

**PolyPathLab: Software for the Calculation of Monotone Paths on  
Polytopes**

By

LIMIN HUANG

SENIOR THESIS

Submitted in partial satisfaction of the requirements for Highest Honors for the degree of

BACHELOR OF SCIENCE

in

MATHEMATICAL ANALYTICS AND OPERATIONS RESEARCH

in the

COLLEGE OF LETTERS AND SCIENCE

of the

UNIVERSITY OF CALIFORNIA,

DAVIS

Approved:

---

Jesús A. De Loera

June 2020

## ABSTRACT.

Linear programming (LP) and the simplex method are closely related to the geometry of polytopes: the feasible region of a bounded and feasible LP problem is a convex polytope. Each vertex is a basic feasible solution (BFS) to the related LP problem. An objective function induces a directed graph on the polytope, where each edge has an orientation toward the vertex with a better objective value. Motivated by open problems related to the efficiency of the simplex method, we build *PolyPathLab*, a package based on MATLAB version R2019a [18]. *PolyPathLab* can compute the following features:

- the *diameter*, which is the maximum distance between vertices
- the *monotone diameter*, which is the maximum monotone distance between vertices connected by monotone paths
- the *monotone height*, which is the length of the longest monotone path
- the *number of monotone paths*
- the *number of arborescences*
- characteristics related to four famous pivot rules: *Dantzig's rule*, *greatest descent*, *steepest edge*, and *Bland's rule*
- the *flip graph* and the *diameter of the flip graph*

*PolyPathLab* is the succeeding work of the undergraduate thesis of Chutong Wu [26] and will be available online for all researchers to experiment.

# Contents

Chapter 1. Introduction	1
1.1. Hyperplanes and Polytopes	1
1.2. Linear Programming	2
1.3. Graphs	3
1.4. Simplex Method and Pivot Rules	7
1.5. Monotone paths	11
1.6. Flips	13
Chapter 2. Main Algorithms	18
2.1. Random 3-dimensional Polytope Generator	19
2.2. Diameter	19
2.3. Monotone Diameter	21
2.4. Monotone Height	23
2.5. Number of Monotone Paths	24
2.6. Number of Arborescences	25
2.7. Characteristics Related to Pivot Rules	26
2.8. Flip Graph	26
Chapter 3. PolyPathLab Software	27
3.1. Creating Input Files Using the Program cdd/cdd+ (Step 1 and 2)	27
3.2. Creating the Objective Function File (Step 3)	31
3.3. What to Put in the Input Folder and the File-naming Convention (Step 4)	32
3.4. Computation Stage for the Features Except the Flip Graph (Step 5)	33
3.5. Computation Stage for the Flip Graph (Step 5)	39
Chapter 4. Computational Results	41
4.1. Choice of 3-dimensional Polytopes	41
4.2. Characteristics of Platonic Solids and Archimedean Solids	41
4.3. The Special Cases of Archimedean Solids: the Snub Cube and the Snub Dodecahedron	51
4.4. The Monotone Diameter and the Monotone Height of $3d$ Simple Polytopes	60
4.5. The Monotone Height of Birkhoff Polytope	65
4.6. The Monotone Height of the Traveling Salesman Polytope	68

Acknowledgements

70

Bibliography

71

## CHAPTER 1

# Introduction

In this thesis, we introduce a software package, *PolyPathLab*, for the calculation of various features on polytopes. We structure the thesis as the follow: in Chapter 1, we begin with some definition, together with some background and motivations of the features we compute with *PolyPathLab*. Then we continue in Chapter 2 with some pseudocode on how *PolyPathLab* computes each feature. In Chapter 3, we explain how to use the software with some examples. Finally, in Chapter 4, we present the results of the experiments we did using *PolyPathLab*, including the features of the *Platonic Solids* and the *Archimedean Solids*, the distribution of the monotone diameters and the monotone heights for random 3-dimensional polytopes, and the distribution of the monotone heights for Birkhoff polytopes and the TSP polytopes.

### 1.1. Hyperplanes and Polytopes

DEFINITION 1. Let  $a_1, a_2, \dots, a_n \in \mathbb{R}$  and at least one of them is not zero. Let  $c \in \mathbb{R}$  be a constant. A **hyperplane** is the set of all vectors  $\vec{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n$  such that  $a_1x_1 + a_2x_2 + \dots + a_nx_n = c$ . Geometrically, a hyperplane in  $\mathbb{R}^n$  is an affine subspace with dimension  $n - 1$ .

DEFINITION 2. A hyperplane in  $\mathbb{R}^n$  separates  $\mathbb{R}^n$  into two **halfspaces**, which are the sets  $\{\vec{x} \in \mathbb{R}^n : a_1x_1 + a_2x_2 + \dots + a_nx_n > c\}$  and  $\{\vec{x} \in \mathbb{R}^n : a_1x_1 + a_2x_2 + \dots + a_nx_n < c\}$ . A hyperplane together with one of the halfspaces is called a **generalized halfspace**.

DEFINITION 3. A set  $S$  is **convex** if  $\forall x, y \in S, z = \lambda x + (1 - \lambda)y \in S, 0 < \lambda < 1$ .

DEFINITION 4. A **convex polyhedron** is the intersection of finitely many generalized halfspaces. A **polytope** is a bounded convex polyhedron.

DEFINITION 5. A  **$d$ -dimensional simple polytope** is a  $d$ -dimension polytope where each vertex is included in at most  $d$  edges.

Simple polytopes are corresponding to non-degenerate LP problems [9]. The maximum *diameter* of a  $d$ -dimensional polytope with  $n$  facets is achieved by simple polytopes since all polytopes can be perturbed into simple polytopes with diameters at least as large [13].

## 1.2. Linear Programming

Linear programming (LP) is a method of maximizing or minimizing a linear objective function with respect to the constraints, which are a system of linear equations or inequalities. It can be written in the *canonical form* as:

$$\begin{aligned} & \text{maximize} && c^T x \\ & \text{subject to} && Ax \leq b \\ & && x \geq 0, \end{aligned}$$

where  $c \in \mathbb{R}^d$ ,  $b \in \mathbb{R}^n$ ,  $A \in \mathbb{R}^{n \times d}$ , and  $x \in \mathbb{R}^d$ .

Each constraint is a generalized halfspace, so the constraints of the LP form a convex polyhedron.

The feasible region of every bounded LP problem is a polytope  $P$ , and each vertex of  $P$  is a basic feasible solution (BFS) to the LP. For example, the 3-dimensional unit cube in Figure 1 comes from an LP which has inequalities

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad x_1, x_2, x_3 \geq 0.$$

This cube has 8 vertices, representing the 8 possible basic feasible solutions to the LP, and 6 facets, representing the 6 inequalities in the LP.

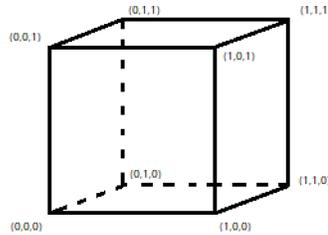


FIGURE 1. A 3d unit cube.

In this thesis, we will consider the LP to be maximizing the objective function, which is the same as minimizing the opposite of the objective function.

### 1.3. Graphs

DEFINITION 6. A **graph**  $G$  is a pair  $G = (V, E)$ , where  $V$  is a set whose elements are called the vertices or nodes, and  $E$  is a set of edges — a set of pairs of two vertices.

In this thesis, we will focus on the graphs that do not contain self-loop or parallel edges.

DEFINITION 7. A **path** is a sequence of distinct vertices where the consecutive pairs are joined by edges.

DEFINITION 8. A graph  $G$  is **connected** if any pair of vertices in  $G$  is connected by a path in  $G$ . A connected graph  $G$  is  **$n$ -connected** if for any set  $S$  of  $n$  vertices in  $G$ ,  $G - S$  is connected.

DEFINITION 9. A **directed edge**  $(i, j)$  is an edge that has a direction from vertex  $i$  to vertex  $j$ . It is an incoming edge for vertex  $j$  and an outgoing edge for vertex  $i$ .

DEFINITION 10. A **directed graph** is a graph that contains only directed edges.

DEFINITION 11. The **source** of a directed graph is the vertex that only has outgoing edges.

DEFINITION 12. The **sink** of a directed graph is the vertex that only has incoming edges.

DEFINITION 13. A **tree** is a minimally connected graph, meaning that deleting any edge will lead to a graph that is not connected. Equivalently, a tree of  $n$  vertices is a connected graph with  $n - 1$  edges.

DEFINITION 14. An **arborescence** is a directed, rooted tree, where for a vertex  $u$  called the root and any other vertex  $v$ , there is exactly one path from  $v$  to  $u$ .

DEFINITION 15. A directed graph is called **cyclic** if it contains at least one cycle. A directed graph is called **acyclic** if there is no cycle in the graph.

Figure 2 below is an example of a cyclic graph.

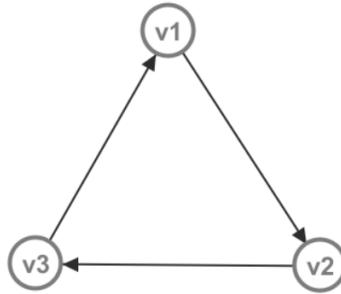


FIGURE 2. An example of a cyclic graph.

We now introduce the some definitions on the *polyhedral graph* and the *directed polyhedral graph*.

DEFINITION 16. The **1-skeleton** or the **polyhedral graph** of a polytope  $P$  is a graph  $G = (V, E)$  where  $V$  is the set of vertices of  $P$  and  $E$  is the set of edges of  $P$ .

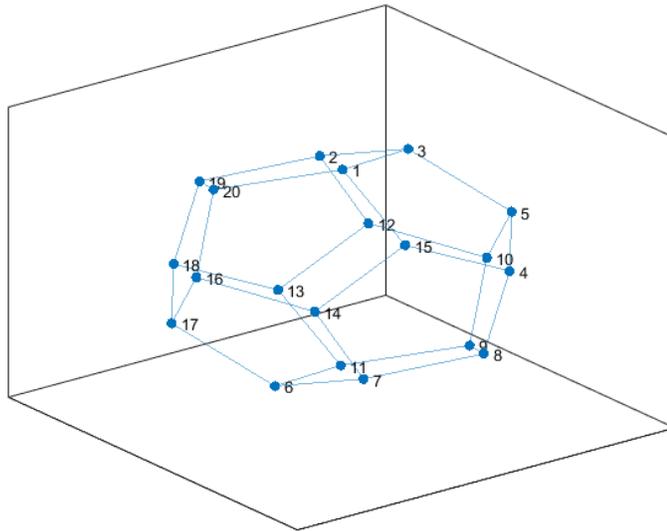


FIGURE 3. The 1-skeleton of the *dodecahedron*.

An objective function  $f$  gives an orientation on the polyhedral graph  $G$  of a polytope  $P$ .  $f$  assigns an objective value to each vertex of  $P$ , and each edge is oriented toward the vertex with the higher objective value. An improving edge for vertex  $v$  is an edge that connects  $v$  with another vertex which has a higher objective value. An incoming edge for vertex  $v$  is an edge that connects  $v$  to another vertex which has a lower objective value.

An objective function on a polytope is *generic* if no two vertices have the same objective value. A polytope with an objective function that is generic will produce a directed acyclic polyhedral graph, and there is a unique source and a unique sink on each face of the polytope. The graph is acyclic since a cycle of vertices  $v_1$  to  $v_n$  will lead to objective values  $f(v_1) < f(v_2) < \dots < f(v_n) < f(v_1)$ , which is a contradiction.

In this thesis, we only consider the directed polyhedral graphs  $DG$  coming from polytopes with generic objective functions. A directed polyhedral graph from a polytope and a generic objective function has a unique source and a unique sink. Applying the objective function  $x + y + z$ , we can see that the source of the unit cube shown in Figure 1 is the vertex  $[0, 0, 0]$  with an objective value zero, and the sink is vertex  $[1, 1, 1]$  with an objective value six.

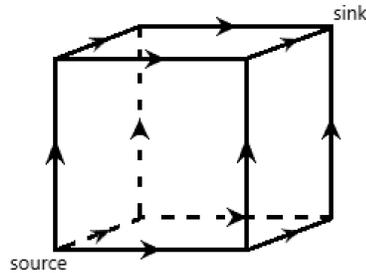


FIGURE 4. A 3d unit cube with the edge directions and labels for the source and the sink.

DEFINITION 17. A **monotone path** is a path that each pair of consecutive vertices are joined by an **improving edge**.

Let  $i$  be any vertex that is not the source  $s$  and the sink  $t$  in a directed polyhedral graph  $DG$ . There is at least one monotone path from  $s$  to  $i$  and from  $i$  to  $t$ . To prove this, we follow one of the incoming edges for  $i$  and trace back to the vertex going into  $i$ . We repeat the process of tracing back from the current vertex until we reach a vertex  $v$  with no incoming edge. By definition,  $v$  is a source. Since there is a unique source in  $DG$ ,  $v$  is the source  $s$ , and the sequence of incoming edges form a monotone path from  $s$  to  $i$ . We can use a similar argument to prove that there is at least one monotone path from  $i$  to  $t$ .

DEFINITION 18. The **length** of a path or a monotone path is the number of edges in the path.

DEFINITION 19. The **distance** between two vertices  $i$  and  $j$  of a connected graph with finite number of vertices is the length of the shortest path between  $i$  and  $j$ .

An  $d$ -dimensional undirected polyhedral graph is  $d$ -connected, which has  $d$  distinct paths (only have the end points in common) joining any pair of vertices in the graph [10]. Thus, we have a distance between any two vertices of the 1-skeleton of a polytope  $P$ .

DEFINITION 20. For a directed graph, the **monotone distance** between two vertices  $i$  and  $j$  that are connected by monotone paths is the length of the shortest monotone path between  $i$  and  $j$ .

DEFINITION 21. For a graph  $G$  with  $n$  vertices, the **adjacency matrix**  $A$  of  $G$  is an  $n \times n$  matrix such that the entry  $A_{i,j}$  is one if there is an edge  $(i,j)$  in  $P$ ; the entry is zero otherwise.

In Figure 5, vertex  $i$  is connected to vertex  $j$  by an edge, so  $A_{i,j} = A_{j,i} = 1$ . On the other hand, vertex  $i$  is not connected to vertex  $k$ , so  $A_{i,k} = A_{k,i} = 0$ .

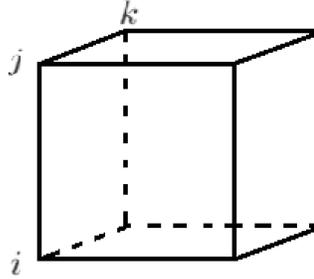
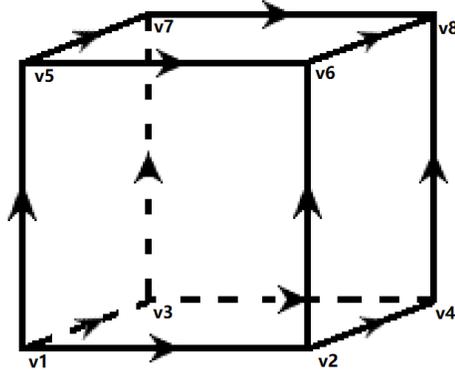


FIGURE 5. A  $3d$  unit cube. Vertex  $i$  and  $j$  are connected; vertex  $i$  and  $k$  are not connected.

DEFINITION 22. The **directed adjacency matrix**  $D$  of a directed graph with  $n$  vertices is an  $n \times n$  matrix such that  $D_{i,j} = 1$  if  $(i,j) \in E$ , and  $D_{i,j} = 0$  otherwise.

If the objective function is  $x + y + z$ , Figure 6 shows the directed 1-skeleton of a unit cube with the objective function.

FIGURE 6. A  $3d$  unit cube with the edge directions.

The directed adjacency matrix  $D$ , with an “1” in the  $i$ -th row,  $j$ -th column representing an out-going edge from  $v_i$  to  $v_j$  will be:

$$D = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

#### 1.4. Simplex Method and Pivot Rules

**Simplex method**, invented and developed by George Bernard Dantzig, changed the field of optimization dramatically. When introduced, the simplex method became the basis of many further branches of LP, such as integer linear programming and nonlinear programming. Even today, the simplex method can outperform and compete with many modern algorithms [22].

We now introduce how the simplex method works. First, we find a BFS to the LP problem, which is a random vertex of the related polytope  $P$ . Each iteration of the simplex method starts from the current vertex and goes along an edge to a vertex with a greater or equal objective value.

When performing the simplex method, we usually have more than one possible improving edges during one iteration. Thus, we need to use a *pivot rule* to tell us which edge the simplex method will go along.

DEFINITION 23. A **pivot rule** specifies which improving edge the simplex method picks. Each pivot rule only picks one edge from every vertex of  $P$  except the optimum.

When the LP problem is *degenerate*, the simplex method may run into a cycle, which means that the simplex method will revisit a vertex. However, if a “good” pivot rule is chosen, the simplex method does not cycle and terminates at the optimum.

We consider four famous pivot rules in this thesis: Dantzig’s rule, greatest descent, steepest edge, and Bland’s rule. For an LP problem, consider the *standard form*:

$$\begin{aligned} & \text{maximize} && c^T x \\ & \text{subject to} && Ax = b \\ & && x \geq 0. \end{aligned}$$

$x = [x_B \ x_N]^T$  is an  $(n + d) \times 1$  vector, where  $x_B$  are the basic variables and  $x_N$  are the non-basic variables. Denote the index set of  $x_B$  as  $\mathbf{B}$  and the index set of  $x_N$  as  $\mathbf{N}$ .  $c = [c_B \ c_N]^T$  is an  $(n + d) \times 1$  vector, where  $c_B$  are associated with  $x_B$ , and  $c_N$  are associated with  $x_N$ .  $A = [B \ N]$  is an  $n \times (n + d)$  matrix where  $B$  is an  $n \times n$  sub-matrix that associated with  $x_B$  and  $N$  being an  $n \times d$  sub-matrix associated with  $x_N$ .

Rewriting the formulation, we get:

$$\begin{aligned} & \text{maximize} && c_B^T x_B + c_N^T x_N \\ & \text{subject to} && Bx_B + Nx_N = b \\ & && x_B, x_N \geq 0. \end{aligned}$$

Solving for  $x_B$ , we get:

$$\begin{aligned} & \text{maximize} && c_B^T x_B + c_N^T x_N \\ & \text{subject to} && x_B = B^{-1}b - B^{-1}Nx_N \\ & && x_B, x_N \geq 0. \end{aligned}$$

Substituting the  $x_B$  in the objective function, we get:

$$\begin{aligned} & \text{maximize} && c_B^T (B^{-1}b - B^{-1}Nx_N) + c_N^T x_N \\ & \text{subject to} && x_B = B^{-1}b - B^{-1}Nx_N \\ & && x_B, x_N \geq 0. \end{aligned}$$

Simplify and we obtain:

$$\begin{aligned}
& \text{maximize} && c_B^T B^{-1}b - (B^{-1}N^T c_B + c_N)^T x_N \\
& \text{subject to} && x_B = B^{-1}b - B^{-1}N x_N \\
& && x_B, x_N \geq 0.
\end{aligned}$$

$c_B^T B^{-1}b$  is the current objective value  $\xi$ . Let  $\bar{c}_N = (B^{-1}N^T c_B + c_N)^T$ ,  $\bar{b} = B^{-1}b$ ,  $\bar{N} = B^{-1}N$ , we have:

$$\begin{aligned}
& \text{maximize} && \xi - \bar{c}_N^T x_N \\
& \text{subject to} && x_B = \bar{b} - \bar{N} x_N \\
& && x_B, x_N \geq 0.
\end{aligned}$$

If all entries of  $\bar{c}_N \leq 0$ , the solution reaches the optimum. Otherwise, the simplex method will pick an entering variable from  $x_N$  into  $x_B$  and a leaving variable from  $x_B$  into  $x_N$ .

Each pivot rule uses different conditions to choose the entering variable. Assume that  $x_i$ ,  $i \in \mathbf{B}$ , is the entering variable. Then, the leaving variable is always going to be  $x_j$ ,  $j \in \mathbf{N}$  where  $\bar{b}_i / \bar{N}_{i,j}$  is the minimum of  $\bar{b}_i / \bar{N}_{i,j} \forall j$  [23].

Consider a tetrahedron, shown below:

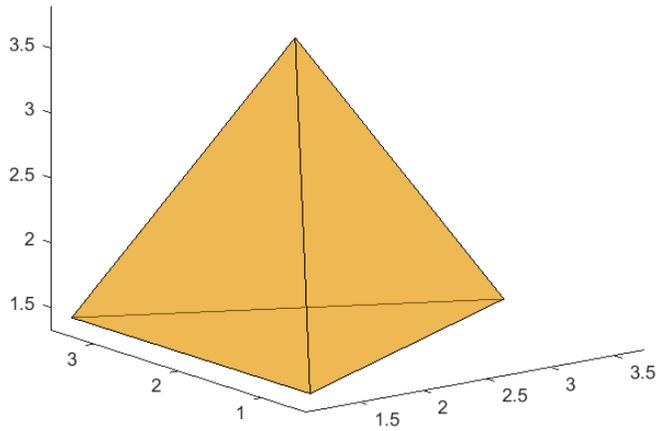


FIGURE 7. A tetrahedron produced by “plotregion” [4].

