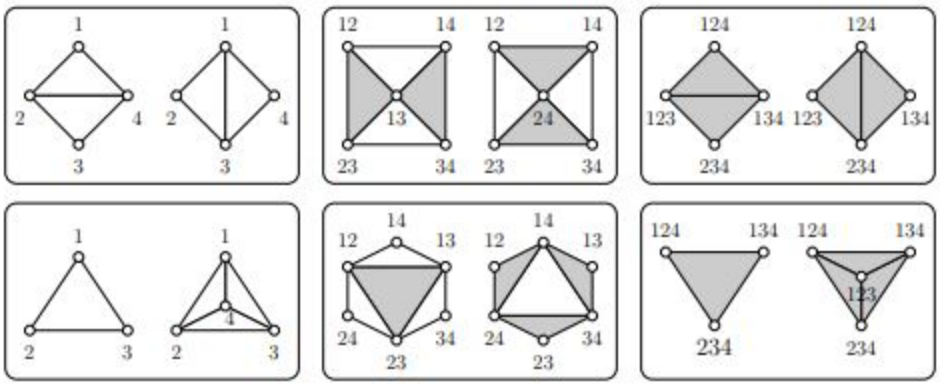


Figure 5: Two configurations of four points in $\mathbb{R}^2$ and their Hypertriangulations

Level $k = 2$: Comprising six points, each represented as the sum of two points from the set A. Under a convex configuration of A, the convex hull of $A^{(2)}$ forms a parallelogram, with the remaining two points residing inside this parallelogram. This configuration allows only two hypertriangulations, each selecting one of the two internal points as a vertex and partitioning the parallelogram into two white and two black triangles. Conversely, if A is in a non-convex configuration, the points of $A^{(2)}$ align as the vertices of a centrally symmetric convex hexagon, which again permits only two hypertriangulations.

Level $k = 3$: This is similar to the scenario at $k = 1$, where all triangles are now black.

We build upon the work of Edelsbrunner, et al. [2] and build our Depth First Search algorithm for every generic point set.

CHAPTER IV

ALGORITHM

The aim of this project is to generate a list of triangles, and their graphic forms, from a set of points for any input set and level $k \geq 1$. The expected output consists of the total number of hypertriangulations and triangles in each convex and non-convex hull. As previously discussed, flip-connectivity is unresolved in higher dimensions, so we do not use flips to find all triangulations as we do in lower dimensions.

To accomplish our task, the project used a form of a Depth First Search algorithm. Depth First Search (DFS) is an algorithm used to traverse or search through tree or graph data structures. Originating with the work of Charles Pierre Trémaux in the 19th century, DFS is particularly effective in maze-solving contexts.

Depth First Search is an algorithm that explores a graph by starting at a root node and traversing as far along each branch as possible before backtracking. This traversal method uses a stack to keep track of discovered nodes and manage backtracking. DFS requires space proportional to the number of vertices to maintain the stack of vertices.

We use the example shown in Figure 6 to illustrate how DFS operates on a tree. A depth-first search starts at node A. The algorithm assumes that the left edges in the shown tree are chosen before the right edges. The algorithm also assumes that the search remembers previously visited nodes and will not repeat them. Therefore, it will visit the nodes in the following order: A, B, D, F, E, C, G.

While DFS works with both graphs and trees, a tree is a special case of a graph without cycles, and with a single unique path from the root to any other node. This allows for a more straightforward traversal approach, as it is done in our example.

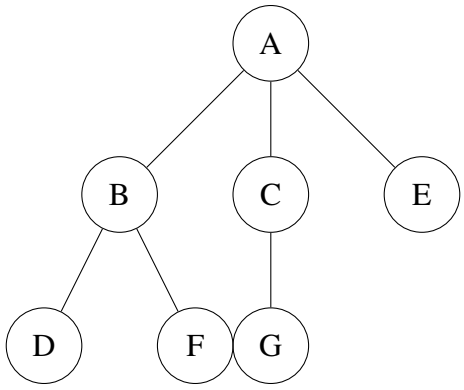


Figure 6: Tree visualization of Depth First Search

In summary, Depth First Search (DFS) is used to explore the graph maximally along each branch before backtracking. This traversal uses a stack to monitor nodes and backtracking. In our context, DFS helps enumerate all hypertriangulations efficiently.

Our algorithm's task is to create, analyze, and check the relationships between generated triangles based on a set of input points, while considering specific "hypertriangulation rules" (see Table 4.1). These rules determine acceptable geometric arrangements for the triangles relative to one another. We next review the code's key components and how the hypertriangulation rules are incorporated. For this project, the programming language python was used. Moreover, our hypertriangulation rules work as conditions for DFS, which checks which triangles can be added.

Table 4.1: Hypertriangulation Rules

Rule 1: Two Coinciding Vertices – The labels of these vertices coincide, and the third vertices are on different sides of the common line.
Rule 2: One Common Vertex – The labels are the same, and the triangles do not intersect.
Rule 3: The triangles do not intersect.

First, standard python libraries were used for the project: numpy for numerical operations, itertools for combinatorics, shapely to address geometric operations, and matplotlib for plotting. In addition, the time function is used to track performance. Next, specific functions were created. A determinant function computes the determinant to determine the orientation of three points and if they are collinear. A function calculates the area of a triangle using the shoelace theorem. The

Shoelace Theorem, also known as Gauss's area formula for polygons, calculates the area of a simple polygon whose vertices are defined in the Cartesian coordinate plane. It is useful as it easily determines the area from the coordinates of the vertices without needing to perform more complex integration or decomposition. To compute the area A of a simple polygon with vertices $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ using the Shoelace Theorem we have:

$$A = \frac{1}{2} \left| \sum_{i=1}^{n} (x_i y_{i+1} - y_i x_{i+1}) \right|$$

where $x_{n+1} = x_1$ and $y_{n+1} = y_1$ to ensure that the polygon is closed by connecting the last vertex back to the first.

After we import our libraries, we generate our triangles from our point configuration A. Triangles are generated based on subsets of points. The code creates 'white' and 'black' triangles based on different criteria of subset selection and point summation. Then a hypertriangulation matrix is created to track valid relationships between all pairs of triangles based on the aforementioned hypertriangulation rules. A *recursive* function solve then fits triangles together to cover the area of the convex hull without violating hypertriangulation rules. This function uses the hypertriangulation matrix to keep adding triangles that comply with the rules until the area of the convex hull is matched or candidates are exhausted.

Next, a *find subset* function generates all combinations of a given set of elements taken n at a time. The *convex hull* area function computes the area of the convex hull formed by n-fold sums of subsets of points.

helper functions determine if hypertriangulations exist within a set of points. A function was generated to calculate and store each triangle's area. Then the *convex hull area* function is computed, which calculates the convex hull of k-fold sums.