The tetrahedron is related to the following LP problem. Each facet of the tetrahedron is one of the constraints, and the coordinates of the vertices are the BFS. The *canonical form* of the LP with an objective function  $x_1 + 2x_2 + \frac{1}{2}x_3$  is:

$$\begin{aligned} \text{maximize} \quad & x_1 + 2x_2 + \frac{1}{2}x_3 \\ \text{subject to} \quad & 1.4x_1 + 2.4x_2 + x_3 \leq 11.5 \\ & 0x_1 + 0x_2 - x_3 \leq -1.4 \\ & 1.4x_1 - 2.4x_2 + x_3 \leq 1.7 \\ & -2.8x_1 + 0x_2 + x_3 \leq -1.9 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

The standard form of the above example is:

$$\begin{aligned} \text{maximize} \quad & x_1 + 2x_2 + \frac{1}{2}x_3 \\ \text{subject to} \quad & x_4 = \frac{23}{2} - \frac{7}{5}x_1 - \frac{12}{5}x_2 - x_3 \\ & x_5 = -\frac{7}{5} + x_3 \\ & x_6 = \frac{17}{10} - \frac{7}{5}x_1 + \frac{12}{5}x_2 - x_3 \\ & x_7 = -\frac{19}{10} + \frac{14}{5}x_1 - x_3 \\ & x_i \geq 0, i = 1, 2, \dots, 7. \end{aligned}$$

We start with an initial basic feasible solution by setting  $x_5 = 0, x_6 = 0, x_7 = 0$  and solve the system. Then, the system becomes:

$$\begin{aligned} \text{maximize} \quad & \frac{841}{280} + \frac{59}{28}x_5 + \frac{5}{6}x_6 + \frac{65}{84}x_7 \\ \text{subject to} \quad & x_1 = \frac{33}{28} + \frac{5}{14}x_5 + \frac{5}{14}x_7 \\ & x_2 = \frac{9}{16} + \frac{5}{8}x_5 + \frac{5}{12}x_6 + \frac{5}{24}x_7 \\ & x_3 = \frac{7}{5} + x_5 \\ & x_4 = \frac{71}{10} - 3x_5 - x_6 - x_7 \\ & x_i \geq 0, i = 1, 2, \dots, 7. \end{aligned}$$

We will use the above example to introduce the four famous pivot rules.

**Dantzig's rule** picks the index of the maximum positive entry of  $\bar{c}_N$  as the entering variable. In practice, Dantzig's rule will fail sometimes due to degeneracy [23]. When

there is degeneracy, and there are two possible choices of entering or leaving variables, we will use the smallest index as the entering or leaving variable.

In the above example, the maximum positive entry of  $\overline{c}_N$  is  $\overline{c}_5 = \frac{59}{28}$ , so  $x_5$  will be the entering variable, while  $x_4$  will leave the basis.

**Greatest descent or greatest improvement** picks the entering variable to be the index  $j \in \mathbf{N}$  of the maximum of  $\overline{c}_j * \overline{b}_i / \overline{N}_{i,j} \forall i \in \mathbf{B}$ . The leaving variable will be the corresponding  $i$  [14].

In the above example, computing  $\overline{c}_j * \overline{b}_i / \overline{N}_{i,j}$ , we get that  $\overline{c}_6 * \overline{b}_4 / \overline{N}_{4,6} = \frac{5}{6} * \frac{71}{10} / 3$  is the biggest. Thus,  $x_6$  will be the entering variable, and  $x_4$  will leave the basis.

**Steepest edge** finds the edge that gives the biggest increase in the objective value per unit length. In order to find the edge, we compute  $\overline{c}_j / \sqrt{\sum_{i \in \mathbf{B}} \overline{N}_{i,j}^2}$  and find the maximum in all possible  $j$ . After finding the  $j$ ,  $x_j$  will be the entering variable [8].

In the above example, computing  $\overline{c}_j / \sqrt{\sum_{i \in \mathbf{B}} \overline{N}_{i,j}^2}$ , we get that  $\overline{c}_7 / \sqrt{\sum_{i \in \mathbf{B}} \overline{N}_{i,7}^2} = \frac{5}{6} \div \sqrt{0^2 + (\frac{5}{12})^2 + (0)^2 + (-1)^2} \approx 0.77$ , which is greater than 0.65 for index 5 and 0.72 for index 7, so  $x_6$  will enter the basis.

**Bland's rule** picks the smallest index of the positive entries of  $c_b$  as the entering variable. If there are two or more possible leaving variables, the variable with the smallest index becomes the leaving variable [23].

In the above example, since  $\overline{c}_5, \overline{c}_6, \overline{c}_7 > 0$ , the Bland's rule will pick the smallest index, so  $x_5$  will be the entering variable.

### 1.5. Monotone paths

The simplex method can solve any feasible LP when an appropriate pivot rule is used [23], but how efficient is the simplex method. Each monotone path to the sink  $t$  on the directed polyhedral graph  $G$  can be seen as a run of the simplex method. The length of the monotone path can be an indicator of the efficiency of the simplex method. A long monotone path means the simplex method needs to pass through many vertices to terminate, while a short monotone path is the contrary. Thus, we can learn more about the efficiency of the simplex method by investigating the behavior of the monotone paths on a polytope.

Various mathematicians contributed in studying the efficiency of the simplex method by constructing a bound on the number of pivot steps with the dimension  $d$  of the LP problem and the number of constraints  $n$ . In the paper published in 1972 [12], Klee and Minty discovered that the worst case of the number of steps required to solve the LP (using the common pivot rules) is exponential in terms of  $n$  and  $d$ . Luckily, in the paper published in 1982 [5], Borgwardt found that the average number of steps required by the simplex method is polynomial in terms of  $n$  and  $d$ . Yet, almost all common pivot rules have been shown to have cases that require exponential steps. The question of whether it is possible to have a pivot rule that always runs in polynomial times is still open [15].

Below are three important characteristics naturally arises from the 1-skeleton and the directed 1-skeleton:

DEFINITION 24. The **diameter** of a polyhedral graph  $G$  is the maximum distance over all pairs of vertices.

DEFINITION 25. The **monotone diameter** of a directed polyhedral graph  $DG$  is the maximum monotone distance between any two vertices in  $G$  that are connected by monotone paths.

DEFINITION 26. The **monotone height** of  $P$  with  $f$  is the length of the longest monotone path on  $P$  with objective function  $f$ .

The diameter of a polytope has long been a field of interest because it is an indicator of the efficiency of the simplex method. Warren Hirsch conjectured in 1957 that the diameter of a  $d$ -dimensional polytope with  $n$  facets cannot exceed  $n - d$  [20]. If the conjecture is true, the simplex method will be efficient, and the problem will be that we do not have good enough pivot rules to fully activate the potential of the simplex method. The monotone diameter version of the Hirsch conjecture, which states that the monotone diameter of a  $d$ -dimensional polytope with  $n$  facets cannot exceed  $n - d$ , was disproved by Michael Todd for  $d \geq 4$  in 1980 [21]. In 2010, the famous Hirsch conjecture was disproved by Francisco Santos using a 43-dimensional polytope with 86 facets, which has a diameter of at least 44 [20]. The disproof of the conjecture shows that there is no possible pivot rule that can always finish in  $n - d$  steps given any polytope. The polynomial Hirsch conjecture, which states the diameter of a polytope with  $n$  facets is  $O(n^k)$  for some constant  $k$ , remains open.

The monotone height of a polytope indicates the worse case scenario for the simplex method. In the 1960s, people were optimistic about the simplex method. In 1965, Klee asked a question, often called the Monotone Upper Bound Problem, how long can a monotone path be [11]. In 1972, Klee and Minty constructed the famous Klee-Minty cube that, using Dantzig's rule, the simplex method will go through all the vertices to find a path between two vertices, but these two vertices are connected by an edge [12]. A sample picture of the Klee-Minty cube is shown below as Figure 8. Since then, many mathematicians used similar ways to construct monotone paths with exponential lengths for other pivot rules. This lead to an open question that if there is a pivot rule that is efficient under any circumstances [15].

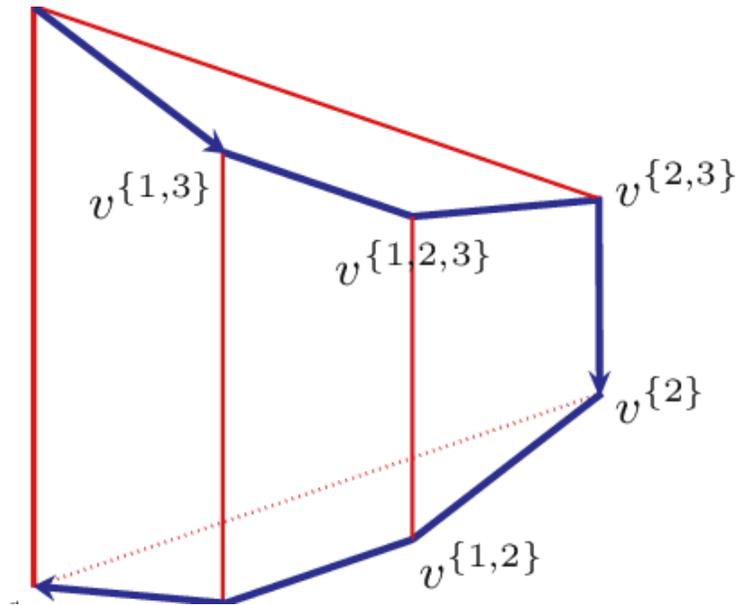


FIGURE 8. A picture of the Klee Minty cube [6]. There is a monotone path that goes through all the vertices.

### 1.6. Flips

In this section, we introduce an elegant idea of building a graph structure over the set of all monotone paths.

DEFINITION 27. Two monotone paths on  $P$  with objective function  $f$  **differ by a polygon flip** (or **flip**) across a 2-dimensional face  $F$  if they agree on all edges not lying on  $F$  but follow the two different monotone paths on  $F$ , from the unique source to the unique sink of  $F$  [2].

Figure 9 represents the directed 1-skeleton of a polytope called dodecahedron. The graph has a source (represented by  $s$ ) and a sink (represented by  $t$ ). On each of the 2-dimensional face  $F$  of the dodecahedron, there is a unique sink and a unique source in the subgraph of  $G$  on  $F$ .

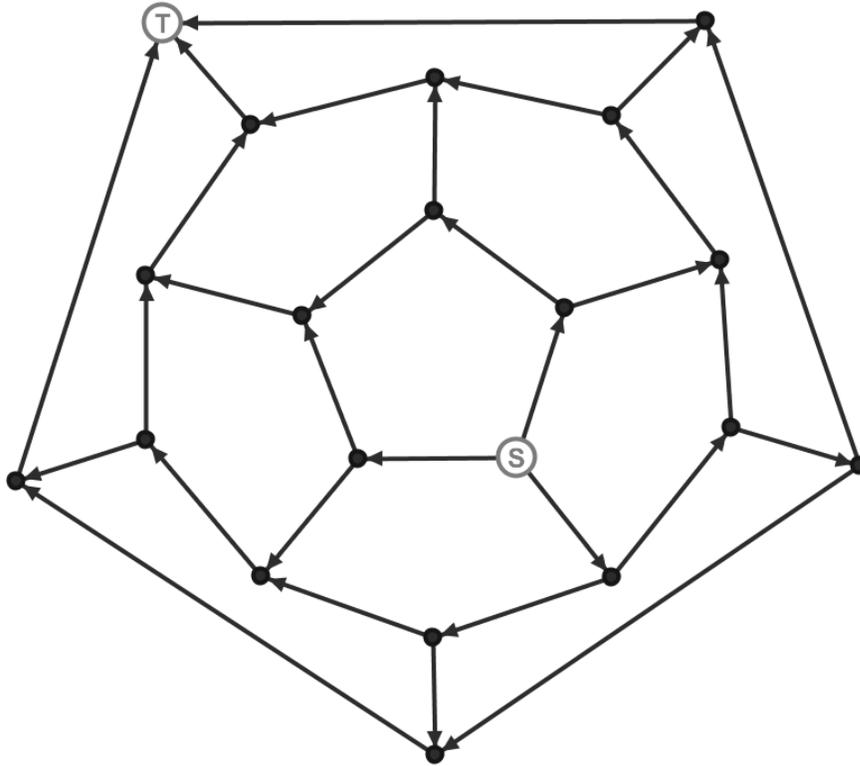


FIGURE 9. The directed 1-skeleton of a dodecahedron. Each face has a unique sink and source.

Figure 10 below shows two different  $s$ - $t$  monotone paths on the directed 1-skeleton of the dodecahedron. The paths are colored red, and the 2-dimensional face they differ is colored orange. These two  $s$ - $t$  paths agree on all other edges except on one 2-dimensional face, so they differ by one flip.

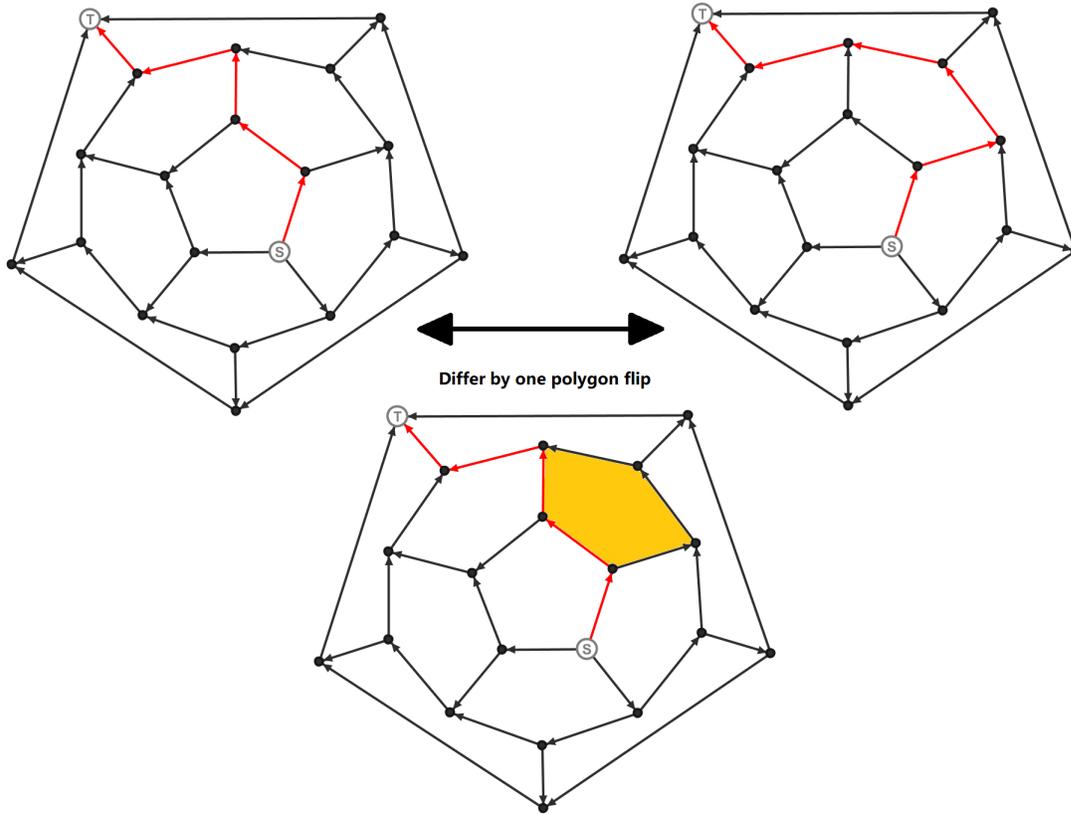


FIGURE 10. Two s-t monotone paths on the dodecahedron that differ by a flip.

DEFINITION 28. The graph of the monotone paths on  $P$  with objective function  $f$  is called the **flip graph** of  $P$  with  $f$ , and it is defined as the simple (undirected) graph where each node represents a unique monotone path on  $P$  with objective function  $f$  and each edge represents a pair of paths that differ by a polygon flip across a 2-dimensional face of  $P$  [2].

The flip graph of the dodecahedron is shown in Figure 11. Each node is an s-t monotone path and has been drawn out explicitly, and each edge (the blue line) connects two paths that differ by a flip.

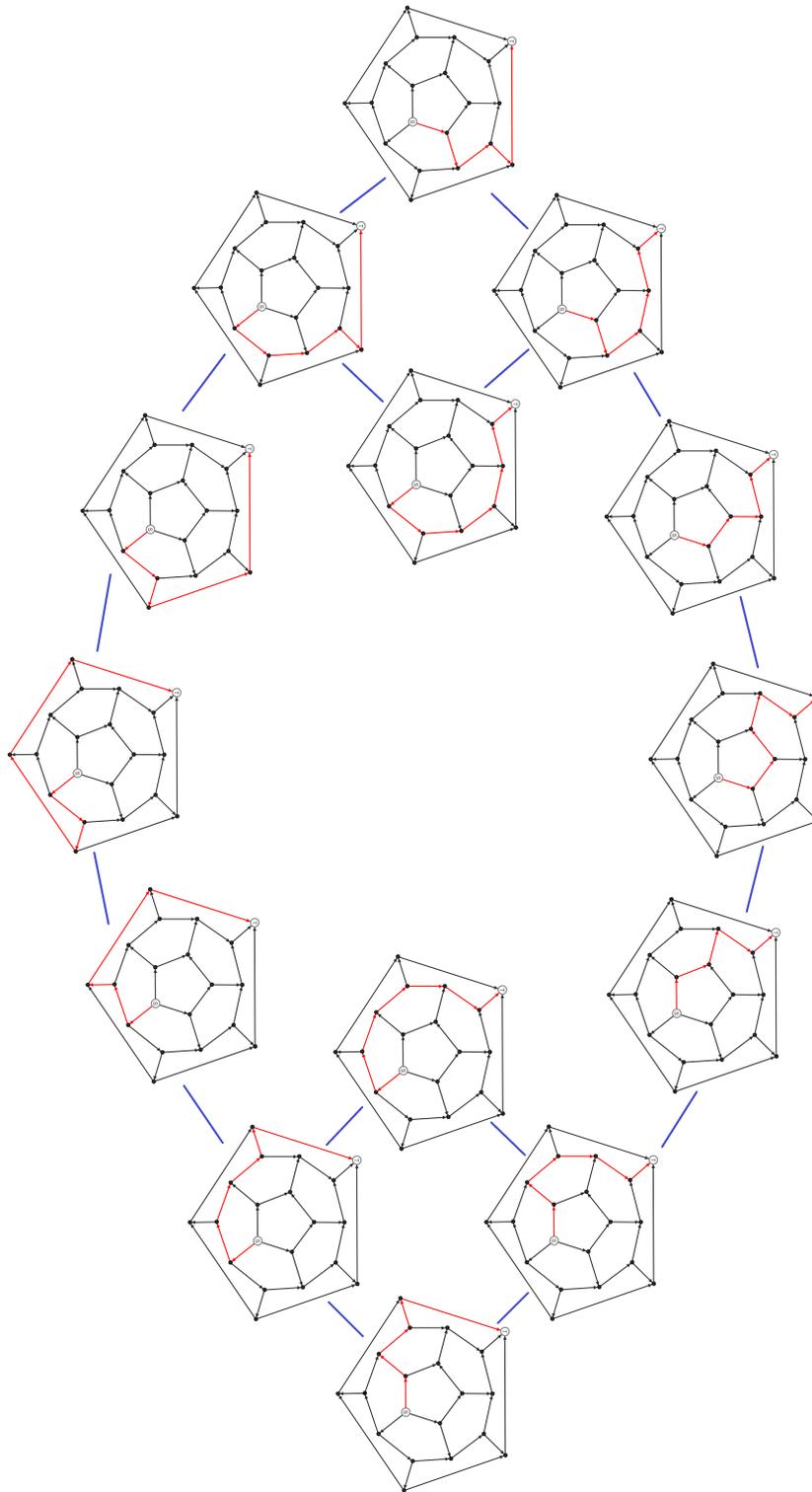


FIGURE 11. The flip graph of a dodecahedron. Each node is a unique monotone path drawn in red.

One might be interested in the connectivity of the flip graph. If the flip graph is connected, we can start from one monotone path and go to any other monotone paths by iteratively performing local polygon flips. In fact, in the paper published in 2000 [3], Athanasiadis et al. proved that, for any  $d$ -dimensional polytope with  $d \geq 3$ , the flip graph is 2-connected. They constructed  $d$ -dimensional polytopes for all  $d \geq 4$  with flip graphs that are 2-connected but not 3-connected. This shows that 2-connectedness is the best we can get for the flip graph of general polytopes. For  $d$ -dimensional simple polytopes, however, they showed that the flip graph is  $(d - 1)$ -connected. The diameter bound for the flip graphs of 3-dimensional polytopes has been studied in [2], and the bound of flip graph diameter for  $d \geq 4$  remains open.

## Main Algorithms

We developed *PolyPathLab*, a MATLAB-based package, that takes in polytopes in cdd/cdd+ format and computes the following features:

- the *diameter*
- the *monotone diameter*
- the *monotone height*
- the *number of monotone paths*
- the *number of directed arborescences*
- characteristics related to four famous pivot rules: Dantzig’s rule, greatest descent, steepest edge, and Bland’s rule
- the *flip graph* and the *diameter of the flip graph*

We will use the dodecahedron and a fixed objective function  $f$  for examples we use in this chapter. The dodecahedron has 20 vertices, 12 facets, and 30 edges. Each facet of the dodecahedron is a regular pentagon.

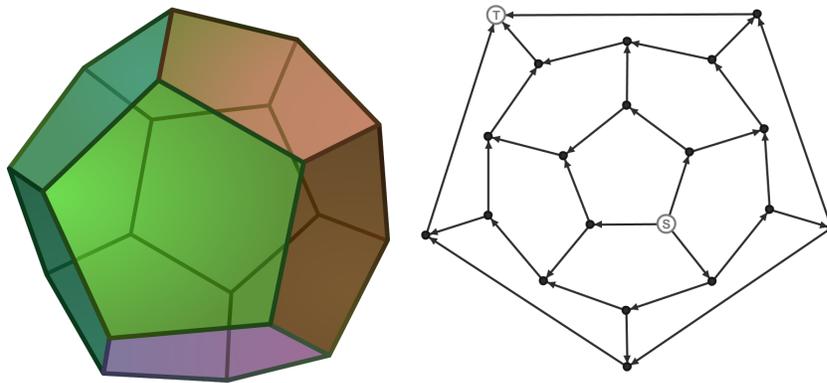


FIGURE 1. A picture of the dodecahedron [24] and the directed 1-skeleton of the dodecahedron.