We next implement our *hypertriangulation rules* function; the centerpiece for implementing the specified triangulation rules. Each rule is addressed in the following way. First, we look for two coinciding vertices. When two triangles share two vertices, their third vertices must be on opposite sides of the line segment joining these shared vertices. This is checked using the determinant to

ensure they lie on opposite sides (cross-product signs will differ). Next, we check for one common vertex. If triangles share only one vertex, they must not intersect beyond this shared vertex. This is checked by ensuring the intersection of the two triangles is exactly the shared point, using geometric intersection functions from *Shapely*. Finally, we check for no common vertices. Triangles that do not share any vertices should not intersect. This rule is checked using the intersection method to confirm that there is no overlapping area between the triangles. Additionally, each vertex of $A^{(k)}$ has a label, which must be considered during these checks.

Finally, the time taken for the entire operation is measured, to assess the algorithm's efficiency. See Figure 7 for visualization of our algorithm.

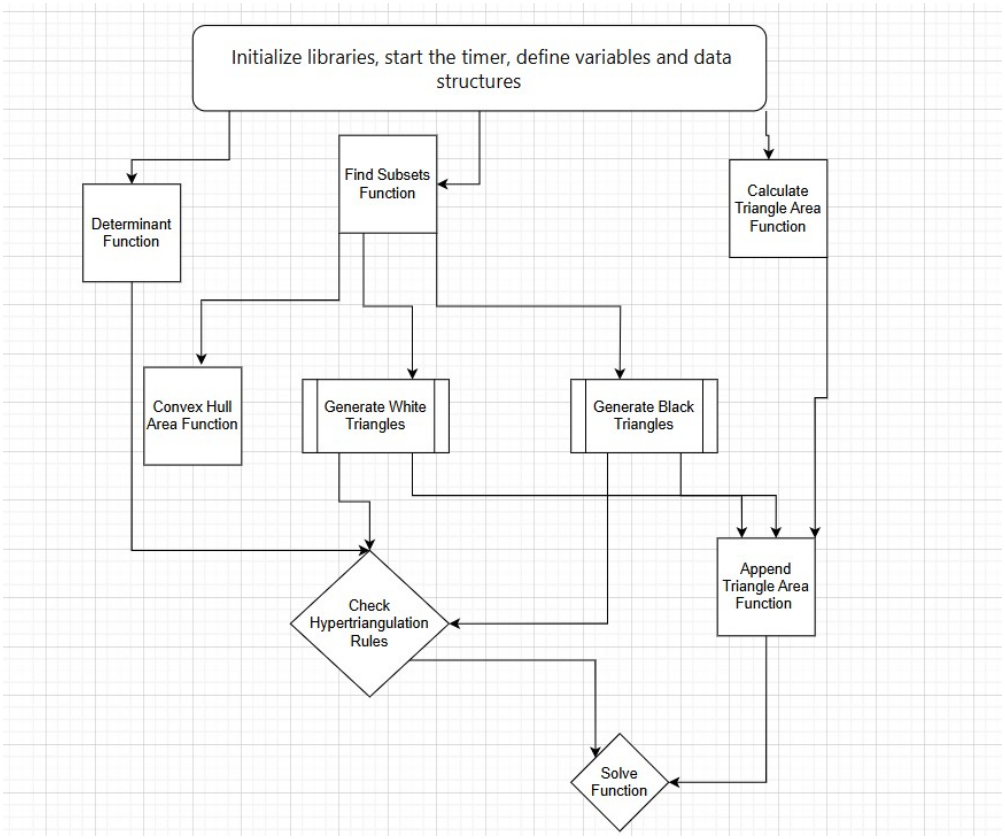


Figure 7: Algorithm Flowchart

The code aims to address problems related to the enumeration of hypertriangulations. This is relevant in theoretical studies of geometric structures and their properties.

CHAPTER V

RESULTS

The computational results verify our theoretical expectations when $k = 1$ and $k = 2$. In our first example, we look at points in convex position for $n = 5, 6, 7$ and $k = 1, 2$. For $k = 1$ we get Catalan numbers, as expected, and for $k = 2$ we verify the numbers from Olarte and Santos[3]. Both results confirm previously known data. See Figures 8, 9, 10, 11, 12, and 13 below. Note that not all sums are included in a specific hypertriangulation. For Figures 7, 11 and 13, we display the actual sum points, and not just a point. We only do this for these three Figures, as graphically it is less appealing. Moreover, in these three Figures, $P = points$ and $S = sums$.

Next, we examined cases where $n = 4$ and $k = 1, 2, 3$. This time, however, we look at points that are in a convex position and points that are not in a convex position. For the convex position, see Figures 14, 15 and 16. For those points that are not in a convex position, see Figures 17, 18 and 19. These hypertriangulations are from Edelsbrunner, et al [2], and are therefore known, demonstrating that the algorithm works in these cases.

In our next example we use $n = 6$, and $k = 2, 3, 4$. We look at a variety of cases with different point locations to more fully examine how our tool works with n=6. In the previous cases, the results would be the same regardless of what the coordinates happen to be. In the following cases, however, the results are contingent upon the coordinates provided.

In our first case, we look at a convex hexagon, with initial coordinate points ([0, 0], [1, 0], [2, 1], [2, 2], [1, 3], [0, 2]).Where $k = 2$, we have 60 white triangles, 20 black triangles, and 70 hypertriangulations (Figure 20). Where $k = 3$, we have 60 white and black triangles, and 148 hypertriangulations (Figure 21). Finally, when $k = 4$, we have 20 white triangles, and 60 black triangles, and 70 hypertriangulations (Figure 22).

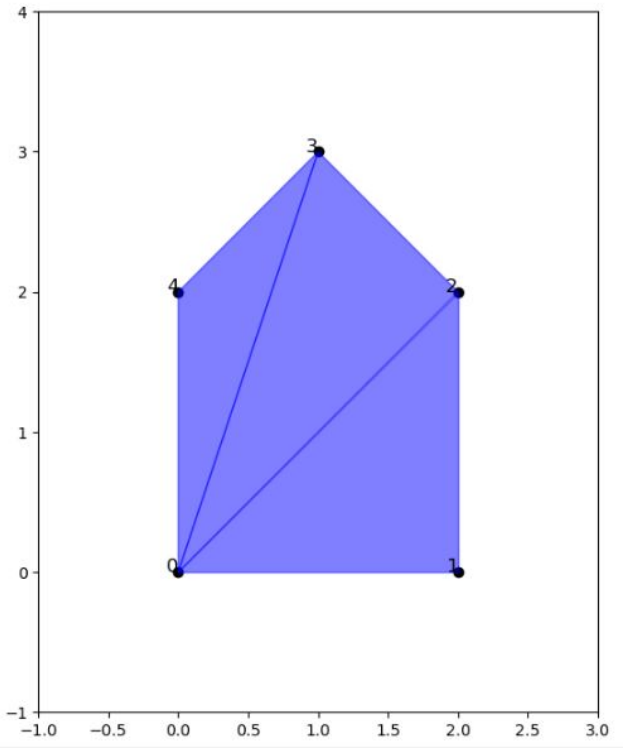


Figure 8: Hypertriangulations where n = 5 and k = 1. The point set has 10 White Triangles, 0 Black Triangles, and 5 Hypertriangulations, one of which is displayed. This Figure shows a hypertriangulation and the original points.

Our next case is of a convex pentagon with a point inside. Our initial coordinates are ([0, 0], [2, 0], [3, 1], [2, 3], [1, 2], [1, 1]). Here, [1, 1] is inside. When $k = 2$, we have 60 white triangles, 20 black triangles, and 106 hypertriangulations (Figure 23). Where $k = 3$, we have 60 black and white triangles, with a total of 240 hypertriangulations, and when $k = 4$ we have 20 black triangles, 60 white triangles and 106 hypertriangulations ((Figure 24 and Figure 25, respectively).

Next, we look at a quadrilateral with two points inside the polygon. The initial coordinates are ([0, 0], [4, 0], [4, 4], [0, 4], [2, 1], [2, 3]), where [2, 1], and [2, 3] are inside the polygon. When $k = 2$, we have 60 white triangles and 20 black triangles, and 130 hypertriangulations (Figure 26). When $k = 3$, we have 60 white triangles and 60 black triangles, and 320 hypertriangulations (Figure 27). When $k = 4$, we have 20 white triangles and 60 black triangles, and 130 hypertriangulations (Figure 28).

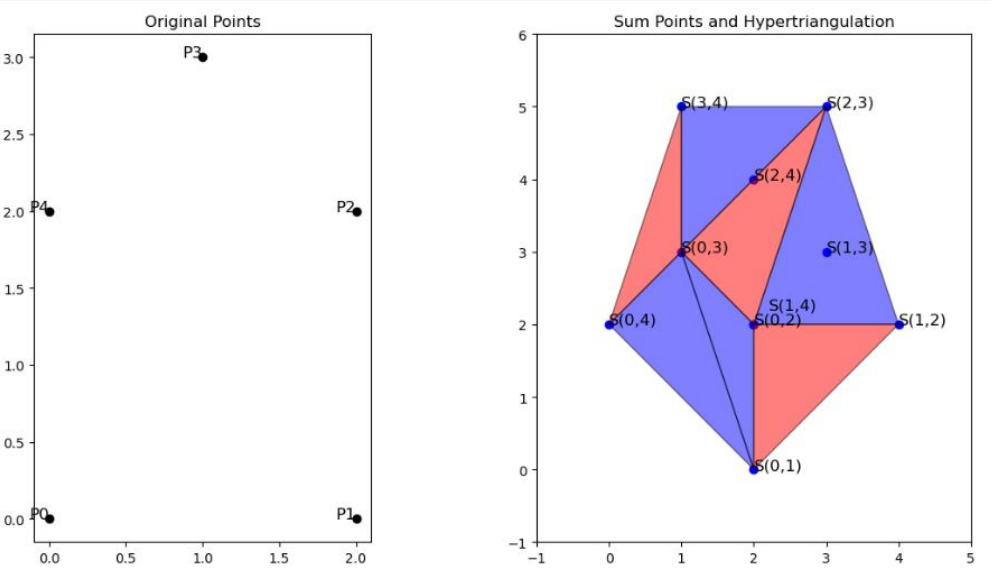


Figure 9: Hypertriangulations where n = 5 and k = 2. The point set has 20 White Triangles, 10 Black Triangles, and 5 Hypertriangulations, one of which is displayed. Notice that not all sums are included in the hypertriangulations, and some are included at the same point; this will not be mentioned in later figures.

Our final case involves a triangle with three points inside. Our initial coordinates are ([0, 0], [4, 0], [2, 7], [2, 1], [1, 3], [3, 3]), where [2, 1], [1, 3], and [3, 3] are inside the triangle. When $k = 2$, we have 60 white triangles and 20 black triangles, and 220 hypertriangulations (Figure 29). When $k = 3$, we have 60 white triangles and 60 black triangles, and 416 hypertriangulations (Figure 30). When $k = 4$, we have 20 white triangles and 60 black triangles, and 220 hypertriangulations (Figure 31).

These results show that our model works beyond the previously proven cases where $k = 1, 2$, to include results where $k = 3, 4$.

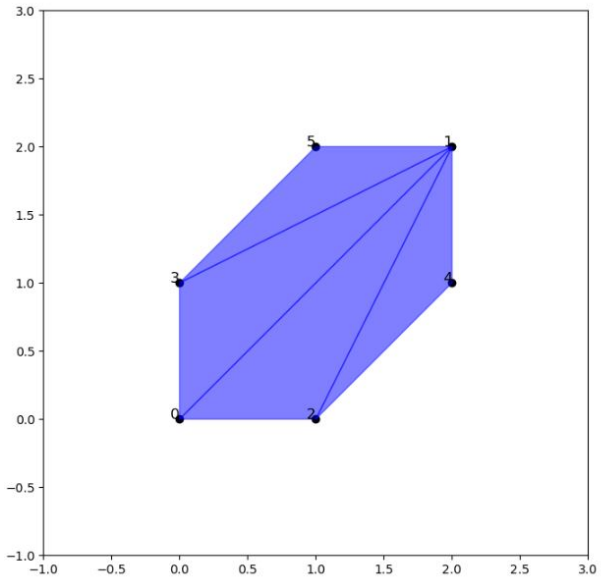


Figure 10: Hypertriangulations where n = 6 and k = 1. The point set has 20 White Triangles, 0 Black Triangles, and 14 Hypertriangulations, one of which is displayed. This Figure shows a hypertriangulation and the original points.

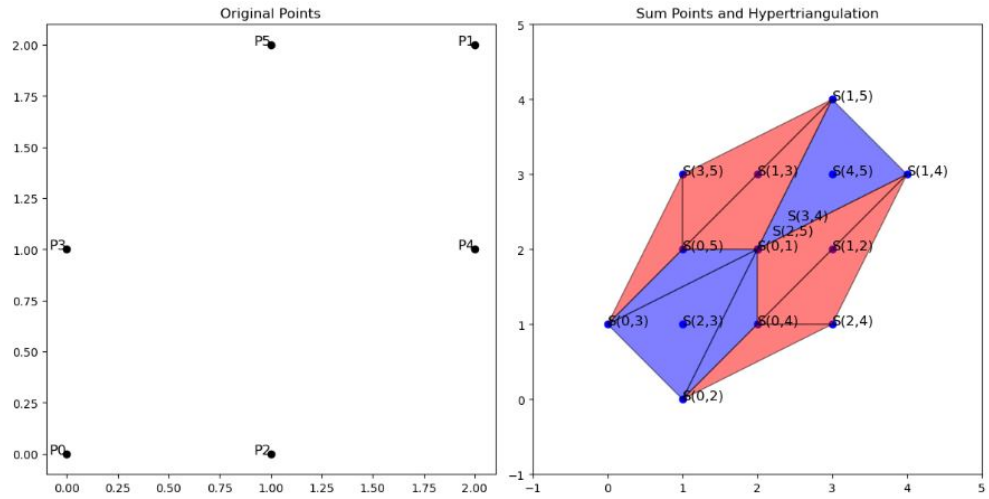


Figure 11: Hypertriangulations where n = 6 and k = 2. The point set has 60 White Triangles, 20 Black Triangles, and 70 Hypertriangulations, one of which is displayed.