Besides computation of the above features, *PolyPathLab* has two functions, “generate\_3d\_CuttingPlane.m” and “generate\_3d\_PointOnSphere.m,” to generate random  $3d$

polytopes, which we use to test some properties of  $3d$  polytopes. The functions are described in the next section.

### 2.1. Random 3-dimensional Polytope Generator

We have two ways to generate random  $3d$  polytopes: random cutting planes and random points on a sphere. The polytopes created by the cutting plane method are simple polytopes, while the points on sphere method may generate non-simple polytopes when the number of vertices is large.

For the cutting plane method, we first construct a random cuboid that is bigger than a  $2 \times 2 \times 2$  cube in any direction. We then generate cutting planes one to two unit lengths away from the center of the cuboid. This ensures that we get a polytope with moderate size which increases the possibility of having effective cutting planes (cutting planes that produces one of the facets of the polytope). If we do not have a limit on the distance, there might be some “deep” cuts that are very close to the center of the cuboid and make many other cuts ineffective.

For the points on sphere method, we first generate a sphere of radius two, then select points at random. The these points form a polytope.

Here is the code to generate a random point on a sphere of radius two:

```

1  %n is the number of vertices of the polytope
2  %set the radius to 2
3  r = 2;
4
5  %create random theta and phi
6  theta = 2 * pi * (2 * rand(n, 1) - ones(n, 1));
7  phi = pi * ( 2 * rand(n, 1) - ones(n, 1));
8
9  %obtain the coordinates of the vertices
10 x = r .* cos(theta) .* sin(phi);
11 y = r .* sin(theta) .* sin(phi);
12 z = r .* cos(phi);
13
14 vertices = [x,y,z];

```

This code is a usage of spherical coordinate system. If we have  $0 \leq \theta \leq 2\pi$  and  $0 \leq \phi \leq \pi$ , we can obtain all the possible points on the sphere by fixing the radius  $r$  to equal the radius of the sphere. If we randomize  $\theta$  and  $\phi$ , we can obtain all the possible points on the sphere, thus randomizing the coordinates of the vertices.

### 2.2. Diameter

The diameter of a polytope shows us how efficient is the simplex method on the polytope. There are two ways to get the diameter of a polytope. First, MATLAB has a function called “distances” which measures the distances between any two pair of vertices [17]. For a polytope  $P$  with  $n$  vertices, it outputs an  $n \times n$  matrix where each entry corresponds

to the distance between the vertices of the row and column of the entry. The maximum number of the matrix produced by the distance function is the diameter according to definition. Second, we can multiply the matrix of  $n \times n$  identity matrix  $I_n$  by adjacency matrix  $A$  plus an  $n \times n$  identity matrix  $I_n$ , or  $(A + I_n)$ . Repeat multiplying the current matrix by  $(A + I_n)$  until all of its entries become bigger than zero. Then, we record the number of times we have multiplied. The idea is that, every time the new matrix has an entry that turns from 0 to some positive number, the vertices represented by the row and column of the entry are connected by a path. For example, if  $A_{i,j}^k = 1$ , there is a path from  $i$  to  $j$  with length  $k$ . By adding  $I_n$  to  $A$  and raise  $(A + I_n)$  to the power  $k$ , we allows the vertices to have self-loops so that if there is a zero in the  $ij$  entry of the current matrix, it must be that there is no path from  $i$  to  $j$  with length  $k$ . If the matrix has only non-zero entries, all the vertices are connected to each other with paths that have lengths less than the number of times we multiply.

In our program, we implemented the second method, matrix multiplication, to keep it consistent with the methods we used for the monotone diameter and the monotone height. Here is the pseudocode that computes the diameter.

```

1  %A is the n-by-n adjacency matrix. I is the n-by-n identity matrix. R is the ...
   matrix that records the entries
2
3  %initialize R as the identity matrix
4
5  R = I;
6
7  diameter = 0;
8
9  %multiply until all the entries are greater than 1
10
11 while ismember(0, R) %if there is 0 in R
12
13     R = R * (A + I); %multiply R by (A + I)
14
15     diameter = diameter + 1; %record the increase in the diameter
16
17 end

```

For the dodecahedron, which has 20 vertices,  $A$  is the matrix shown in the next page:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

If we multiply the identity matrix by  $(A + I_{20})^4$ , we obtain a matrix that only has a few zeros; if we multiply the current matrix by  $(A + I_{20})$  again, we have a non-zero matrix. Thus, the diameter of the dodecahedron is five, meaning that any vertex is connected to any other vertices in a path with length less than five.

### 2.3. Monotone Diameter

Recall that the monotone diameter of  $DG$  is the maximum monotone distance between any two vertices in  $DG$  that are connected by monotone paths. The monotone diameter is not the same as the monotone distance between the source and the sink. Figure 2 is an example where the source  $s$  does not have a longer monotone distance to the sink  $t$  than some other vertex  $i$ . The monotone distance from  $s$  to  $t$  is two, but the monotone distance from  $i$  to  $t$  is four.

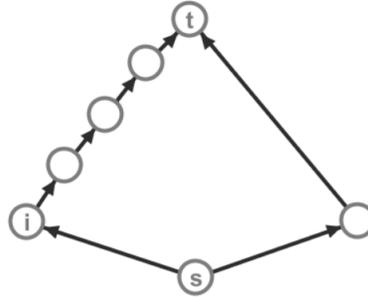


FIGURE 2. A directed graph where the monotone distance between the source  $s$  and the sink  $t$  is smaller than the monotone distance between  $i$  and  $t$ .

To get the monotone distance for vertex  $i$ , we multiply  $I_n$  by the matrix  $(D + I_n)$ , the directed adjacency matrix with diagonals being all 1. We repeat multiplying the current matrix by  $(D + I_n)$  until the column of the sink  $t$  has non-zero values on  $i$ -th row for the first time. The number of multiplications  $k_i$  means that there is at least one monotone path from  $i$  to  $t$  with length  $k_i$ . We take the maximum  $k_i$  among all vertices to get the monotone diameter. In other words, we multiply the starting matrix  $I_n$  again and again by  $(D + I_n)$  and record the number of times of multiplication  $k$  when the column of the sink has all non-zero values.

Here is the pseudocode for computing the monotone diameter:

```

1  %D is the directed adjacency matrix; I is the identity matrix; R is the current ...
   matrix; t is the index of the sink
2
3  %initialize R as the identity matrix
4
5  R = I;
6
7  monotone.diameter = 0;
8
9  %multiply until all the entries in column t are nonzero
10
11 while ismember(0, A2(:, t)) == 1 %if there is a zero in column t
12     R = R * (D + I);
13     mono.diameter = mono.diameter + 1;
14
15 end

```

For the same dodecahedron in Section 2.2 with an objective function  $f$ , the directed adjacency matrix is shown below:

$$D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$D$  is calculated from the adjacency matrix  $A$  and the objective function  $f$  by comparing the objective values of the two connected vertices. If  $i$  is connected with  $j$ ,  $A_{i,j} = A_{j,i} = 1$ . Then, we compare the objective value of  $i$  and that of  $j$ . If the objective value of  $i$  is less than that of  $j$ ,  $D_{i,j} = 1$  and  $D_{j,i} = 0$ .

The sink of this example is vertex 10. If we raise  $(D + I_{20})$  to the fourth power, there are zeros in the tenth column; if we raise  $(D + I_{20})$  to the fifth power, there is no zero in the tenth column, which means that there is at least one path from any vertex to the sink with the length less than five. Thus, the monotone diameter of this dodecahedron with the objective function  $f$  is five.

## 2.4. Monotone Height

Recall that the monotone height of  $DG$  is the length of the longest monotone path on  $DG$ . Equivalently, monotone height is the maximum length of the monotone path from the source  $s$  to the sink  $t$ . We can prove this equivalence by contradiction. Let  $i$  be a vertex that is not the source or the sink. Assume the maximum length of the monotone path from  $i$  to  $t$  is  $k$ , which is longer than the maximum length of the monotone path from  $s$  to  $t$ . There is a monotone path of length bigger than one from  $s$  to  $i$  because of the property of  $DG$  we shown in Section 1.3. Then, we can build a new monotone path from  $s$  to  $i$  to  $t$ , which has the length at least  $(k + 1)$ . This is a contradiction to our assumption.

To get the monotone height, we raise  $D$  to power  $k$ , where  $k$  is the *first time* that the matrix becomes a zero matrix. For all  $i$  and  $j$ ,  $D_{i,j}^k = 0$ , meaning that there is no monotone path with length  $k$  between any pair of vertices. Thus, the monotone height is equal to  $(k - 1)$  because the maximum length of the monotone paths will be  $(k - 1)$ .

```

1  %D is the directed adjacency matrix; R is the recording matrix; I is the identity ...
    matrix
2
3  R = I;
4
5  mono_height = -1; %starts from -1 because when the matrix becomes all zero, we ...
    have no monotone path.
6
7  %Multiply R by D until all the entries are zero
8
9  while ~ismember(0, R) %if the current matrix is all zero, stop
10
11     mono_height = mono_height + 1;
12
13     R = R * D;
14
15 end

```

Using the directed adjacency matrix  $D$  from the same dodecahedron and the same objective function  $f$  from the last section, we see that, for  $D^7$ , there are still ones in the matrix; for  $D^8$ , the matrix becomes a zero matrix for the first time. Thus, there is at least one path with length equals seven. Therefore, The monotone height of this dodecahedron with the same objective function  $f$  is seven.

## 2.5. Number of Monotone Paths

The number of monotone paths (from the source to the sink) tells us how many ways can the source  $s$  go to the sink  $t$ . The number of monotone paths is recorded when *PolyPathLab* computes the monotone height. Every time when the recording matrix is multiplied with the directed adjacency matrix, the entry  $st$  of the resulting matrix tells us how many monotone paths with the current length from  $s$  to  $t$ . When the matrix becomes all zero, we sum the values to get the number of total monotone paths.

For the dodecahedron and the same objective function, the recorded values in entry  $st$  are: 0, 0, 0, 0, 6, 6, 2. Summing these values, we get that the number of monotone paths is 14. There are only seven values because the monotone height is seven, so anything after the last value will be zero.

## 2.6. Number of Arborescences

Because the pivot rule picks one improving edge from every vertex of  $DG$  that is not the sink, it creates a directed tree, which is a subgraph of  $DG$ . Furthermore, there is a monotone path from every non-sink vertex to the sink, so the subgraph is an arborescence. We say two pivot rules are equivalent on  $DG$  if they produce the same arborescence. Then, the number of arborescence is the number of the equivalence groups of pivot rules, and we can estimate the number of pivot rules by computing the number of arborescences [2]. Figure 3 shows one arborescence of the dodecahedron with objective function  $f$ .

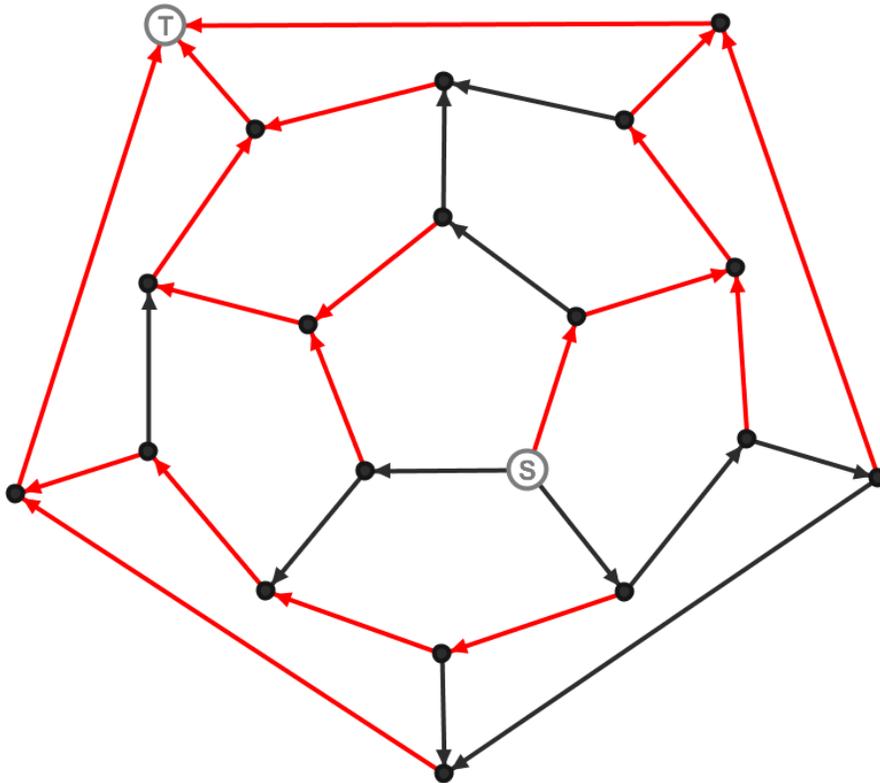


FIGURE 3. An arborescence of the dodecahedron. From each vertex, the pivot rule picks only one outgoing edge.

To compute the number of arborescences, we multiply the number of outgoing edges of each vertex except the sink (which has 0 outgoing edges) because changing the outgoing edge from a vertex will give us a different arborescence.

For the dodecahedron and the objective function  $f$ , if we count the outgoing edges of each vertex and put the numbers into a vector, we get:

$$[ 2 \ 2 \ 1 \ 2 \ 1 \ 2 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 2 \ 2 \ 1 \ 3 \ 2 \ 1 \ 2 \ 2 ]$$

If we multiply the nonzero entries, we get 1536, which is the number of arborescences of the dodecahedron with the objective function  $f$ .

### 2.7. Characteristics Related to Pivot Rules

In our program, we include four pivot rules: Dantzig's rule, Greatest Descent, Steepest Edge, and Bland's rule. Based on these four pivot rules, we collect characteristics related to the arborescences outputted by these pivot rules to see the performance of each pivot rule on the polytope  $P$  with the objective function  $f$ . The characteristics include: the average length of the monotone paths, the standard deviation of the length of the monotone paths of the arborescence, the standard deviation of the indegrees of the arborescence, the length of the monotone path between the source and the sink, the maximum length of the monotone paths of the arborescence, and the number of leaves in the arborescence. We did not include the average indegree of the arborescence for a polytope with  $n$  vertices because it is always  $\frac{n-1}{n}$ .

### 2.8. Flip Graph

One interesting combinatorial feature of a polytope with an objective function is the flip graph. The flip graph tells us how are the monotone paths related to each other. Since the flip graph is 2-connected, we are also interested in the diameter of the flip graph because the diameter tells us how many flips do the furthest pair of monotone paths differ.

To compute the flip graph, first, *PolyPathLab* uses a code called *getpaths* (see [1]) to obtain all the monotone paths. Then, to find out if two monotone paths,  $p_i$  and  $p_j$ , are connected by a flip, we find out the first vertex  $u$  and the last vertex  $v$  that the two monotone paths differ from each other. Collecting all the vertices in between, we obtain a set of vertices between  $u - 1$  and  $v + 1$  for  $p_i$  and  $p_j$ .

If  $p_i$  and  $p_j$  differ by a polygon flip, then this set of vertices belong to the same 2-dimensional face. In order to compare to the faces, We put the coordinates of the vertices into a matrix  $V$  with each row being a vertex in the set. Then, we compute  $AV^T$ , with  $A$  being the inequalities in the form  $Ax \leq b$ . If there is a zero in the entry  $ij$  of the resulting matrix, vertex  $j$  is on the facet represented by the  $i$ -th row of  $A$ ; if the  $i$ -th row is a zero vector, all the vertices between  $u - 1$  and  $v + 1$  are on the facet represented by the  $i$ -th row of  $A$ . For dimension  $d$ , the intersection of  $d - 2$  hyperplanes will provide us a 2-dimensional face. Therefore, if there are  $d - 2$  rows that are zero vectors, we know  $p_i$  and  $p_j$  differ by a polygon flip. We will show some examples for the flip graph in Section 3.5.

## PolyPathLab Software

To run *PolyPathLab*, users need to know how to use `cdd/cdd+`. The process of using *PolyPathLab* is:

- (1) Create an inequality file or a vertices file for `cdd/cdd+`.
- (2) Run `cdd/cdd+` to obtain the files needed for the input of *PolyPathLab*, including `*.ine`, `*.ead`, `*.ext` files. We will call these the polytope files.
- (3) Create an objective function file `*.txt` for the input of *PolyPathLab*.
- (4) Put the polytope files with the correct file-naming convention and the objective function file into the input folder under the package directory.
- (5) Run “`main_general.m`” or “`flip_graph.m`” based on what users want to compute and follow the instructions to set up the settings. Collect the outputs.

We explain each step in this Chapter, with step 5 being separated into two sections: computation stage of the features except the flip graph (`main_general.m`) and computation stage of the flip graph (`flip_graph.m`).

Caution: to reduce the impact of round-off errors during computation, we set the tolerance to be  $10^{-8}$  for the pivoting computation and the flip graph, meaning that any value with an absolute value less than  $10^{-8}$ , it will be seen as zero. If users run into problems such as the pivot rules do not terminate or the flip graph is not connected, users can try adjusting the input files or adjusting the tolerance.

### 3.1. Creating Input Files Using the Program `cdd/cdd+` (Step 1 and 2)

The program `cdd/cdd+` [7] uses Double Description method to generate the vertices, the edges, and the system of inequalities of a polytope, given the vertices or the inequalities. Please follow the instructions in the `cdd/cdd+` Reference Manual [7] for installation and for using the program `cdd/cdd+`. From now on, we assume users have installed the program `cdd/cdd+` and know how to run it.

We need the following three files from the program `cdd/cdd+`:

- the inequality file, which has an extension `*.ine`,
- the vertices file, which has an extension `*.ext`, and
- the adjacency relationship of vertices, which has an extension `*.ead` file.

Running the program `cdd/cdd+` using the vertices file as an input will generate the inequality file; running the program `cdd/cdd+` using the inequality file will generate the

vertices file and the adjacency relationship of vertices. Thus, if users have a list of the coordinates of the vertices, users will need to:

- (1) use the coordinates of the vertices to write an input file \*.ext,
- (2) run the program cdd/cdd+ on the \*.ext file to obtain the \*.ine file, and
- (3) run the program cdd/cdd+ on the \*.ine file to obtain the \*.ead file and the \*.ext file.

In this process, the \*.ext file will be replaced with a new one, but it will not cause any problem. If users have a list of the inequalities, users will need to:

- (1) use the inequalities to write an input file \*.ine,
- (2) run the program cdd/cdd+ on the \*.ine file to obtain the \*.ead file and the \*.ext file.

From our experience, if the input file has a rounding error, the program cdd/cdd+ may output some redundant vertices or inequalities. Thus, users might have to delete some of the vertices or inequalities from the output if they are redundant. Chapter 8 of the cdd/cdd+ Reference Manual [7] also provides some useful tips on reducing the error.

We will provide a simple explanation of how to construct an input file for the program cdd/cdd+ in this thesis; for a detailed explanation, please refer to the cdd/cdd+ manual [7].

Figure 1, 2, 3 below are the \*.ext, \*.ine, \*.ead files for tetrahedron, respectively. In each of the files, the line starts with an \* will be ignored by the program cdd/cdd+. In \*.ext and \*.ine file, the first line without \* tells the program cdd/cdd+ which file it is: H-representation for inequalities and V-representation for vertices. Then, we need to have a line of “begin” to tell the program cdd/cdd+ where to start. The next line tells the program cdd/cdd+ the parameters of the following matrix, including the number of rows, the number of columns, and what format is the entries of the matrix (rational or floating points). Then, we put in the matrix. A more detailed explanation on the matrix is in the caption of each figure. After that, we have a line of “end” to tell the program cdd/cdd+ finish reading. Finally, if it is in V-representation, we type “hull”; if it is in H-representation, we type “adjacency.”

```

* cdd+: Double Description Method in C++:Version 0.77 (August
19, 2003)␣
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch␣
* Compiled for Rational Exact Arithmetic with GMP␣
*Input File:tetrahedron.ine(4x4)␣
*HyperplaneOrder: LexMin␣
*Degeneracy preknowledge for computation: None (possible
degeneracy)␣
*Vertex/Ray enumeration is chosen.␣
*Output adjacency file is requested.␣
*Computation starts    at Wed Jul 24 00:04:58 2019␣
*
      terminates at Wed Jul 24 00:04:58 2019␣
*Total processor time = 0 seconds␣
*
      = 0h 0m 0s␣
*FINAL RESULT:␣
*Number of Vertices =4, Rays =0␣
V-representation␣
begin␣
4 4 rational␣
1 5917517/5000000 292893/500000 14226497/10000000␣
1 3632993/1000000 2 14226497/10000000␣
1 2 2 3732051/1000000␣
1 5917517/5000000 1707107/500000 14226497/10000000␣
end␣
hull␣

```

FIGURE 1. A screen shot of the \*.ext file for tetrahedron. The purple box tells the program cdd/cdd+ what this file is for, where V-representation tells cdd/cdd+ this file is the file for vertices. The green box is the parameter of the matrix below, including the number of rows, the number of columns, and the format of the entries. The red box is the coordinates and the blue box is an 1-vector (for extreme points).

```

* cdd+: Double Description Method in C++:Version 0.77 (August
19, 2003)␣
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch␣
* Compiled for Rational Exact Arithmetic with GMP␣
*Input File:tetrahedron.ext(4x4)␣
*HyperplaneOrder: LexMin␣
*Degeneracy preknowledge for computation: None (possible
degeneracy)␣
*Hull computation is chosen.␣
*Output adjacency file is requested.␣
*Computation starts    at Wed Jul 24 00:04:42 2019␣
*          terminates at Wed Jul 24 00:04:42 2019␣
*Total processor time = 0 seconds␣
*          = 0h 0m 0s␣
*Since hull computation is chosen, the output is a minimal
inequality system␣
*FINAL RESULT:␣
*Number of Facets = 4␣
H-representation␣
begin␣
4 4 rational␣
 4410748107341085267/384900260417000000 -23094013/16329930
-70710680832206/28867519531275 -1␣
-14226497/10000000 0 0 1␣
 1918535388870295801/1154700781251000000 -23094013/16329930
70710680832206/28867519531275 -1␣
-7857978237367/4082483000000 23094013/8164966 0 -1␣
end␣
adjacency␣

```

FIGURE 2. A screen shot of the \*.ine file for tetrahedron. The matrix in the middle is in the form of  $[b \ -A]$ . H-representation tells the program cdd/cdd+ that this is a file for the facets. The “adjacency” in the end tells the program cdd/cdd+ to output the adjacency relationship of the vertices.

```

* cdd+: Double Description Method in C++:Version 0.77(August
19, 2003)↵
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch↵
* Compiled for Rational Exact Arithmetic with GMP↵
*Adjacency List of output (=vertices/rays)↵
*cdd input file : tetrahedron.ine (4 x 4)↵
*cdd output file: tetrahedron.ext↵
begin↵
  4↵
  1 3 : 2 3 4↵
  2 3 : 1 3 4↵
  3 3 : 1 2 4↵
  4 3 : 1 2 3↵
end↵

```

FIGURE 3. A screen shot of the \*.ead file for tetrahedron. Each line has the format: the index of the vertex, how many neighbors for that vertex: the indices of the adjacent vertices, separated by spaces.