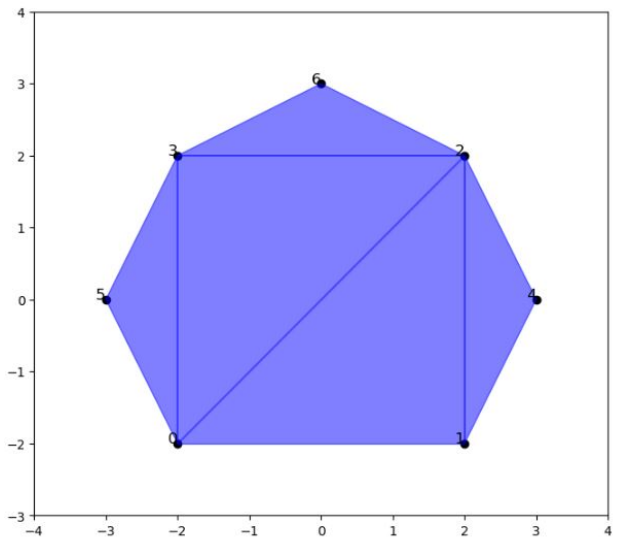


Figure 12: Hypertriangulations where n = 7 and k = 1. The point set has 35 White Triangles, 0 Black Triangles, and 42 Hypertriangulations, one of which is displayed. This Figure shows a hypertriangulation and the original points.

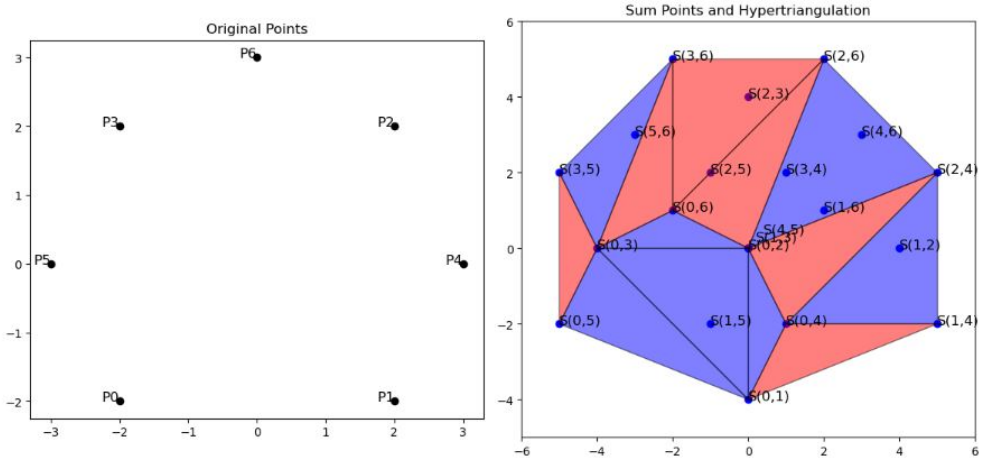


Figure 13: Hypertriangulations where n = 7 and k = 2. The point set has 140 White Triangles, 45 Black Triangles, and 574 Hypertriangulations, one of which is displayed.

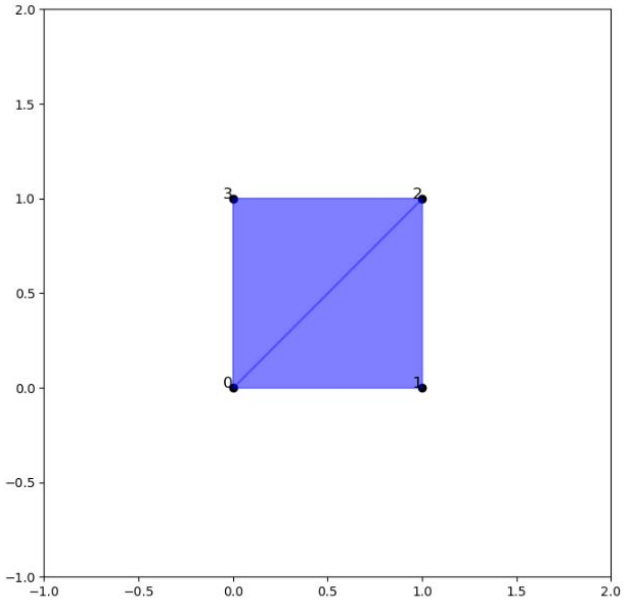
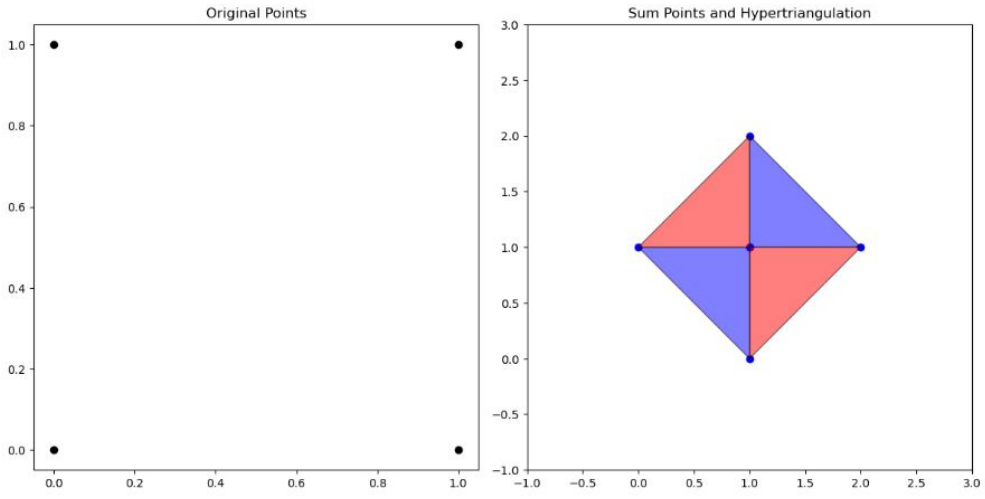


Figure 14: Hypertriangulations in the convex position where n=4 and k = 1. The point set has 4 White Triangles, 0 Black Triangles, and 2 Hypertriangulations, one of which is displayed.



l[ht]

Figure 15: Hypertriangulations in the convex position where n=4 and k = 2. The point set has 4 White Triangles, 4 Black Triangles, and 2 Hypertriangulations, one of which is displayed.

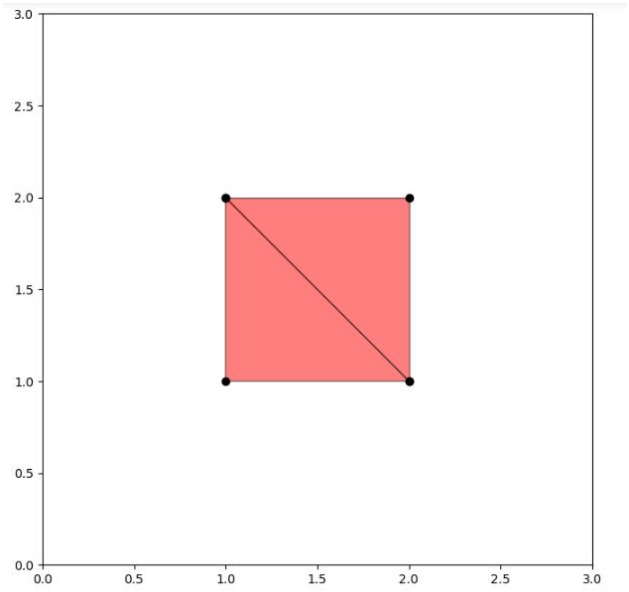


Figure 16: Hypertriangulations in the convex position where n=4 and k = 3. The point set has 4 White Triangles, 2 Black Triangles, and 2 Hypertriangulations, one of which is displayed. This shows the hypertriangulation and the original points.

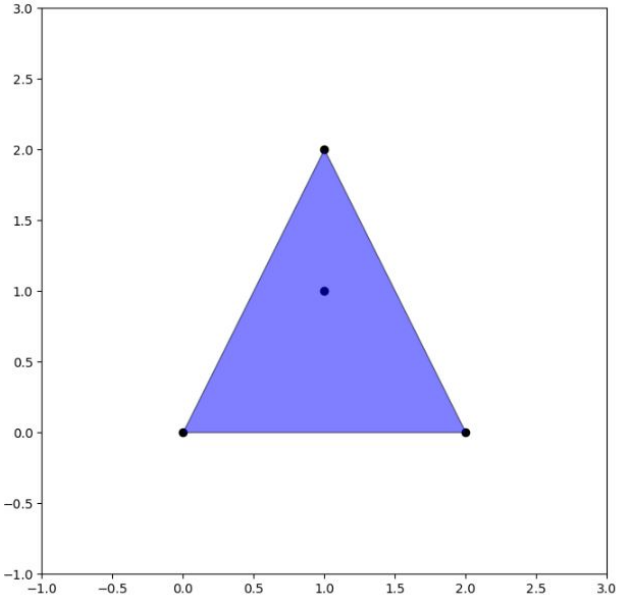


Figure 17: Points not in the convex position where n=4 and k = 1. The point set has 4 White Triangles, 0 Black Triangles, and 2 Hypertriangulations, one of which is displayed.

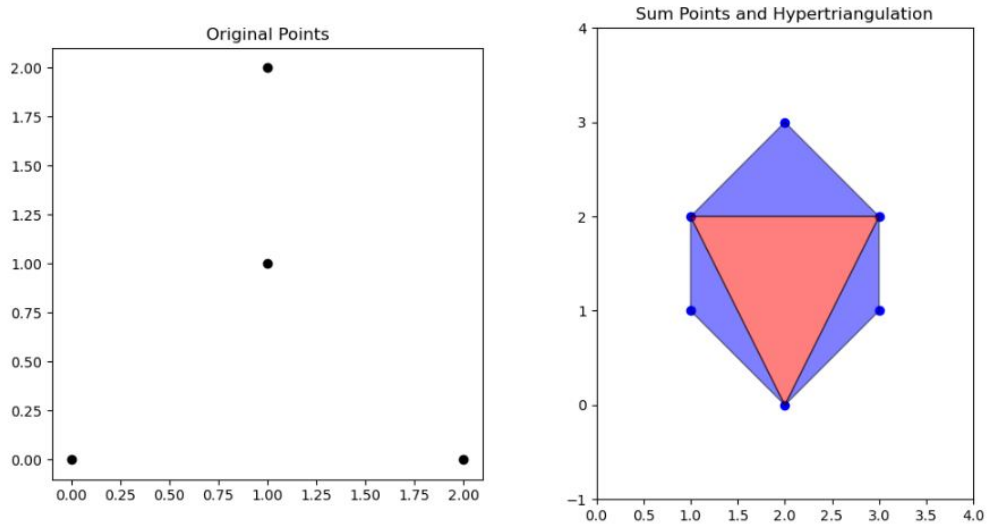


Figure 18: Points not in the convex position where n=4 and k = 2. The point set has 4 White Triangles, 8 Black Triangles, and 2 Hypertriangulations, one of which is displayed.

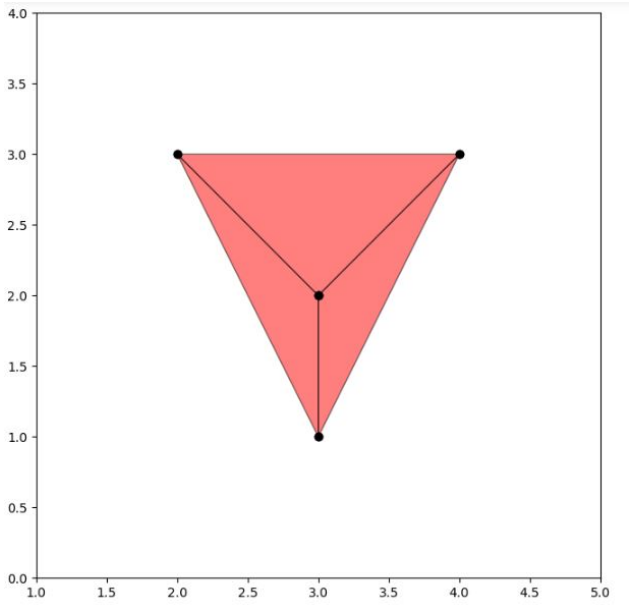


Figure 19: Points not in the convex position where n=4 and k = 3. The point set has 0 White Triangles, 4 Black Triangles, and 2 Hypertriangulations, one of which is displayed.