### 3.2. Creating the Objective Function File (Step 3)

The objective function file has a file extension of \*.txt or \*.csv. Users need to write the coefficients of the objective function into a file, and each coefficient is separated by a comma. Users can enter more than one objective function using “enter” as separator. Users can also create a matrix in the MATLAB and use the command “writematrix” to produce the objective function file.

*PolyPathLab* accepts objective functions that have more entries than the  $d$ -dimensional LP by using only the first  $d$  entries of the objective functions, but *PolyPathLab* does not accept objective functions that have less entries than  $d$ . An example of an objective function file corresponding to a 3-dimensional LP problem is show below as Figure 4.

```

0.311481398313174,0.660154347409728,-0.175542215364204
-0.928576642851621,0.392720093300886,-0.192312039912728
0.698258611737554,-0.332925212101578,-0.693654425709147
0.867986495515101,0.160464983609381,0.750132750249391
0.357470309715547,-0.42432359246051,0.754671133495374
0.515480261156667,-0.472058678313153,-0.814310610055575
0.486264936249832,-0.480197469024457,-0.334815155704933
-0.215545960931664,0.354160071149703,0.386983732737892
0.310955780355113,0.0396530984337071,-0.658208464835439
-0.657626624376876,-0.846492530827493,-0.308155008361366
0.412092176039218,-0.888309489930916,-0.269858065893958
-0.936334307245159,-0.482543361113537,-0.316679343376894
-0.44615403007822,-0.120145162343774,0.830775424427847
-0.907657218737692,-0.431421767199785,-0.657572512231135
-0.805736437528305,0.357535699851115,-0.613253177874528
0.646915656654585,0.899159095344755,0.932423258086316
0.389657245951634,0.547910178629617,-0.0779210913302946
-0.365801039878279,0.272278017291587,0.469756564218262
0.90044409767671,0.507162940395124,-0.826409304606432
-0.931107838994182,0.493633208760915,-0.569329077567315
-0.122511280687204,0.172062307317267,0.141139824477743
-0.236883085813983,0.546112671415381,-0.352867836632007
0.531033576298005,-0.21502348345947,0.914515682853362
0.590399802274126,0.210580238145938,0.180786996638068
-0.626254790891243,-0.505134252384825,-0.759716375903009

```

**This is an objective function file with 25 objective functions.**

**Each objective function is separated by "\n" and has 3 coefficients that are separated by commas.**

FIGURE 4. A screen shot of the objective function file. The coefficients are separated by commas, and the different objective functions are separated by lines.

### 3.3. What to Put in the Input Folder and the File-naming Convention (Step 4)

Users need to put the following files into the “input” folder under the directory before starting the computation:

- \*.ine file for the polytope  $P$ ,
- \*.ext file for the polytope  $P$ ,
- \*.ead file for the polytope  $P$ , and
- \*.txt or \*.csv file for the objective functions.

The file-naming convention for the three polytope files is:

[filename][number].[file extension].

[filename] is what users prefer to call the polytopes; [number] is a number used by *PolyPathLab* to keep track of multiple polytopes as inputs; [file extension] is the file extension, including \*.ine, \*.ead, and \*.ext. For example, if users want to perform calculations on 100 different polytopes, users can call the polytope files “3d\_polytope1.ead” “3d\_polytope2.ead” ... “3d\_polytope100.ead”, “3d\_polytope1.ine” “3d\_polytope2.ine” ... “3d\_polytope100.ine”, and “3d\_polytope1.ext” “3d\_polytope2.ext” ... “3d\_polytope100.ext”.

There is no specific naming convention for the objective function file. For example, the file can be called “obj.txt”.

After putting these input files into the folder “input”, users can run the two computation functions in the package *PolyPathLab*: `main_general.m` and `flip_graph.m`.

### 3.4. Computation Stage for the Features Except the Flip Graph (Step 5)

After putting the files mentioned in the previous section into the input folder, users can now run the `main_general.m`. This function computes all the characteristics mentioned in Chapter 3 *except* the flip graph. Then, it asks users a sequence of questions in the command window, and these questions will adjust the settings of the calculation as well as the output of the program. After answering these questions, users just need to wait until the program finishes running.

Figure 5 below shows what the program will ask and what users need to answer in the command window in an sample run of the program. The blue boxes represent the sample answers to the questions, and anything else is what the program asks. The empty blue boxes represent a direct “Enter” as an answer.

```

>> main_general
Please put your .ine, .ead, .ext files under the folder called 'input'

Do you want to apply this program to multiple LPs?
If YES, type 1. Anything that is not a "1" will be a NO:
1
Make sure your files are in the following form: filename+number.extention

For example: if you want to test 100 different spindles, you can call them:
spindle1, spindle2, ... spindle100.ine/ead/ext

What's the input polytope filename, WITHOUT .ine, .ead, .ext?
In the example above, you will type "spindle":
3d_polytope
How many different polytopes are you going to use?
In the example above, you will type "100":
100
Objective function's file's name, INCLUDING suffix like .txt or .csv
(should be in a matrix form that can be loaded by MATLAB):
obj.txt
How many objective functions do you want to use?
This number should be an integer, and not bigger than
the number of rows in your objective function's file:
500
Output file's name, without .csv:
3d_polytope_data
Which pivot rules do you want to use? Type all the numbers.
1 for Dantzig, 2 for Greatest Descent, 3 for Steepest Edge
4 for Bland from smallest index, 5 for Bland from biggest index,
6 for Bland from random order.123456 means all of them:
123456
Do you want the directed graphs? See figure 8 for example
If YES, type 1. Anything that is not a "1" will be seen as a NO:
Do you want the pivot arborescence diagram? See figure 9 for example
If YES, type 1. Anything that is not a "1" will be seen as a NO:
Do you want more explanations and labels on the output matrix? See figure 7
If YES, type 1. Anything that is not a "1" will be seen as a NO: for example
Do you want more information, including the directed 1-skeleton of the polytope,
how far the vertex is from the optimum, how many indegrees does the vertex have,
and which vertex it will go to next? See figure 10 and 11 for example
If YES, type 1. Anything that is not a "1" will be seen as a NO:

```

FIGURE 5. The command window of a sample run of the program.

The outputs will be in the “output” folder. The default output \*.csv file only includes the matrix of the results for the ease of further use. The following figure is a sample matrix output:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	12	20	5	7	5	8.5726	0.38197	14	1536	3.1757	2.5	1.2042	1.3571	0.63332
2	12	20	5	7	5	8.5726	0.38197	14	1536	4.232	2.5	1.2042	1.3571	0.63332
3	12	20	5	7	5	8.5726	0.38197	14	1536	2.9926	2.5	1.2042	1.3571	0.63332
4	12	20	5	7	5	8.5726	0.38197	14	1536	3.5802	2.6	1.241	1.3571	0.63332
5	12	20	5	7	5	8.5726	0.38197	14	1536	3.5276	2.5	1.2042	1.3571	0.63332
6	12	20	5	7	5	8.5726	0.38197	14	1536	2.6388	2.5	1.2042	1.3571	0.63332
7	12	20	5	7	5	8.5726	0.38197	14	1536	3.339	2.5	1.2042	1.3571	0.63332
8	12	20	5	7	5	8.5726	0.38197	14	1536	1.8136	2.5	1.2042	1.3571	0.63332
9	12	20	5	7	5	8.5726	0.38197	14	1536	4.2757	2.5	1.2042	1.3571	0.63332
10	12	20	5	7	5	8.5726	0.38197	14	1536	3.7782	2.5	1.2042	1.3571	0.63332
11														
12														
13														

FIGURE 6. The output matrix in \*.csv format generated by *PolyPathLab* by default.

Below are the additional outputs controlled by the settings. Refer to Figure 5 to see which question and information does each figure corresponds to:

	A	B	C	D	E	F	G	H	I	J	K			
1	objective function number1													
2	number o	num of v	diameter	mono_he	mono_dia	gamma	delta	num_of_p	num_of_a	min_max_obj	difference			
3	12	20	5	7	5	8.5726	0.38197	14	1536	3.1757				
4	Dantzig													
5	avg_length	std_length	avg_inD	std_inD	min_to_m	depth	breadth							
6	2.5	1.2042	1.3571	0.63332	5	5	6							
7	Greatest Descent													
8	avg_length	std_length	avg_inD	std_inD	min_to_m	depth	breadth							
9	2.5	1.2042	1.3571	0.63332	5	5	6							
10	Steepest Edge													
11	avg_length	std_length	avg_inD	std_inD	min_to_m	depth	breadth							
12	2.5	1.2042	1.3571	0.63332	5	5	6							
13	Blands from smallest index													
14	avg_length	std_length	avg_inD	std_inD	min_to_m	depth	breadth							
15	2.9	1.5133	1.4615	0.66023	6	6	7							
16	Blands from biggest index													
17	avg_length	std_length	avg_inD	std_inD	min_to_m	depth	breadth							
18	3	1.7607	1.5833	0.66856	7	7	8							
19	Blands from random index order													
20	avg_length	std_length	avg_inD	std_inD	min_to_m	depth	breadth							
21	2.75	1.337	1.3571	0.63332	5	5	6							
22	objective function number2													
23	number o	num of v	diameter	mono_he	mono_dia	gamma	delta	num_of_p	num_of_a	min_max_obj	difference			
24	12	20	5	7	5	8.5726	0.38197	14	1536	4.232				
25	Dantzig													

FIGURE 7. The output matrix in \*.csv format with explanations generated by *PolyPathLab*.

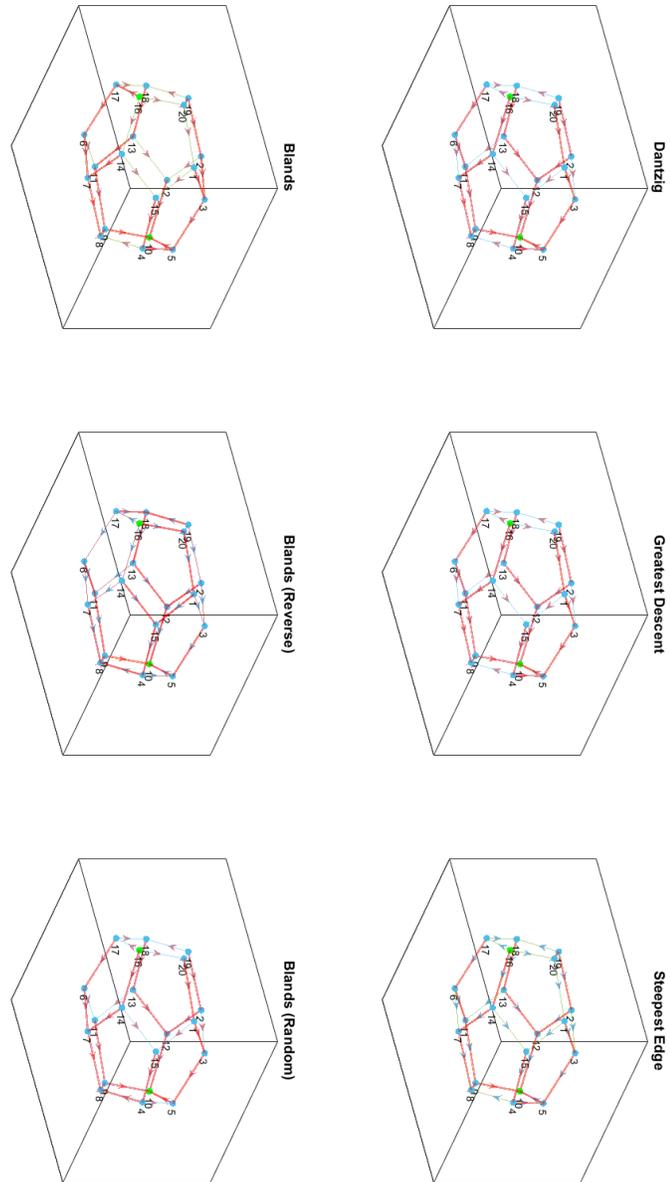


FIGURE 8. The directed graphs of  $P$  with  $f$  in \*.fig format (MATLAB figure) under different pivot rules. The red edges represent the pivot-rule-selected path.

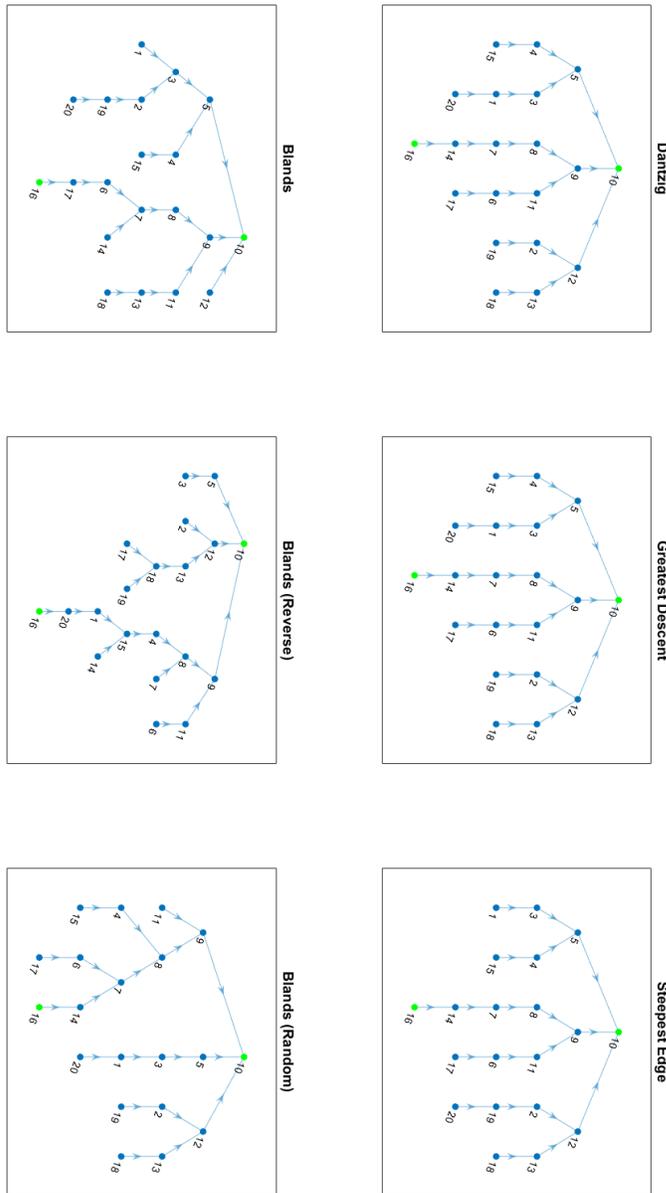


FIGURE 9. The arborescences of  $P$  with  $f$  in \*.fig format (MATLAB figure) under different pivot rules. The red edges represent the pivot-rule-selected path.

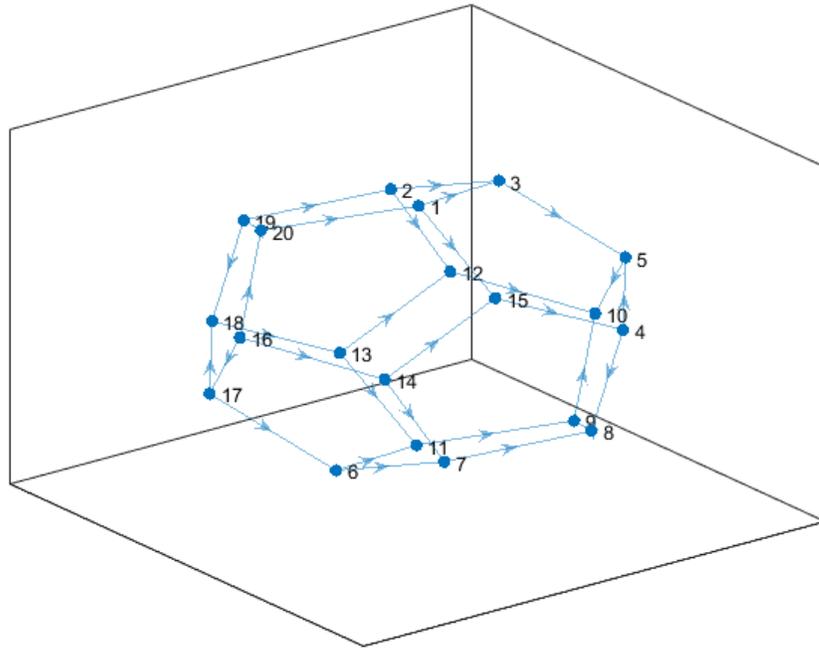


FIGURE 10. The directed 1-skeleton in \*.fig format (MATLAB figure) generated by *PolyPathLab*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	objective function number1																			
2	extra info on vertices																			
3	Dantzig																			
4	length to optimum																			
5	3	2	2	2	1	3	3	2	1	0	2	1	2	4	3	5	4	3	3	4
6	indegrees																			
7	1	1	1	1	2	1	1	1	2	3	1	2	1	1	0	0	0	0	0	0
8	which vertex is next																			
9	3	12	5	5	10	11	8	9	10	0	9	10	12	7	4	14	6	13	2	1
10	Greatest Descent																			
11	length to optimum																			
12	3	2	2	2	1	3	3	2	1	0	2	1	2	4	3	5	4	3	3	4
13	indegrees																			
14	1	1	1	1	2	1	1	1	2	3	1	2	1	1	0	0	0	0	0	0
15	which vertex is next																			
16	3	12	5	5	10	11	8	9	10	0	9	10	12	7	4	14	6	13	2	1
17	Steepest Edge																			
18	length to optimum																			
19	3	2	2	2	1	3	3	2	1	0	2	1	2	4	3	5	4	3	3	4
20	indegrees																			
21	0	1	1	1	2	1	1	1	2	3	1	2	1	1	0	0	0	0	1	0
22	which vertex is next																			
23	3	12	5	5	10	11	8	9	10	0	9	10	12	7	4	14	6	13	2	19

FIGURE 11. The extra information in \*.csv format generated by *PolyPathLab*.

### 3.5. Computation Stage for the Flip Graph (Step 5)

The flip graph is computed separately. The function `flip_graph.m` takes in *one* polytope and computes the flip graph diameter based on the given objective function. Similar to `main_general.m`, `flip_graph.m` asks users for the file name of the polytope file, the objective function file name, and how many objective functions users want to use. The output is displayed in MATLAB instead of in an output file. Depending on the number of objective functions, the output will be different.

If users want to investigate a polytope with one objective function, the output of the program will be a figure, which is the flip graph, and some text details in the command window:

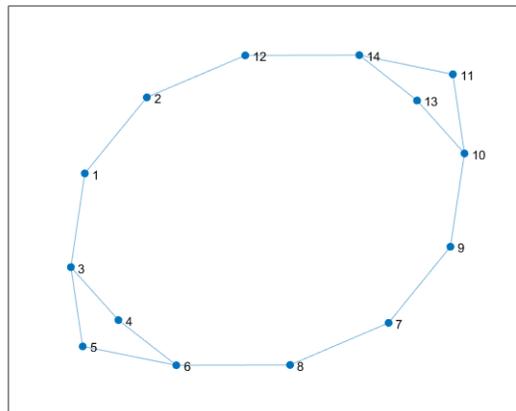


FIGURE 12. The flip graph of dodecahedron.

```
The flip diameter is: 6.
One pair of two paths that have the furthest flip diameter is:
16 17 6 11 9 10
16 20 1 3 5 10
The shortest flip path between the pair of paths is:
(These numbers each represents a path and corresponds to a vertex in the figure)
9 7 8 6 4 3 1
```

FIGURE 13. The text output of dodecahedron.

If users want to investigate a polytope under different objective functions, the output of the program will be the maximum, the minimum, and the average flip diameter, shown below:

```
The maximum flip diameter is: 6
The minimum flip diameter is: 6
The average flip diameter is: 6
```

FIGURE 14. A sample output with multiple objective functions.

The above example uses the dodecahedron as the input polytope, and the flip diameter of a dodecahedron under any objective function turns out to be six. In general, the minimum, the average, and the maximum flip diameter is not the same. For example, the following output comes from a random  $3d$  simple polytope with 31 faces with 100 different random objective functions:

```
The maximum flip diameter is: 16
The minimum flip diameter is: 15
The average flip diameter is: 15.16
```

FIGURE 15. Output from a different polytope.