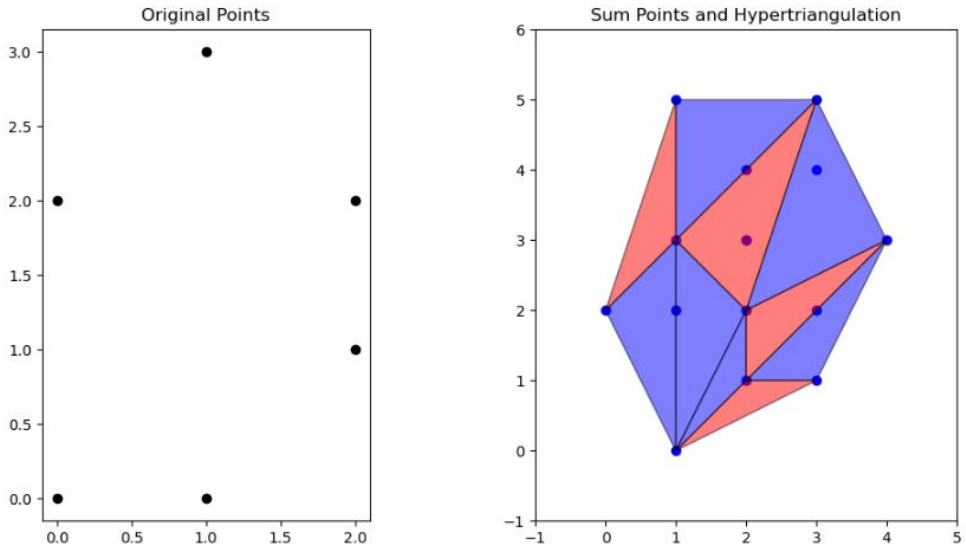


Figure 20: Hypertriangulations of a Convex Hexagon with n = 6 and k = 2. The point set has 60 White Triangles, 20 Black Triangles, and 70 Hypertriangulations, one of which is displayed.

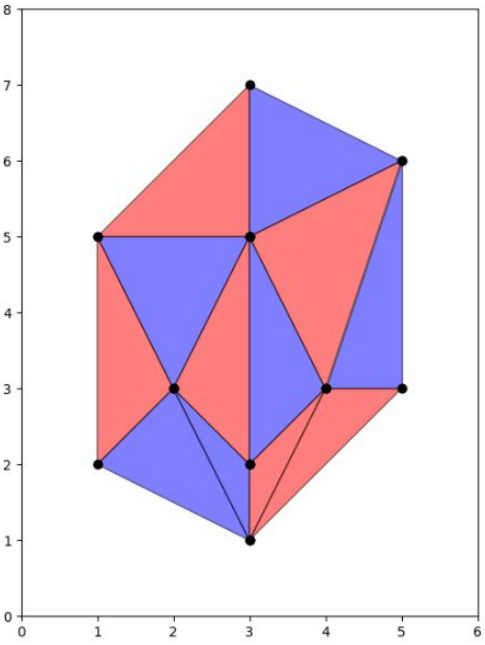


Figure 21: Hypertriangulations of a Convex Hexagon with n = 6 and k = 3. The point set has 60 White Triangles, 60 Black Triangles, and 148 Hypertriangulations, one of which is displayed. In this graph, and for future graphs, we only display the selected hypertriangulations and their points

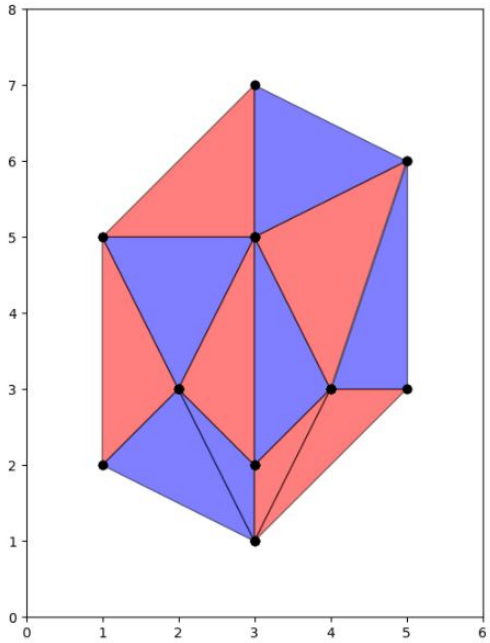


Figure 22: Hypertriangulations of a Convex Hexagon with n = 6 and k = 4. The point set has 60 White Triangles, 20 Black Triangles, and 70 Hypertriangulations, one of which is displayed.

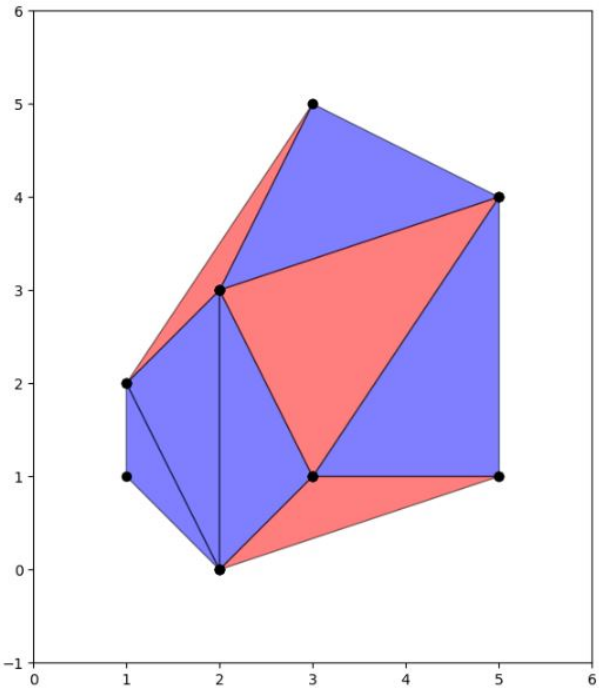


Figure 23: Hypertriangulations of a Convex Pentagon with n = 6 and k = 2. The point set has 60 White Triangles, 20 Black Triangles, and 106 Hypertriangulations, one of which is displayed.

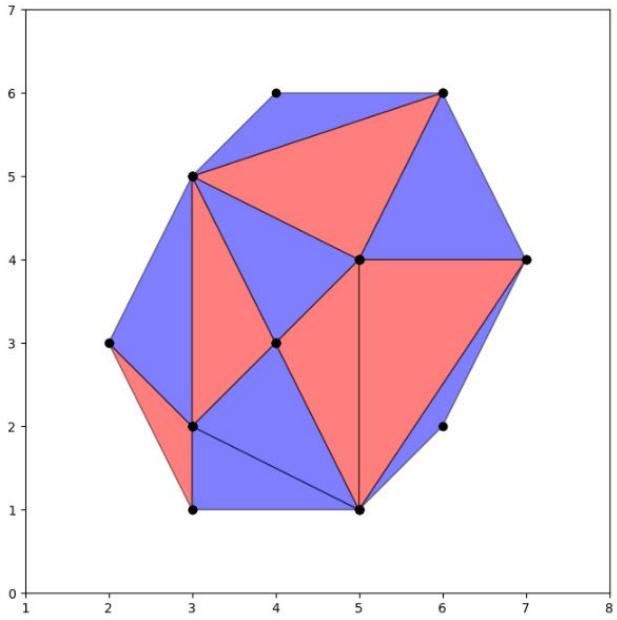


Figure 24: Hypertriangulations of a Convex Pentagon with n = 6 and k = 3. The point set has 60 White Triangles, 60 Black Triangles, and 240 Hypertriangulations, one of which is displayed.

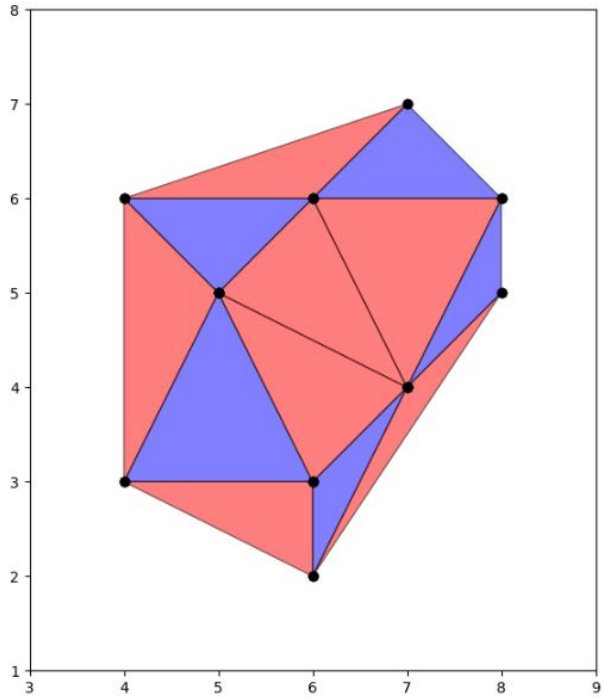


Figure 25: Hypertriangulations of a Convex Pentagon with n = 6 and k = 4. The point set has 20 White Triangles, 60 Black Triangles, and 106 Hypertriangulations, one of which is displayed.

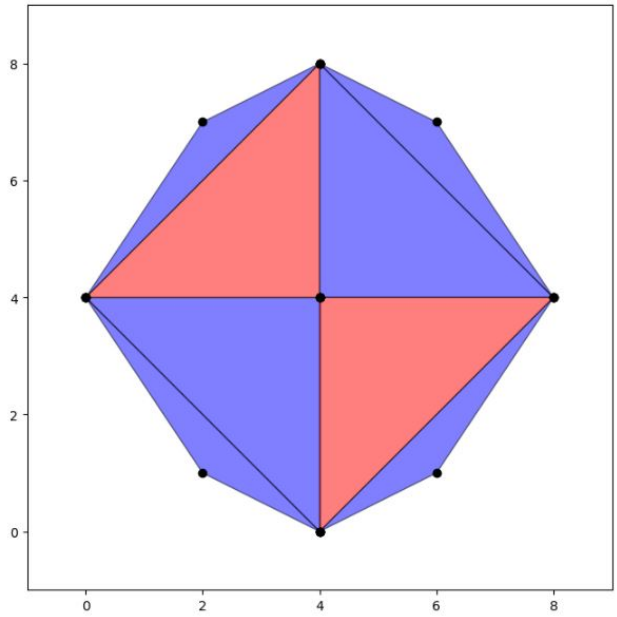


Figure 26: Hypertriangulations of a Quadrilateral with two points inside and n = 6 and k = 2. The point set has 60 White Triangles, 20 Black Triangles, and 130 Hypertriangulations, one of which is displayed.

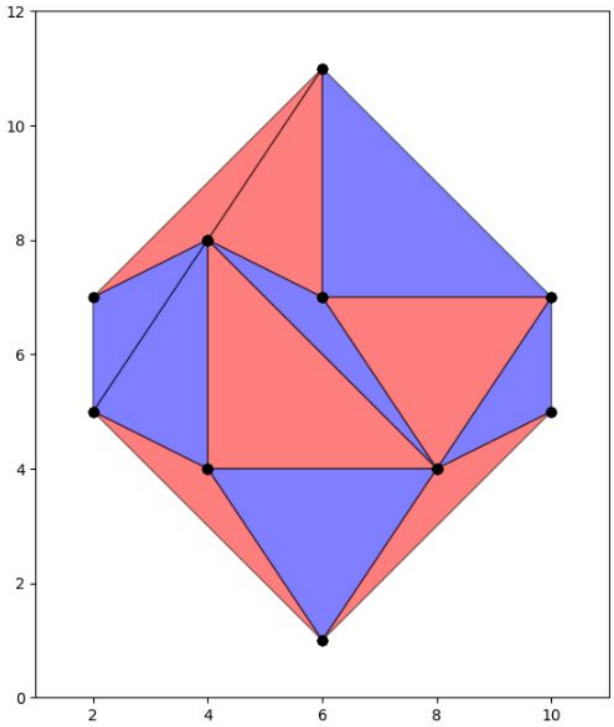


Figure 27: Hypertriangulations of a Quadrilateral with two points inside and n = 6 and k = 3. The point set has 60 White Triangles, 60 Black Triangles, and 320 Hypertriangulations, one of which is displayed.

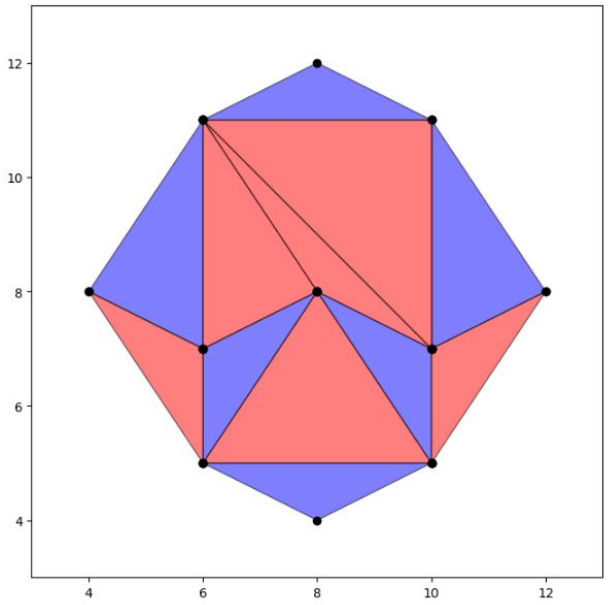


Figure 28: Hypertriangulations of a Quadrilateral with two points inside and n = 6 and k = 4. The point set has 20 White Triangles, 60 Black Triangles, and 130 Hypertriangulations, one of which is displayed.