## Computational Results

### 4.1. Choice of 3-dimensional Polytopes

We are interested in exploring the characteristics of  $3d$  polytopes. We studied famous polytopes such as the Platonic solids and the Archimedean solids as well as random  $3d$  simple polytopes. For the Platonic solids and the Archimedean solids, we took the vertices from *Wikipedia* [24] and *Visual Polyhedra* [19]. Besides that, we also computed the distribution of the monotone height and the monotone diameter by generating random polytopes using the random cutting plane method mentioned in Section 2.1. Then, we experimented on two types of famous polytopes, the Birkhoff Polytope and the traveling salesman polytope. We will present our computational results here.

### 4.2. Characteristics of Platonic Solids and Archimedean Solids

Platonic solids [25] are regular  $3d$  polytopes which are constructed by congruent, regular polygonal faces, with same number of faces meet at each vertex. There are 5 Platonic solids: the tetrahedron, the cube, the octahedron, the dodecahedron, and the icosahedron.

Archimedean solids [24] have regular polygonal faces and identical vertices (each vertex are surrounded by the same type of faces), and there are 13 Archimedean solids (from least number of facets to the most): the truncated tetrahedron, the cuboctahedron, the truncated cube, the truncated octahedron, the rhombicuboctahedron, the truncated cuboctahedron, the snub cube, the icosidodecahedron, the truncated dodecahedron, the truncated icosahedron, the rhombicosidodecahedron, the truncated icosidodecahedron, and the snub dodecahedron.

We ran experiments on all five Platonic solids and all 13 Archimedean solids, and we collected all the characteristics in the table below:

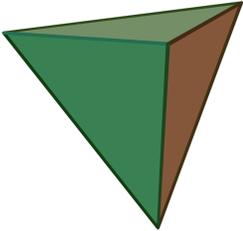
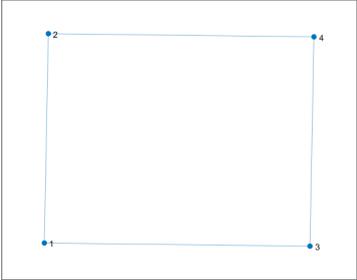
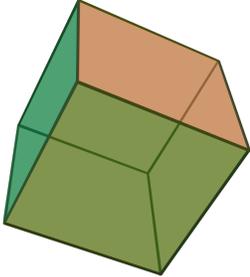
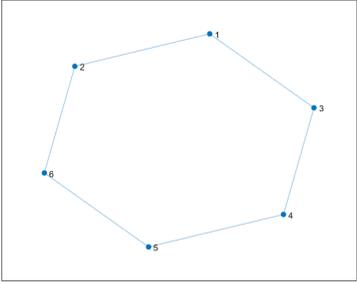
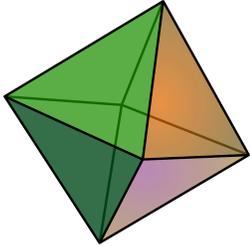
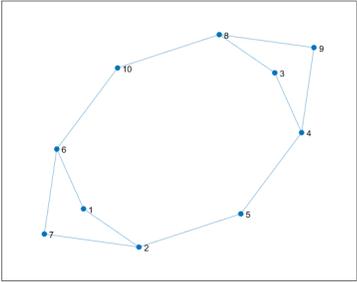
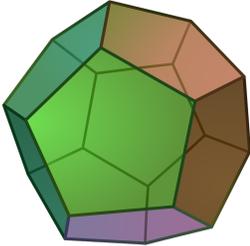
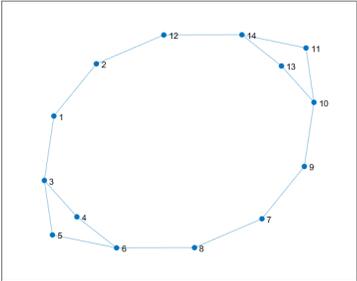
Polytope	# of facets	# of edges	# of vertices	diameter	monotone diameter	monotone height	# of paths	# of arborescences	flip diameter
Tetrahedron	4	6	4	1	1	3	4	6	2
Cube	6	12	8	3	3	3	6	24	3
Octahedron	8	12	6	2	2	4	10	48	4
Dodecahedron	12	30	20	5	5	7	14	1536	6
Icosahedron	20	30	12	3	3	7	62	25920	12
Truncated Tetrahedron	8	18	12	3	3	5	9	96	4
Cuboctahedron	14	24	12	3	3	5	24	2304	7
Truncated Cube	14	36	24	6	6	8	20	6144	7
Truncated Octahedron	14	36	24	6	6	6	16	6144	7
Rhombicuboctahedron	26	48	24	5	5	7	76	5.31e+6	13
Truncated Cuboctahedron	26	72	48	9	9	9	42	2.52e+7	13
Snub Cube	38	60	24	4	4	9,10,11	varies, 274-476	8.06e+8/9.07e+8/ 1.08e+9/1.21e+9	varies, 19-22
Icosidodecahedron	32	60	30	5	5	11	192	3.40e+8	18
Truncated Dodecahedron	32	90	60	10	10	14	108	1.61e+9	16
Truncated Icosahedron	32	90	60	9	9	11	62	1.61e+9	16
Rhombicosidodecahedron	62	120	60	8	8	12	892	6.49e+16	33
Truncated Icosidodecahedron	62	180	120	15	15	15	286	1.73e+18	31
Snub Dodecahedron	92	150	60	7	7	15,16,17, 18,19,20	varies, 8672-20316	varies, 3.64e+22 to 8.19e+22	varies, too many paths for computation

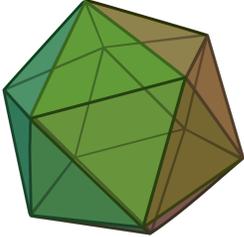
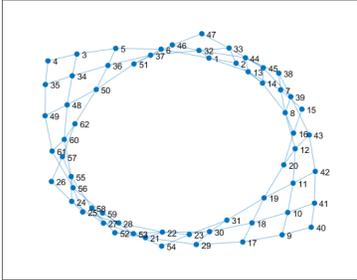
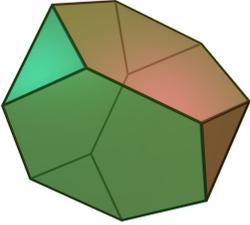
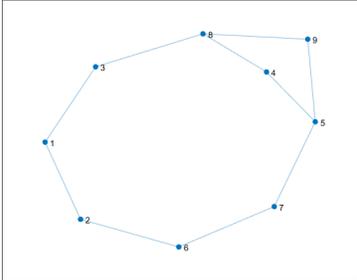
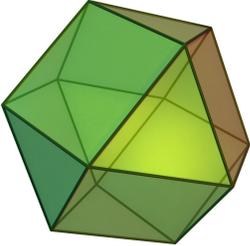
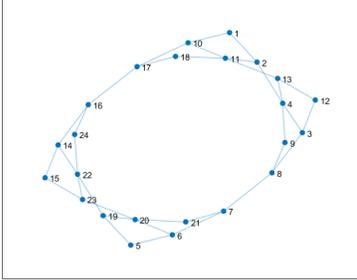
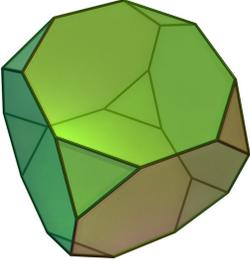
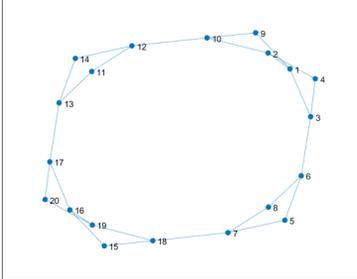
This table is given by applying 500 different random objective functions on each polytope. Most of the polytopes have the same numbers for all these objective functions, except the snub cube and the snub dodecahedron. The reason is that these polytopes are highly symmetrical, and applying different objective functions have similar effects. On the other hand, the snub cube and the snub dodecahedron have different monotone height, number of the monotone paths, number of arborescences, and flip diameter.

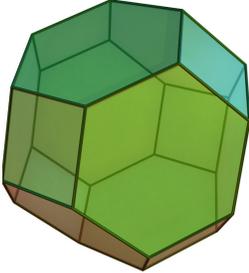
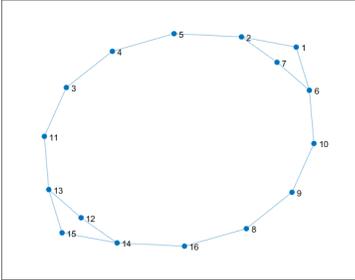
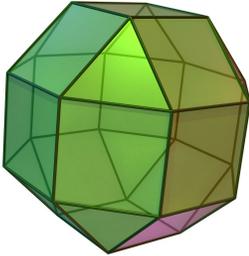
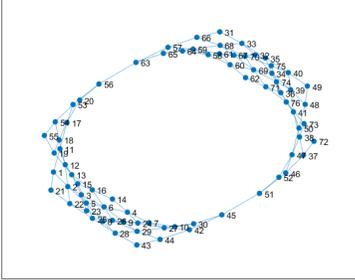
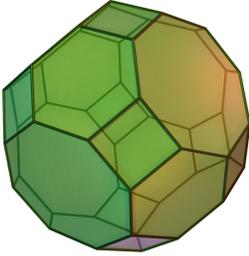
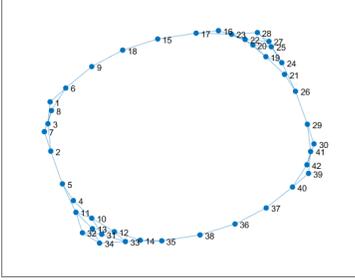
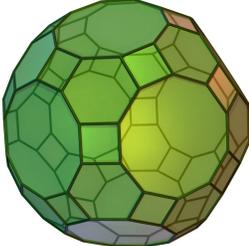
We can see that for all 18 polytopes, the monotone diameters equal diameters, but the monotone heights are greater than or equal to the diameters. Also, the flip diameters are at least a half of the number of facets for all of these polytopes.

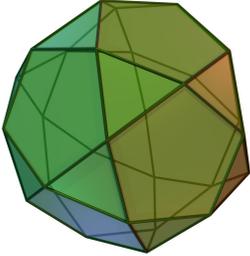
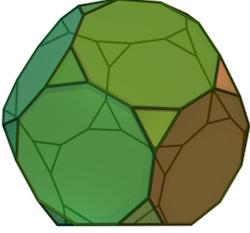
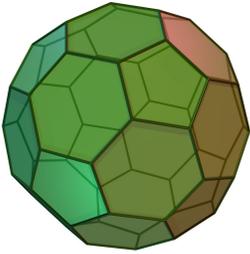
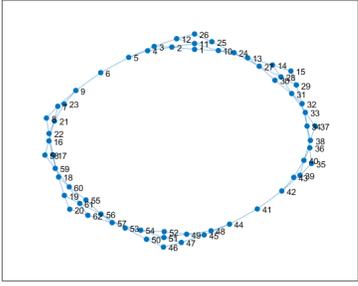
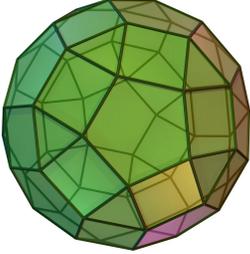
Below are 16 of them, excluding the snub cube and the snub dodecahedron. These 16 polytopes are highly symmetric, so their characteristics and the general shape of their flip graphs do not depend on the objective function.

The pictures of the Platonic solids [25] and the Archimedean solids [24] are from Wikipedia, and the flip graph is obtained by applying the “graph” function in MATLAB on the adjacency matrix of the flip graph:

Polytope	Picture (from Wikipedia)	Flip Graph
Tetrahedron		
Cube		
Octahedron		
Dodecahedron		

Polytope	Picture (from Wikipedia)	Flip Graph
Icosahedron		
Truncated Tetrahedron		
Cuboctahedron		
Truncated Cube		

Polytope	Picture (from Wikipedia)	Flip Graph
Truncated Octahedron		
Rhombicuboctahedron		
Truncated Cuboctahedron		
Truncated Icosidodecahedron		<p data-bbox="1040 1503 1198 1535">See Figure 1</p>

Polytope	Picture (from Wikipedia)	Flip Graph
Icosidodecahedron		See Figure 2
Truncated Dodecahedron		See Figure 3
Truncated Icosahedron		
Rhombicosidodecahedron		See Figure 4

Because several Archimedean solids have dense flip graphs and are hard to see, we put the magnified flip graphs below:

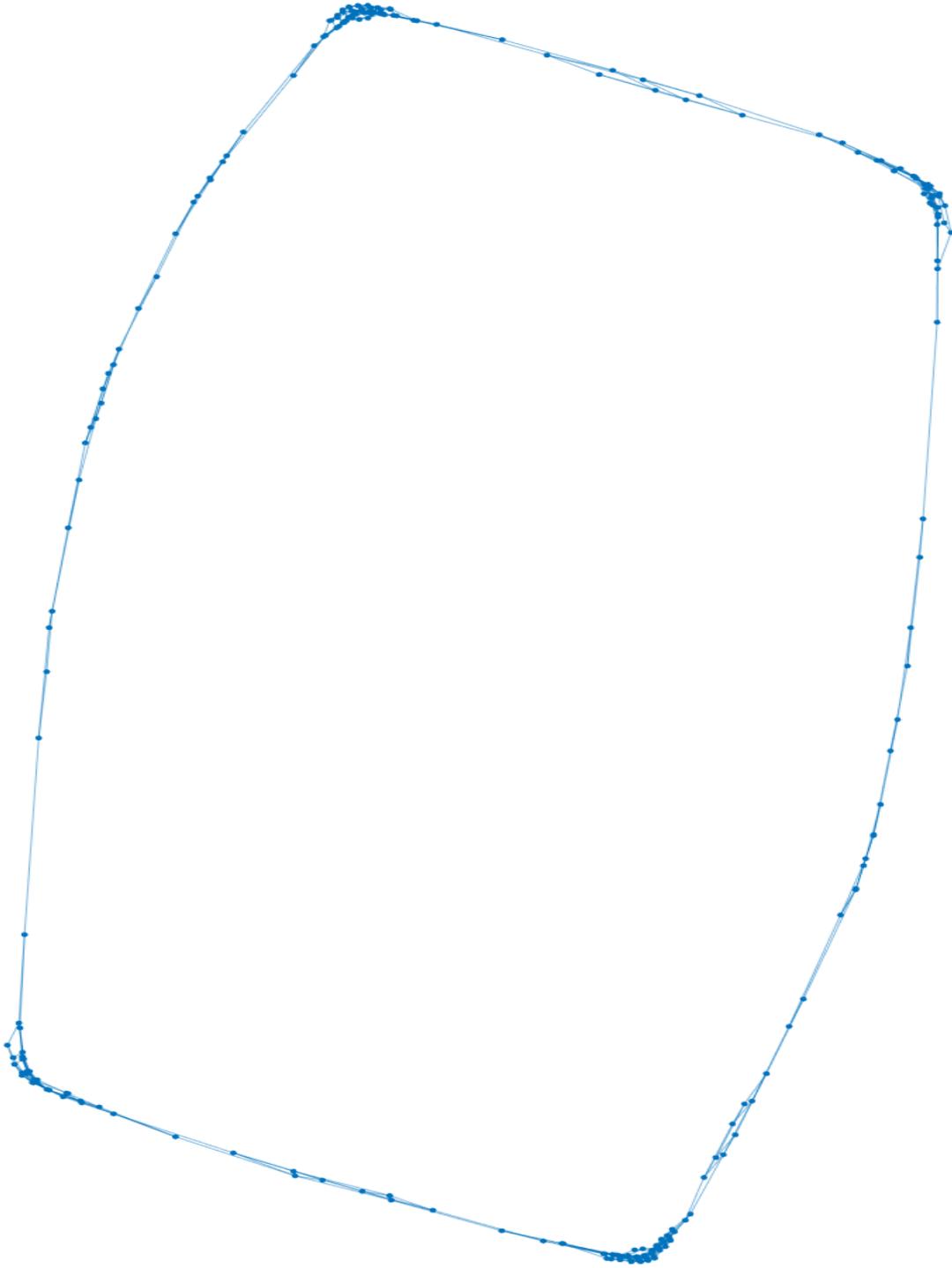


FIGURE 1. The flip graph of truncated icosidodecahedron.

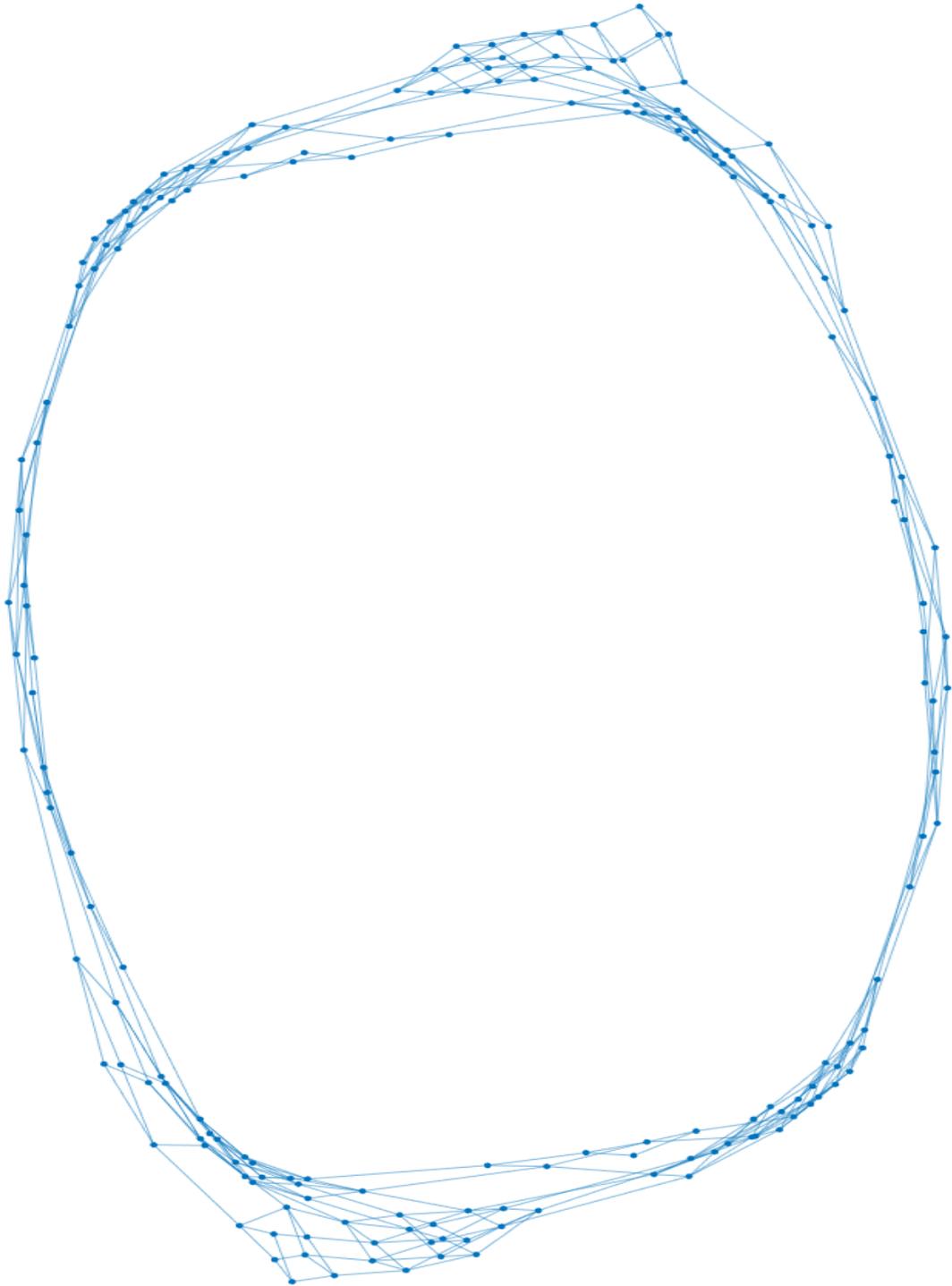


FIGURE 2. The flip graph of icosidodecahedron.

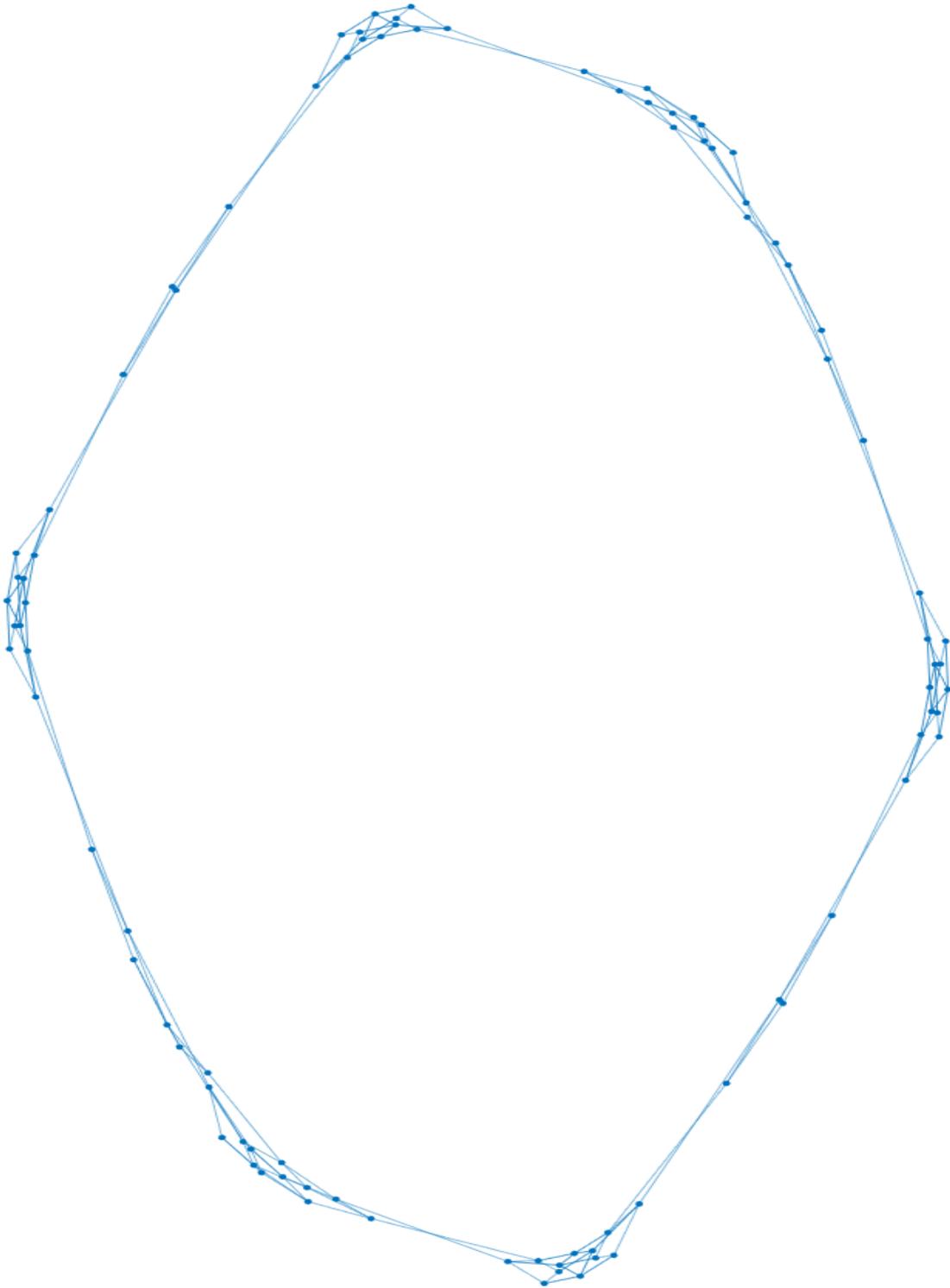


FIGURE 3. The flip graph of truncated dodecahedron.

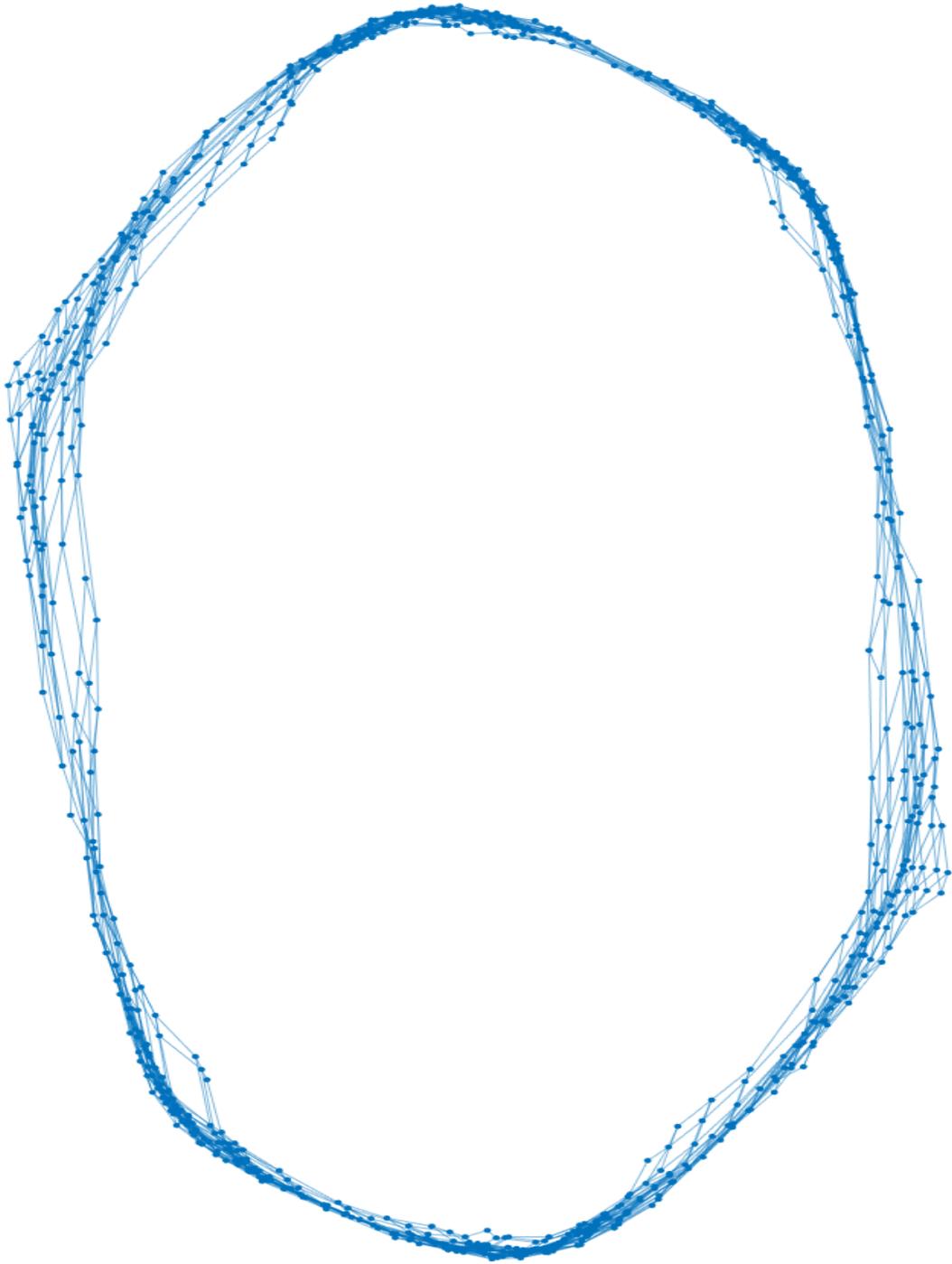


FIGURE 4. The flip graph of rhombicosidodecahedron.

### 4.3. The Special Cases of Archimedean Solids: the Snub Cube and the Snub Dodecahedron

The snub cube and the snub dodecahedron are two special Archimedean solids that, given different objective functions, they will have different characteristics, including the monotone height, the number of monotone paths, the number of directed arborescences, the flip graph, and the flip diameter.

For the snub cube, Figure 6 shows the distribution of the monotone height; Figure 7 shows the distribution of the number of monotone paths; Figure 8 shows the distribution of the number (in  $10^9$ ) of arborescences; Figure 9 shows the distribution of the flip diameter.

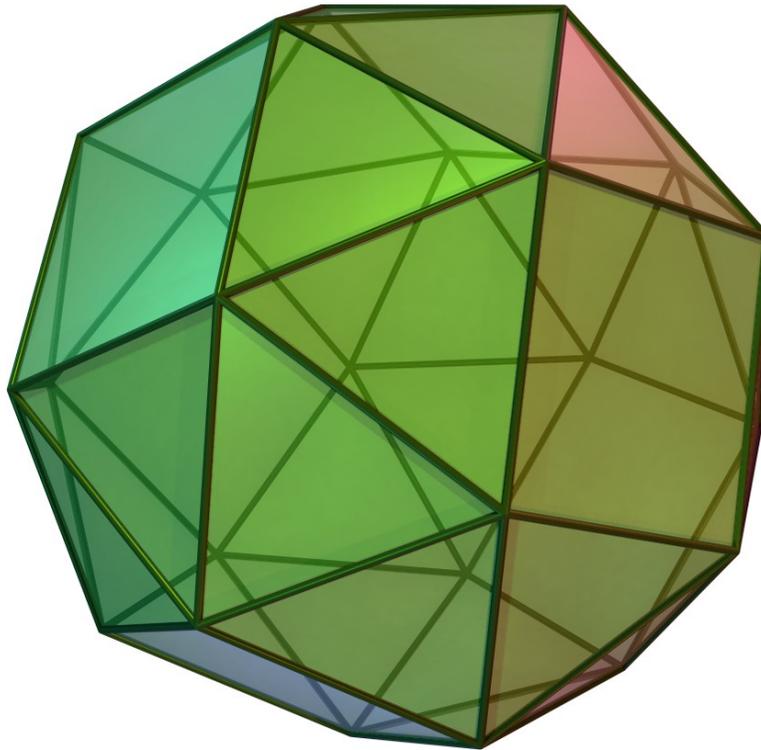


FIGURE 5. A picture of the snub cube [24].

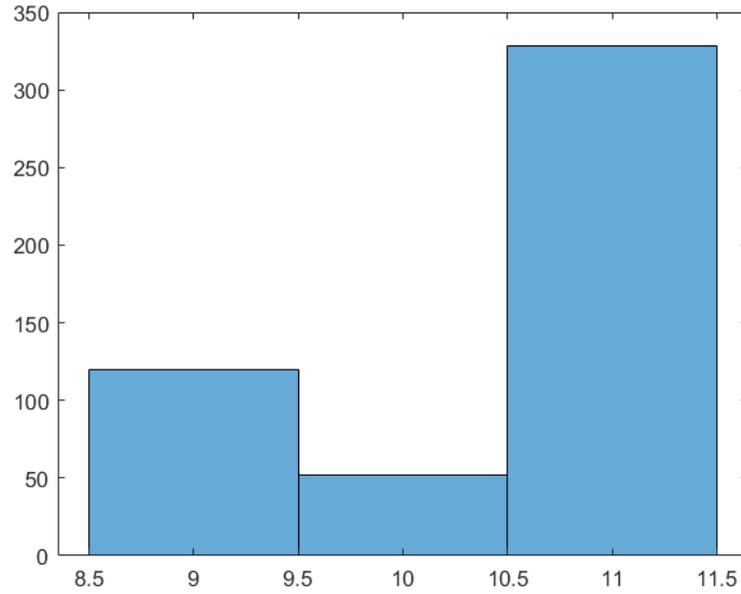


FIGURE 6. The distribution of the monotone height of the snub cube given by 500 random objective functions.

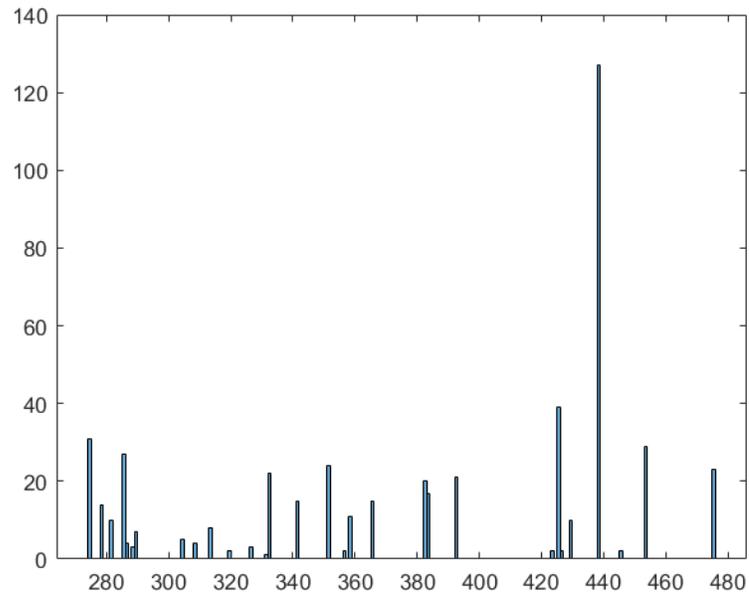


FIGURE 7. The distribution of the number of monotone paths of the snub cube given by 500 random objective functions.

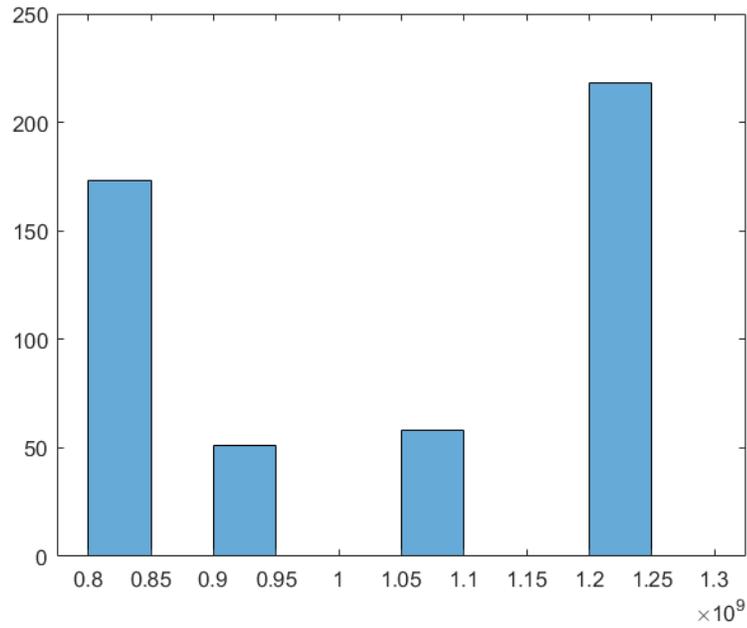


FIGURE 8. The distribution of the number of arborescences of the snub cube given by 500 random objective functions.

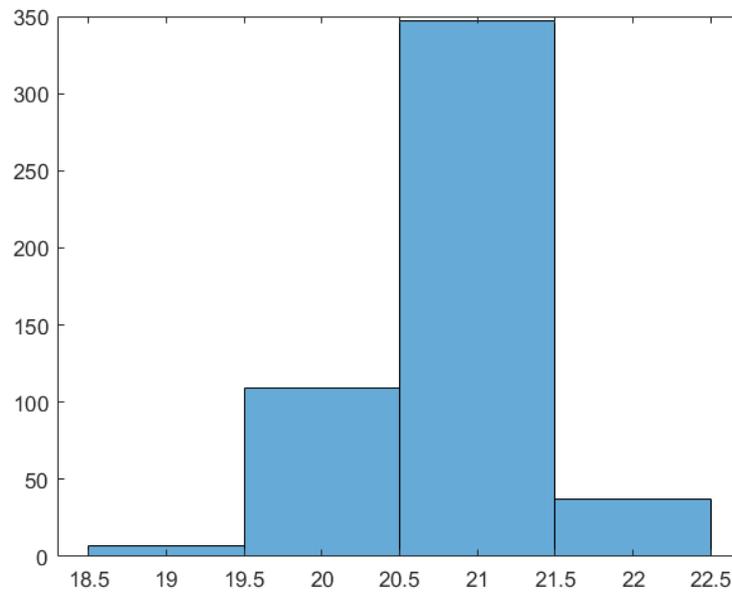


FIGURE 9. The distribution of the flip diameter of the snub cube given by 500 random objective functions.

Since the flip graph depends on the number of monotone paths, we cannot display all of the possible flip graphs. Instead, we will show three flip graphs of the snub cube: one with the fewest number of paths (274 paths, Figure 10), one with the most number of paths (476 paths, Figure 11), and one with the most occurrence (438 paths, Figure 12).

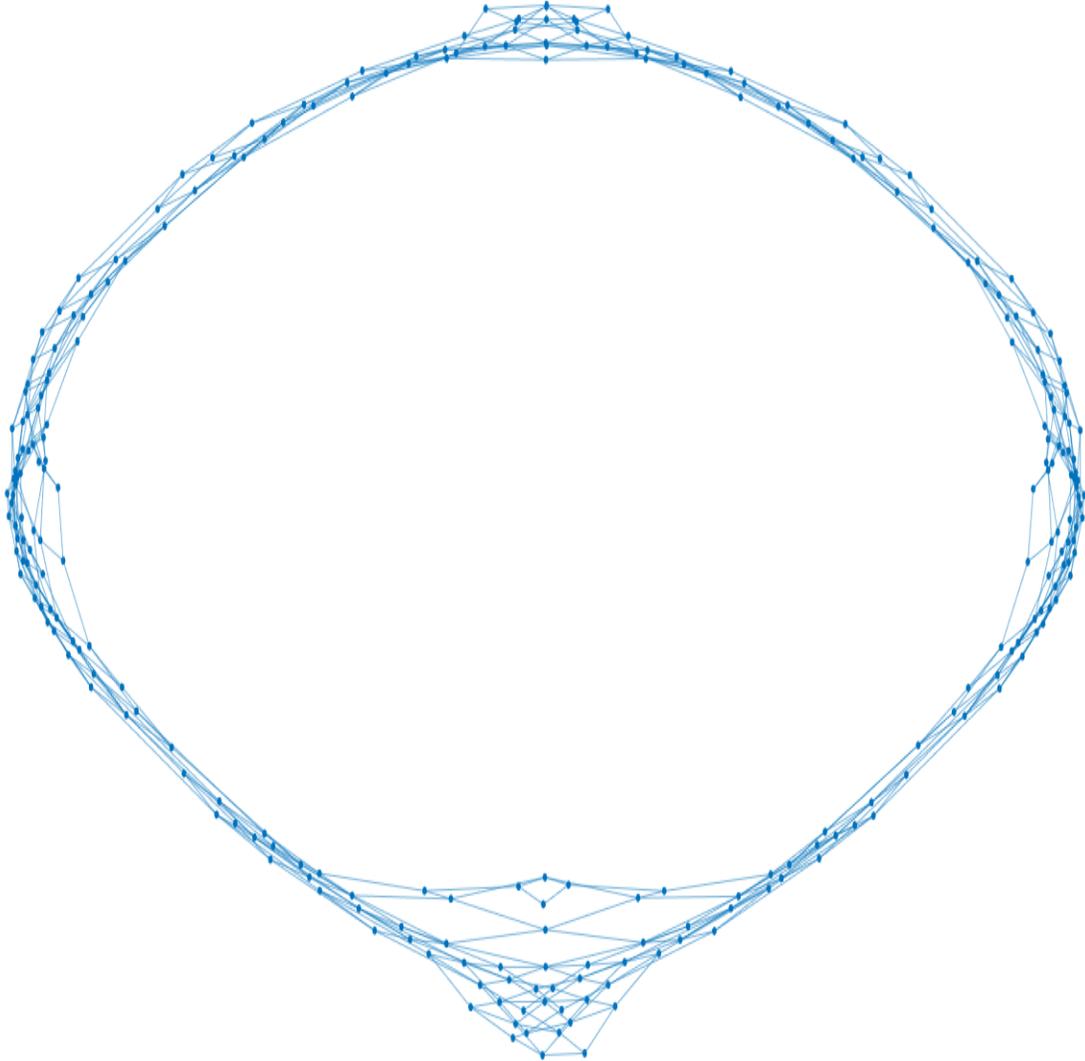


FIGURE 10. The flip graph of the snub cube with 274 monotone paths.

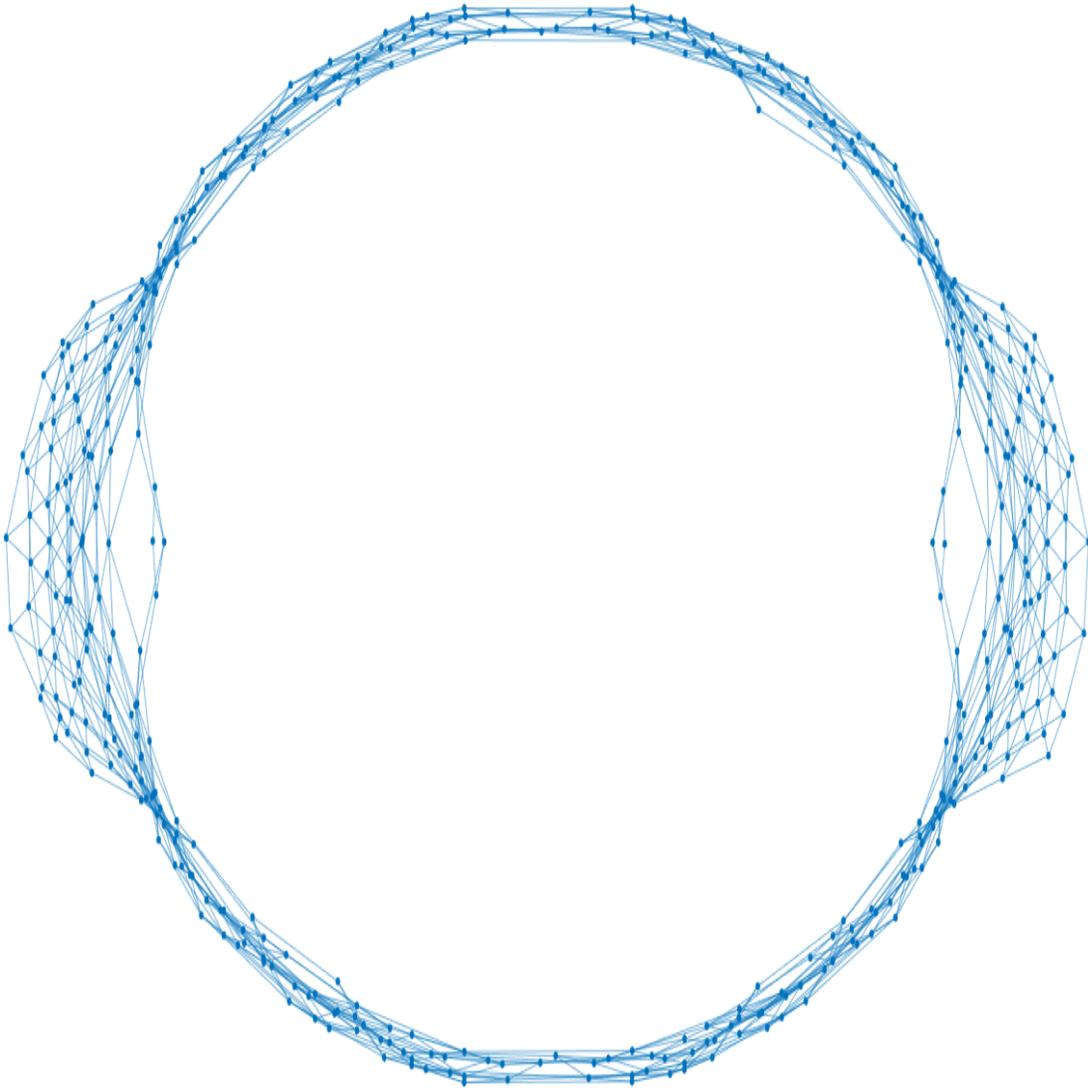


FIGURE 11. The flip graph of the snub cube with 476 monotone paths.