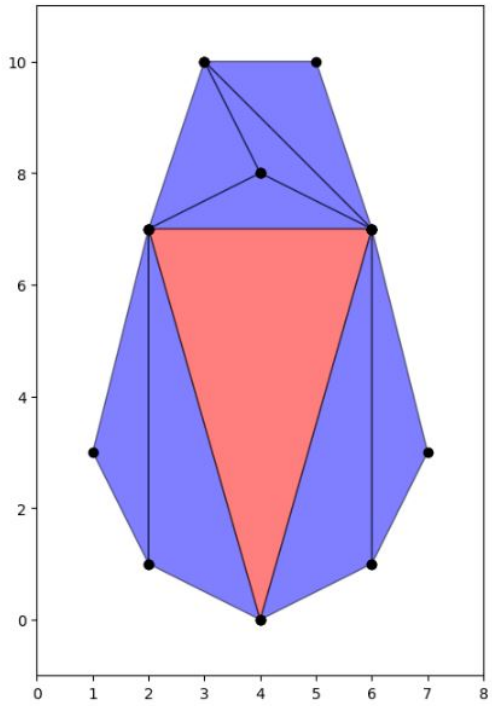


Figure 29: Hypertriangulations of a Triangle with 3 points inside and n = 6 and k = 2. The point set has 60 White Triangles, 20 Black Triangles, and 220 Hypertriangulations, one of which is displayed.

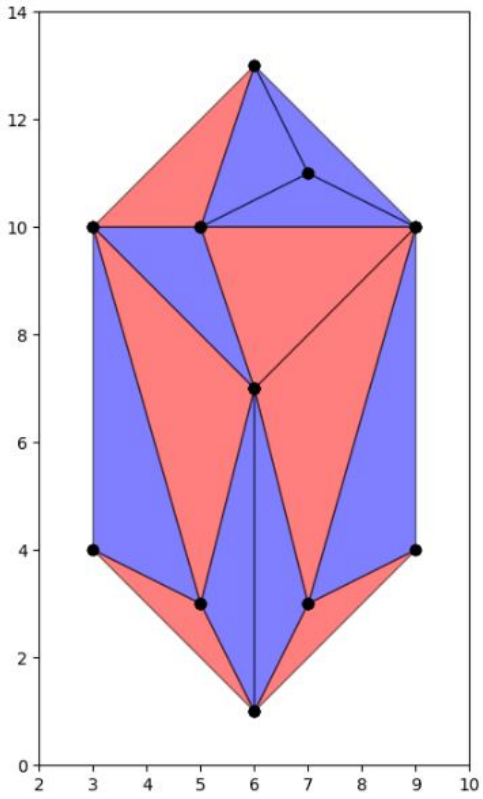


Figure 30: Hypertriangulations of a Triangle with 3 points inside and n = 6 and k = 3. The point set has 60 White Triangles, 60 Black Triangles, and 416 Hypertriangulations, one of which is displayed.

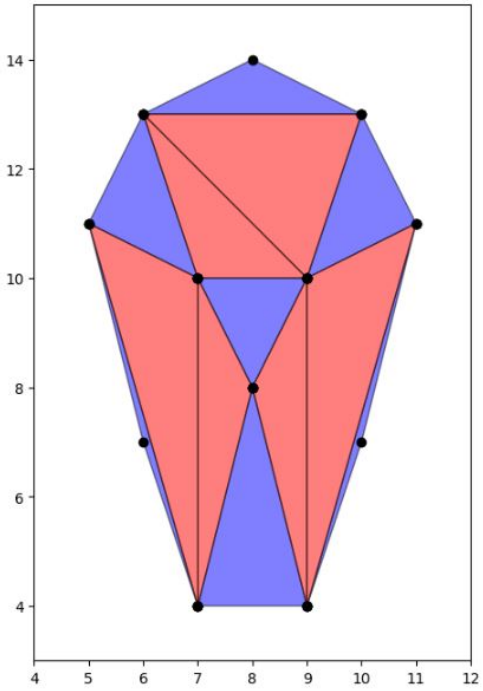


Figure 31: Hypertriangulations of a Triangle with 3 points inside and n = 6 and k = 4. The point set has 20 White Triangles, 60 Black Triangles, and 220 Hypertriangulations, one of which is displayed.

CHAPTER VI

FUTURE RESEARCH

We have shown that our algorithm works where k is equal to 3 or 4. Future research should first look to better optimize our DFS algorithm. Indeed, in several instances when k was 3, and n was greater than 6 the code took much longer to run. These algorithms are slow and computational expensive, and it would be beneficial to improve upon this. In addition, as we pointed out flip-connectivity has not been shown to work beyond k=2. Future research should also examine methods to improve upon flips to see if there is a way to incorporate them into algorithms. Finally, effort should be put into creating a substitute for the python program shapely so that the algorithm can work with non-integers.

REFERENCES

[1] J. A. DE LOERA, J. RAMBAU, AND F. SANTOS, *Triangulations: Structures for Algorithms and Applications*, Springer Publishing Company, Incorporated, 1st ed., 2010.

[2] H. EDELSBRUNNER, A. GARBER, M. GHAFARI, T. HEISS, AND M. SAGHAFIAN, *Flips in two-dimensional hypertriangulations*. `https://arxiv.org/abs/2212.11380`, 2024.

[3] J. A. OLARTE AND F. SANTOS, *Hypersimplicial subdivisions*, Nov. 2021.

[4] H. OYVIND AND M. DAEHLEN, *Triangulations and Applicatins*, Springer Science & Business Media, 2006.

[5] G. M. ZIEGLER, *Lectures on Polytopes: Volume 152 of Graduate Texts in Mathematics*, Springer Science & Business Media, 1st ed., 2012.

VITA

Patrick was born and raised in an old steel mill town in western Pennsylvania. Patrick earned his Master of Science (MS) in August, 2024, from the University of Texas Rio Grande Valley. During his 20s, he worked as a community activist in Philadelphia with low-income tenants. In his early to mid-30s he worked in behavioral health, and was at one point the Manager of Clinical Program Evaluation for the Department of Behavioral Health/Community Behavioral Health (Philadelphia). All the while he was attending graduate school, and he earned 2 master's degrees and a PhD. After he earned his PhD, he joined the United States Army, and served with the 101st Airborne Division (Air Assault), 82nd Airborne Division (ABN), 32nd Medical Brigade, and other units. While on active duty he deployed to Iraq. He was honorably discharged from the US Army. He then returned to graduate school and was employed by the United States Department of Treasury, Internal Revenue Service where he works in research. He currently holds two advanced degrees from the University of Pennsylvania, a PhD from Bryn Mawr College, and other degrees. He was married to Saamara Sanchez in 2006, and they live together in Philadelphia, PA with their two English Mastiffs (Jax and Tiger), and their Rottweiler (Winnie). He can be reached at pkaylor1969@gmail.com.