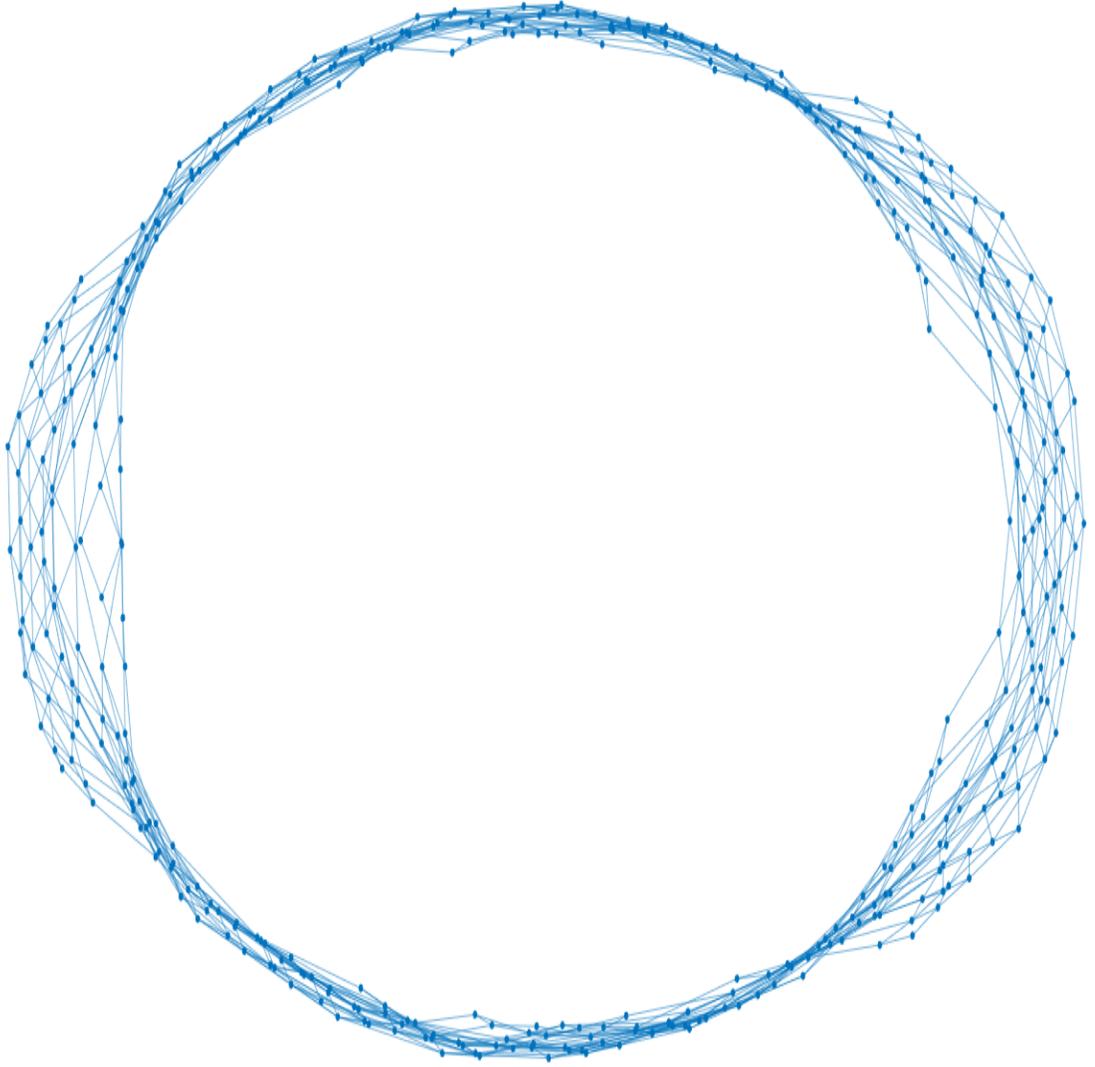


FIGURE 12. The flip graph of the snub cube with 438 monotone paths.

We failed to calculate the flip graph for the snub dodecahedron because of two reasons. First, depending on the objective function, the snub dodecahedron has different numbers of monotone paths (from 8672 to 20316 paths using 500 different objective functions), and computing a flip graph for a polytope that has big number of monotone paths takes a lot of computation power.

Second, our polytope data is not accurate enough to carry out the calculation of the flip graph. When inputting the data for the vertices of the snub dodecahedron [19] into the program `cdd/cdd+`, the program returns too many inequalities, and some of them are extremely similar and redundant. We deleted the redundant inequalities and put the inequality file back into the program `cdd/cdd+`. From the program `cdd/cdd+`, we got a vertex file of 120 vertices, which is exactly two times the number of original vertices. When we investigate the coordinates of these vertices, we found that each vertex has split into two. Thus, we deleted the redundant vertices. Although we can use the edited files to do calculations for the features except the flip graph, our edit increases the round-off error which is too big to compute the flip graph.

The following are some figures for the snub dodecahedron, including the distribution of the monotone height (Figure 14), the distribution of the number (in  $10^4$ ) of monotone paths (Figure 15), and the distribution of the number (in  $10^{22}$ ) of directed arborescences (Figure 16).

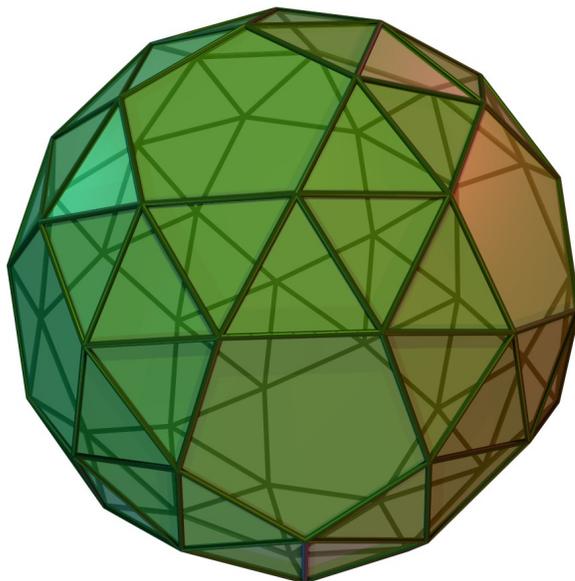


FIGURE 13. The picture of the snub dodecahedron [24].

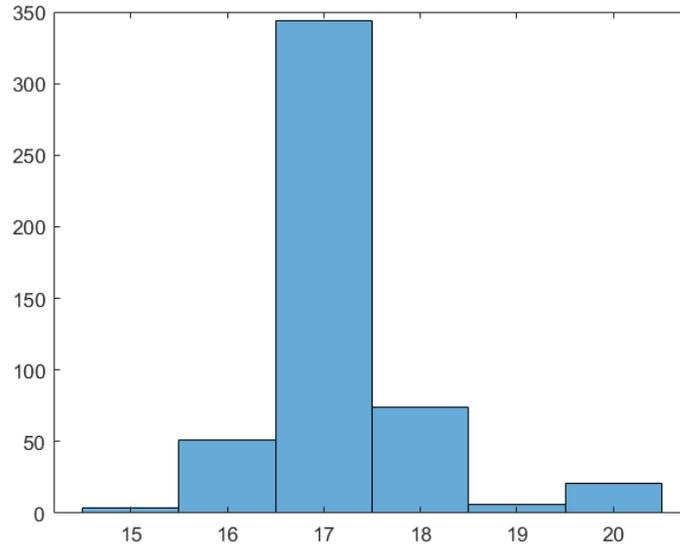


FIGURE 14. The distribution of the monotone height of the snub dodecahedron given by 500 random objective functions.

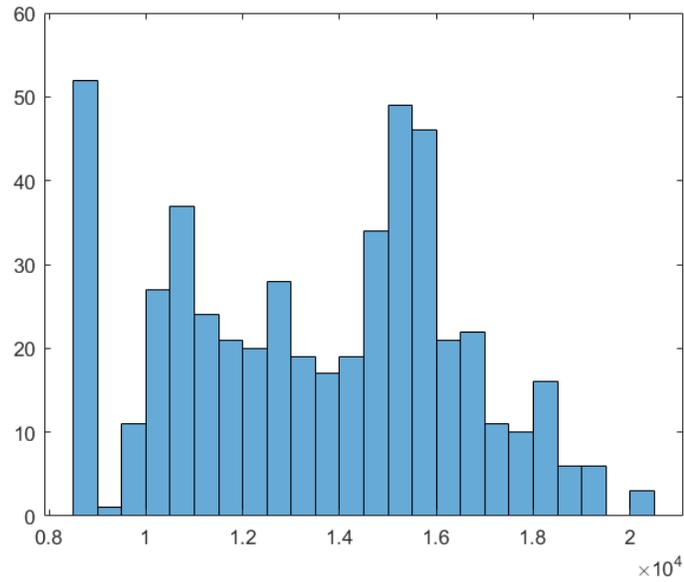


FIGURE 15. The distribution of the number of monotone paths of the snub dodecahedron given by 500 random objective functions.

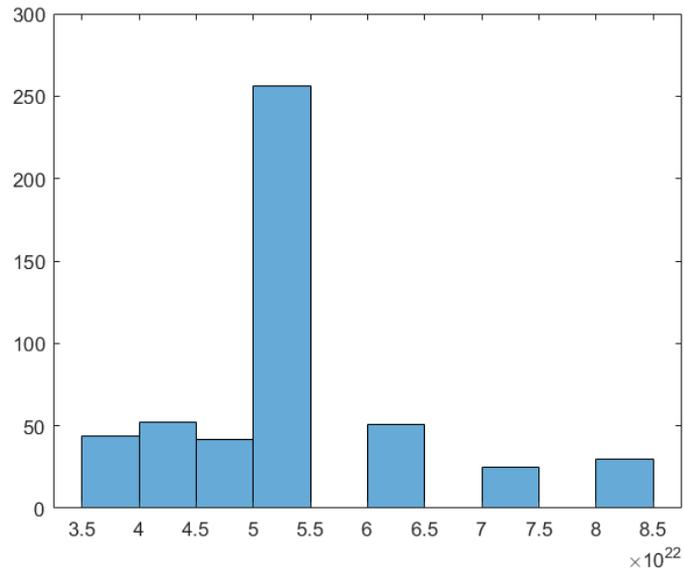


FIGURE 16. The distribution of the number of arborescences of the snub dodecahedron given by 500 random objective functions.

#### 4.4. The Monotone Diameter and the Monotone Height of $3d$ Simple Polytopes

We used the random  $3d$  polytope generator which uses cutting plane method (see Section 2.1) to generate 50,000 random  $3d$  simple polytopes. The polytopes generated using this method will be roundish and will not have a similar behavior of polytopes of special design, like a Klee-Minty cube. By adjusting the number of cuts, we were able to obtain polytopes with different numbers of facets, ranged from eight facets to 80 facets. We separated these polytopes into groups, and kept the groups that contain at least 30 different polytopes, which are groups from eight facets to 72 facets.

Then, we applied a fixed set of 26 objective functions on each polytope to obtain the characteristics. These objective functions come from the set  $F = \{[x_1, x_2, x_3] \text{ s.t. } x_1 = 0, \pm 1 ; x_2 = 0, \pm 1 ; x_3 = 0, \pm 1\} \setminus [0, 0, 0]$  with a little permutation to reduce the possible degeneracy.

In this experiment, we are interested in two ratios: the monotone diameter/the number of facets (the monotone diameter ratio) and the monotone height/the number of facets (the monotone height ratio). The graphs below are the results for the monotone diameter ratio, and each subgraph is a normalized histogram, representing the ratio for the polytopes with the same number of facets. The x-axis of each subgraph represents the monotone diameter ratio, and the y-axis shows the percentage of occurrence.

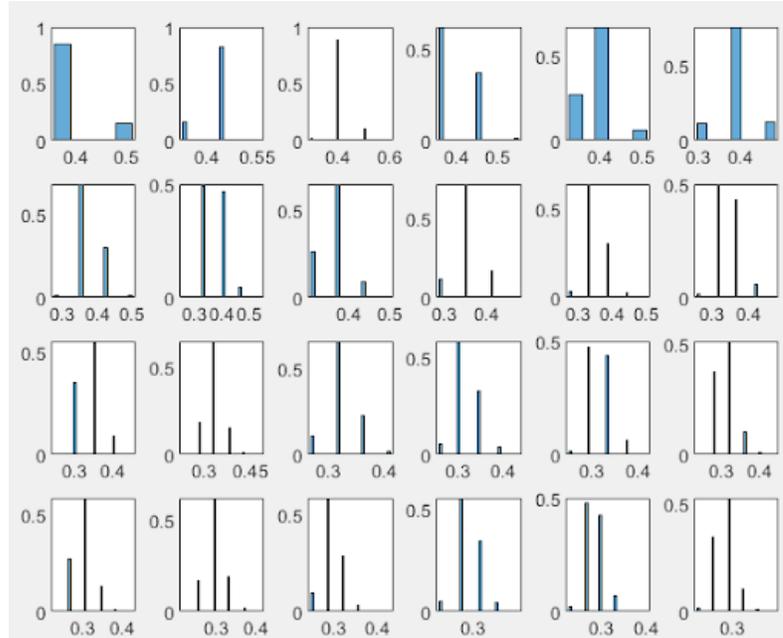


FIGURE 17. The monotone diameter ratio for polytopes with 8-31 facets.

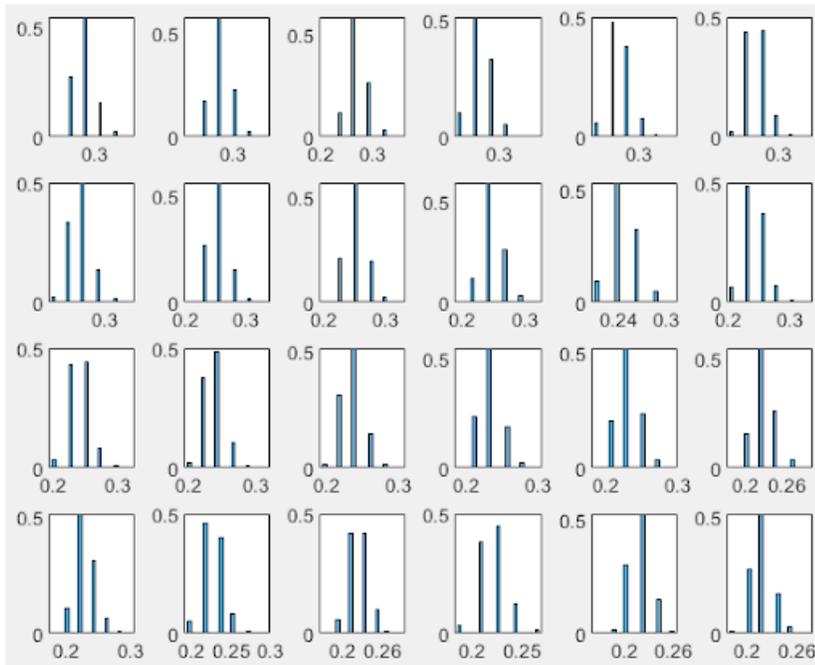


FIGURE 18. The monotone diameter ratio for polytopes with 32-55 facets.

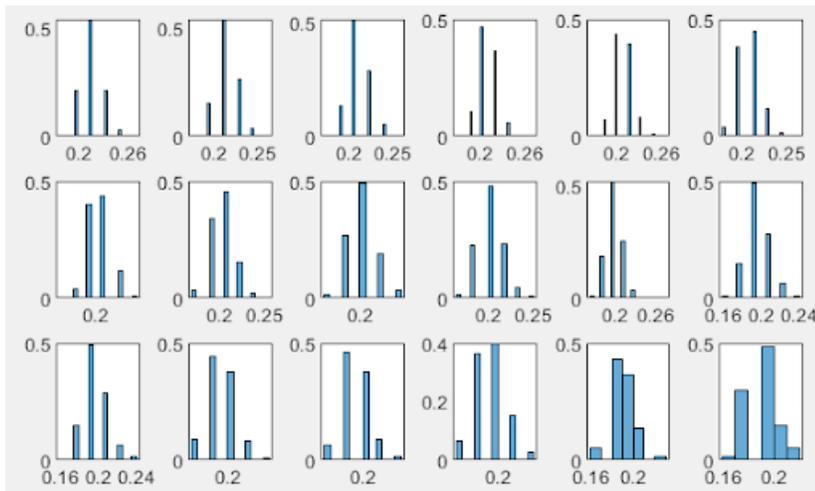


FIGURE 19. The monotone diameter ratio for polytopes with 56-72 facets.

From these graphs, we can see that each subgraph looks like a skewed normal distribution. Hence, we observed that, in general, as the number of facets increases, the mean of the monotone diameter ratio decreases. From this observation, we found it to be interesting to put all these ratios into one normalized histogram. We decided to give an equal weight to each number of facets from eight to 72. The figure below shows the histogram of the occurrence of each ratio:

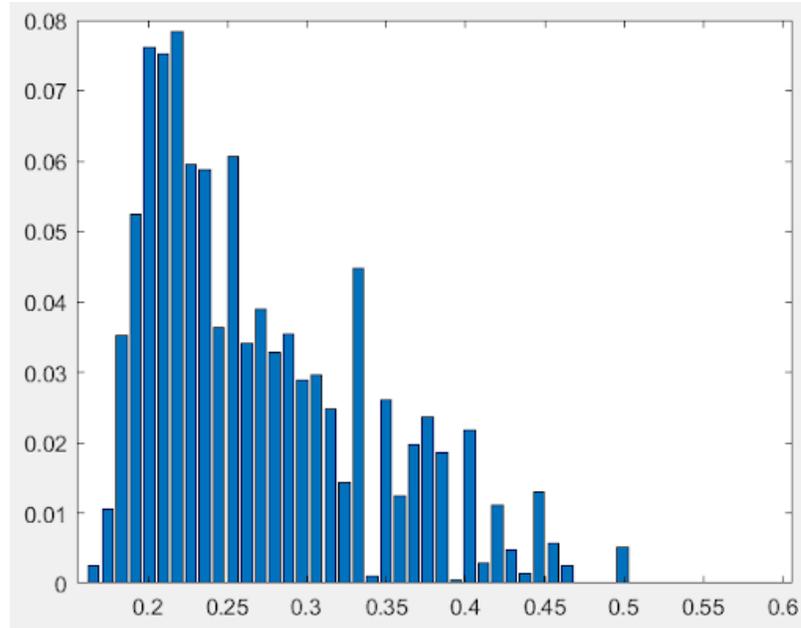


FIGURE 20. Weighted monotone diameter histogram.

If our observation is correct that the mean of the monotone diameter ratio decreases as the number of facets increases, we will see more bars on the left side of the 0.2 on the x-axis, and the ratio may approach zero or a small number as the number of facets increases.

The monotone height ratio also follows a similar pattern, and the results of the monotone height ratio are shown in the figures below. The x-axis represents the monotone height ratio, and the y-axis represents the percentage of occurrence. Similarly, each subgraph looks like skewed normal distribution, and the mean is decreasing as the number of facets increases:

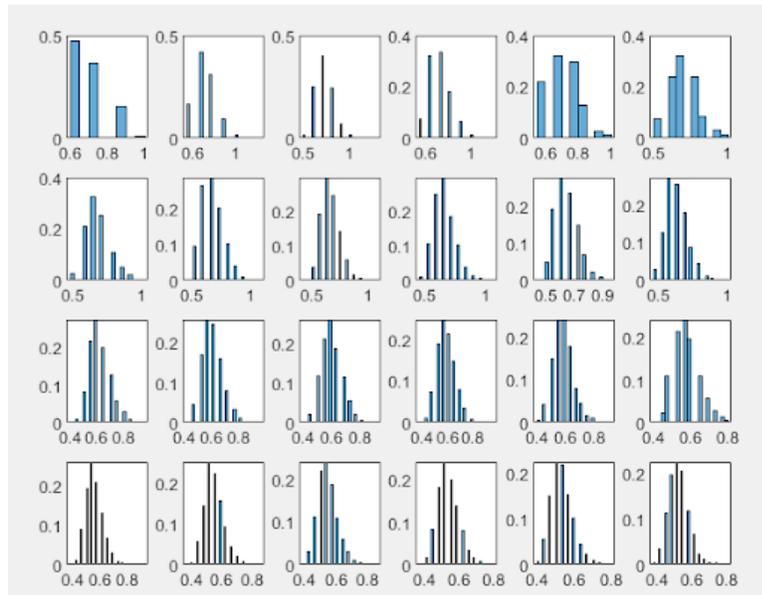


FIGURE 21. The monotone height ratio for polytopes with 8-31 facets.

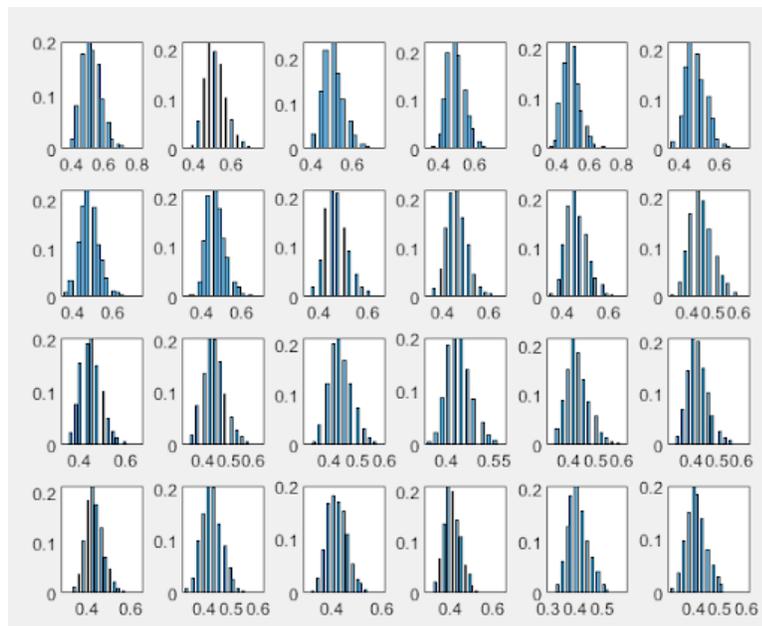


FIGURE 22. The monotone height ratio for polytopes with 32-55 facets.

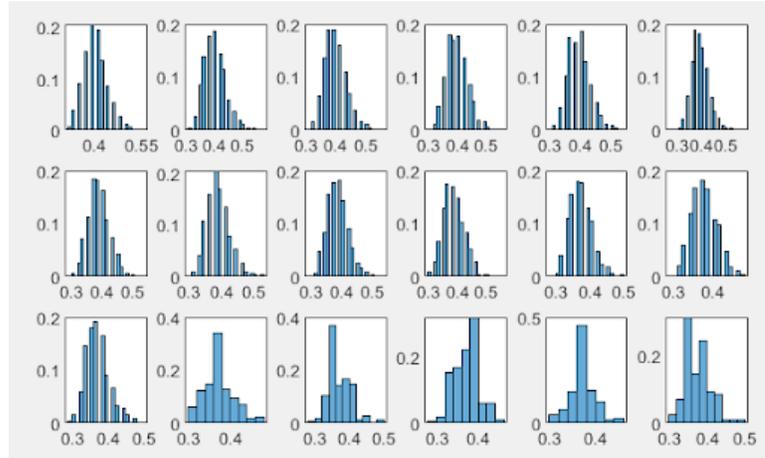


FIGURE 23. The monotone height ratio for polytopes with 56-72 facets.

We also put all the ratios into one normalized histogram presented below, and it seems to follow the pattern we observed in the monotone diameter ratio:

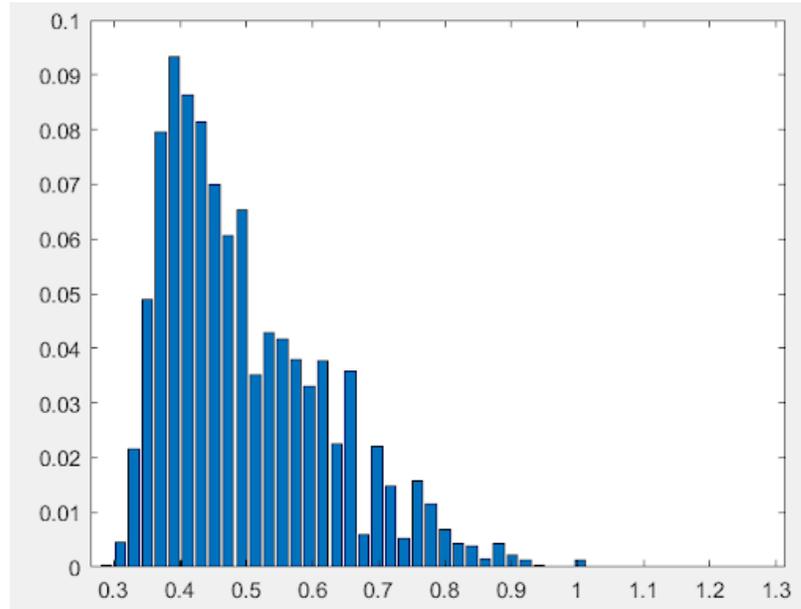


FIGURE 24. Weighted monotone diameter histogram.

#### 4.5. The Monotone Height of Birkhoff Polytope

$B_n$  is a convex polytope that comes from an  $n \times n$  doubly stochastic matrix, a matrix with non-negative real entries and every column sum and every row sum equal to 1. It is also called the Birkhoff Polytope [16]. We want to explore some features of the Birkhoff Polytope, with a focus on the monotone height. We generated  $B_3$ ,  $B_4$ ,  $B_5$ , and  $B_6$  based on the idea in [16]. Then, we applied 500 different objective functions on  $B_3$ ,  $B_4$ , and  $B_5$ , as well as 150 different objective functions on  $B_6$ . The following table is what we get:

Birkhoff Polytope	# of runs	# of facets	# of vertices	diameter	monotone diameter	monotone height	# of paths	# of arborescences
$B_3$	500	9	6	1	1	5	16	120
$B_4$	500	16	24	2	2	15 to 23	188340 to 2812400	0.75e21 to 1.83e21
$B_5$	500	25	120	2	2	80 to 107	9.26e25 to 3.79e30	0.16e180 to 4.36e180
$B_6$	150	36	720	2	2 to 3	453 to 514	2.13e138 to 5.99e150	Out of bound

Among these numbers, we are interested in the distribution of the monotone height, so we ran more experiments on  $B_4$ ,  $B_5$ , and  $B_6$ . The distributions are presented in Figure 25, 26, 27 respectively. The  $x$ -axis represents the monotone height, and the  $y$ -axis represents the number of occurrences of each monotone height:

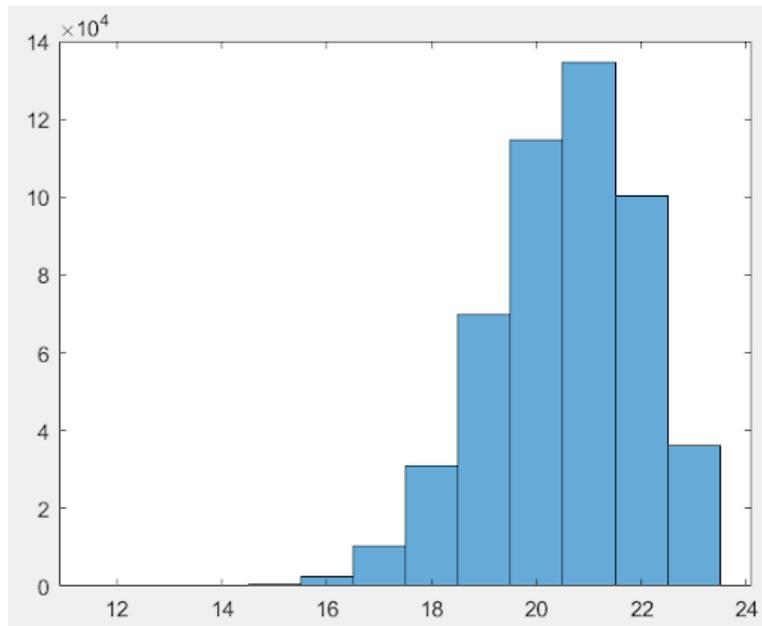
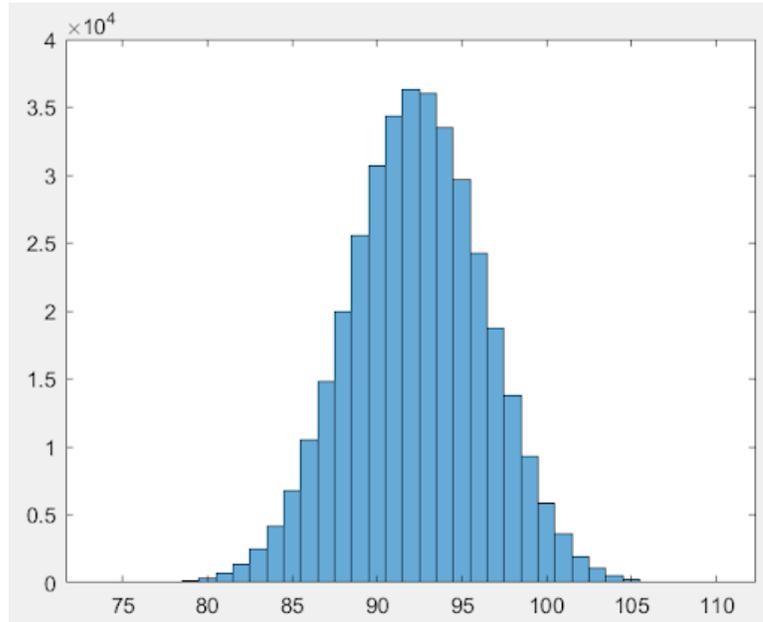
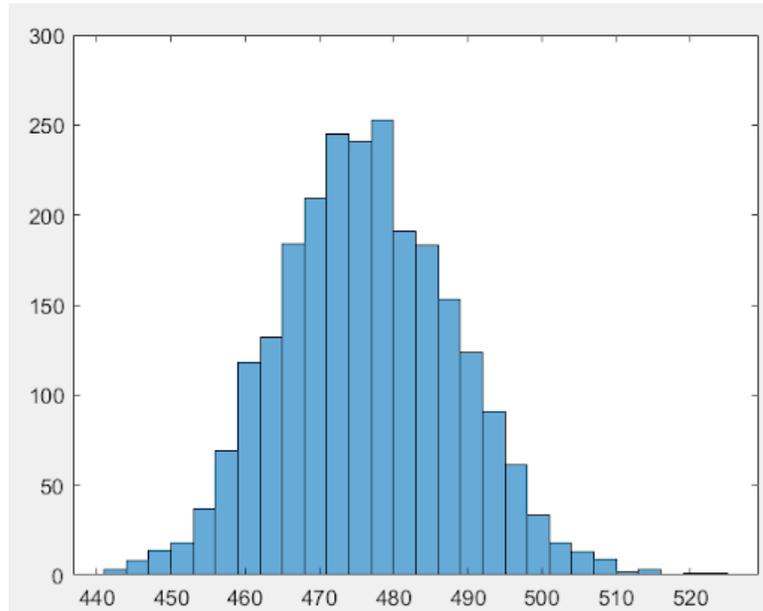


FIGURE 25. The monotone height distribution of  $B_4$ .

FIGURE 26. The monotone height distribution of  $B_5$ .FIGURE 27. The monotone height distribution of  $B_6$ .

We also tried to compute the features for  $B_7$ . Unfortunately,  $B_7$  requires too much computation power. Instead of calculating the exact monotone height, we chose to calculate the monotone height divided by ten to speed up the process. Below is what we get for 60 runs. The numbers in  $x$ -axis times ten is roughly the monotone height:

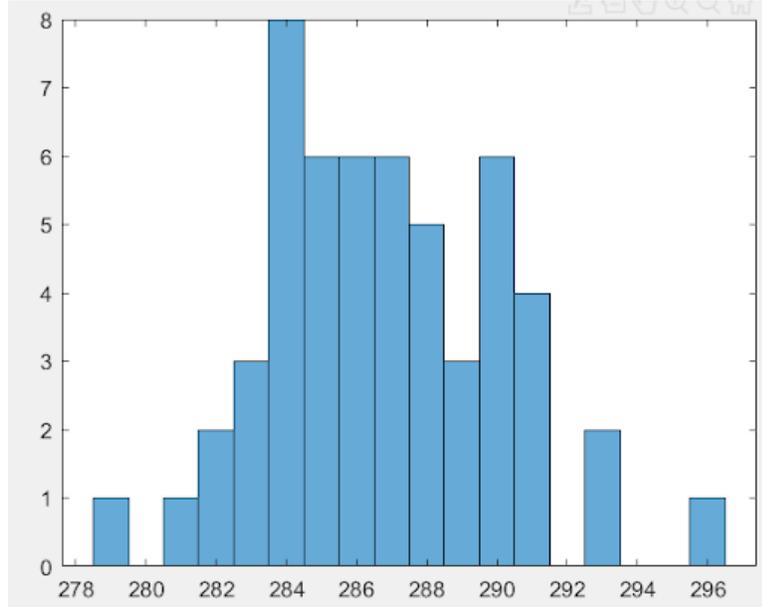


FIGURE 28. The monotone height distribution of  $B_7$ .

From these figures, we can see that the monotone height of  $B_n$  based on randomly generated objective functions is normally distributed with a mean is growing exponentially in terms of  $n$ . In particular, there can be a monotone path traveling through all the vertices of the directed polyhedral graph coming from  $B_4$  and special objective functions.

#### 4.6. The Monotone Height of the Traveling Salesman Polytope

Traveling salesman problem (TSP) is an NP-hard problem, and there is no known efficient algorithm for solving the TSP. The setting of the problem is that a salesman wants to minimize the cost to travel all  $n$  cities and return to the starting city, without going into the same city twice. In the traveling salesman polytope (TSP polytope for short), each vertex represents a solution to the TSP (a Hamiltonian cycle in a complete graph of  $n$  nodes), and each entry in the objective function represents the cost to travel between a pair of cities. By varying the objective functions, we obtain TSPs with different costs between cities.

We are interested in the distribution of the monotone height of the TSP polytope. However, the computation power needed for the  $n$ -city TSP polytopes is exponential, and we could only compute the features for the 5-city and the 6-city TSP polytopes, as well as the distribution of the monotone height of the 6-city and the 7-city TSP polytope. The following table shows the features for the 5-city and the 6-city TSP polytopes:

TSP Polytope	# of runs	# of facets	# of vertices	diameter	monotone diameter	monotone height	# of paths	# of arborescences
5-city	500	20	12	2	2	10	682	1.81e7
6-city	500	100	60	2	2	39 to 55	7.69e12 to 2.08e15	9.15e71 to 3.49e72

From the features, we see that the 5-city TSP polytope is not very interesting. Therefore, we ran more experiments to compute the monotone height for the 6-city and the 7-city TSP polytopes. The histogram of the distribution of the monotone height are shown below in Figure 29 and Figure 30, with the x-axis representing the monotone height and the y-axis representing the number of occurrences of each monotone height.

One thing interesting for the 6-city TSP polytope is that the maximum of the monotone height of the 6-city TSP polytope is 58, which means that, starting from one vertex, the simplex method needs to go through all except one vertex (59 in total except the starting vertex) to reach the optimum, which is very inefficient. The minimum of the monotone height is 32, and we do not know if it could be lower. The mean is 46.58, the standard deviation is 2.69, and the median is 47. The minimum for the monotone height for the 7-city TSP polytope in 8800 runs is 193, and the max is 266. The mean is 223.3, the standard deviation is 9.43, and the median is 223.

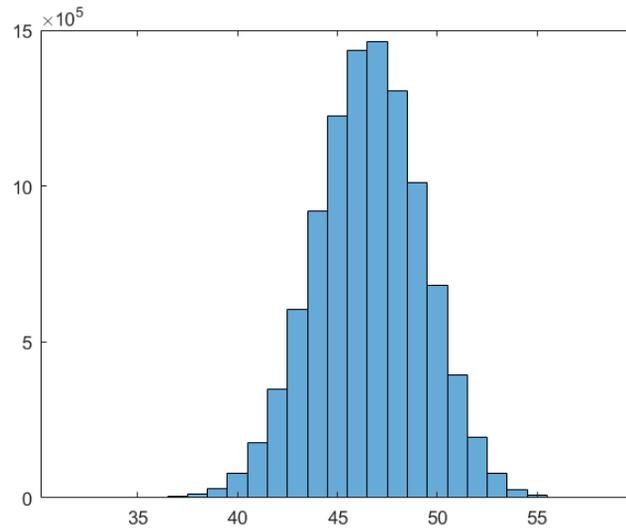


FIGURE 29. The distribution of the monotone height for 6-city TSP polytope using 10,000,000 random objective functions.

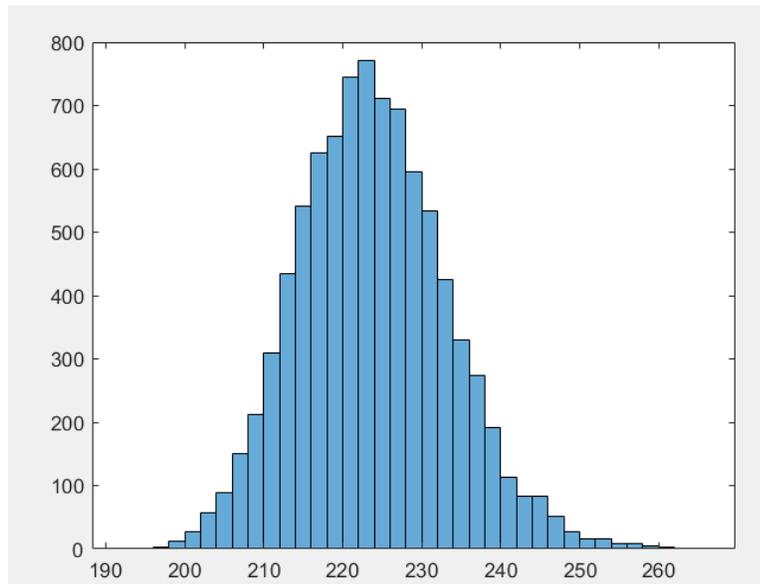


FIGURE 30. The distribution of the monotone height for 7-city TSP polytope using 8,800 random objective functions.

## Acknowledgements

I would like to express my thanks to my thesis advisor Dr. Jesús A. De Loera for providing me the opportunity to explore this topic, the guidance and support throughout this project. I would like to thank Dr. Christos A. Athanasiadis for introducing me to the fascinating topic of the flip graph of polytopes. I would also like to express my gratitude to Moïse Blanchard and Zhenyang Zhang who provided me a substantial amount of help during the research. Finally, we thank NSF for the support from NSF grant DMS-1818969.

## Bibliography

1. MATLAB Answers, <https://www.mathworks.com/matlabcentral/answers/417396-calculating-all-paths-from-a-given-node-in-a-digraph>.
2. Christos Athanasiadis, Jesús A. De Loera, and Zhenyang Zhang, *Enumerative problems for arborescences and monotone paths on polytopes*, <https://arxiv.org/abs/2002.00999>, 2020.
3. Christos A. Athanasiadis, Paul H. Edelman, and Victor Reiner, *Monotone paths on polytopes*, *Mathematische Zeitschrift* **235** (2000), 315–334.
4. Per Bergström, *Plot 2D/3D region (version 1.1.0.0)*, <https://www.mathworks.com/matlabcentral/fileexchange/9261-plot-2d-3d-region>, 2010.
5. Karl H. Borgwardt, *The average number of pivot steps required by the simplex-method is polynomial*, *Zeitschrift für Operations Research* **26** (1982), 157–177.
6. Antoine Deza, Eissa Nematollahi, and Tamás Terlaky, *How good are interior point methods? Klee-Minty. cubes tighten iteration-complexity bounds*, *Math. Program.* **113** (2008), 1–14.
7. Komei Fukuda, *cdd/cdd+ Reference Manual*, Institute for Operations Research, ETH-Zentrum, CH-8092 Zurich, Switzerland, March 1999.
8. Donald Goldfarb and William Y. Sit, *Worst case behavior of the steepest edge simplex method*, *Discrete Applied Mathematics* **1** (1979), 277–285.
9. Gil Kalai, *Linear programming, the simplex algorithm and simple polytopes*, *Mathematical Programming* **79** (1997), 217–233.
10. Victor Klee, *Diameters of polyhedral graphs*, *Canadian Journal of Mathematics* **16** (1964), 602–614.
11. Victor Klee, *Heights of convex polytopes*, *Journal of Mathematical Analysis and Applications* **11** (1965), 176 – 190.
12. Victor Klee and George J. Minty, *How good is the simplex algorithm?*, *Inequalities–III*, Proceedings Third Symposium, 1969, Academic Press, New York, 1972, pp. 159–175. MR 0332165 (48 #10492)
13. Victor Klee and David W. Walkup, *The  $d$ -step conjecture for polyhedra of dimension  $d < 6$* , *Acta Mathematica* **117** (1967), 53–78.
14. Chengjun Li, *Study on Using the Greatest Improvement Pivot Rule of Simplex Method to the Klee and Minty Example*, High Performance Networking, Computing, and Communication Systems, Springer, Berlin, Heidelberg, 2011, pp. 431–438.
15. Jesús A. De Loera, *New insights into the complexity and geometry of linear optimization*, *Optima* **87** (2011), 1–11.
16. Jesús A. De Loera, Fu Liu, and Ruriko Yoshida, *A generating function for all semi-magic squares and the volume of the Birkhoff polytope*, *Journal of Algebraic Combinatorics* **30** (2008), no. 1, 113–139.
17. MATLAB, *Shortest path distances of all node pairs - MATLAB*, <https://www.mathworks.com/help/matlab/ref/graph.distances.html>.
18. MATLAB, *9.6.0.1072779 (r2019a)*, The MathWorks Inc., Natick, Massachusetts, 2019.
19. David I. McCooley, *Archimedean solids*, <http://dmccooley.com/polyhedra/java/Archimedean.html>.
20. Francisco Santos, *A counterexample to the hirsch conjecture*, *Annals of Mathematics* **176** (2012), no. 1, 383–412.
21. Michael J. Todd, *The monotonic bounded hirsch conjecture is false for dimension at least 4*, *Mathematics of Operations Research* **5** (1980), no. 4, 599–601.
22. ———, *The basic George B. Dantzig, by Richard W. Cottle*, *Bulletin of the American Mathematical Society* **48** (2011), no. 1, 123–129.
23. Robert J. Vanderbei, *Linear programming*, Springer US, 2014.

24. Wikipedia, *Archimedean solid*, [https://en.wikipedia.org/wiki/Archimedean\\_solid](https://en.wikipedia.org/wiki/Archimedean_solid), Apr 2020.
25. \_\_\_\_\_, *Platonic solid*, [https://en.wikipedia.org/wiki/Platonic\\_solid](https://en.wikipedia.org/wiki/Platonic_solid), Apr 2020.
26. Chutong Wu, *Statistical Analysis of Four Pivot Rules for the Simplex Method*, [https://www.math.ucdavis.edu/files/3415/5301/8154/Statistical\\_Analysis\\_of\\_Four\\_Pivot\\_Rules\\_for\\_the\\_Simplex\\_Method.pdf](https://www.math.ucdavis.edu/files/3415/5301/8154/Statistical_Analysis_of_Four_Pivot_Rules_for_the_Simplex_Method.pdf), 2019.