Statistical Analysis of Four Pivot Rules for the Simplex Method

By

CHUTONG WU

SENIOR THESIS

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

BACHELOR OF SCIENCE

in

MATHEMATICS

in the

COLLEGE OF LETTERS AND SCIENCE

of the

UNIVERSITY OF CALIFORNIA,

DAVIS

Approved:

Jesús A. De Loera

March 2019

ABSTRACT.

In 1947, George B. Dantzig first invented the simplex method that can solve most cases of linear programming problems. Later, other pivot rules such as Bland, Greatest Descent, and Steepest Edge are developed as well. However, none of the pivot rules was able to perform well in all cases, and it remains unknown the existence of such pivot rule that can out-perform the existing ones.

This thesis is served as a preliminary version of an ongoing project. We examine behaviors of four pivot rules on cone-type convex polytopes. To be specific, we randomly generate non-degenerate LP from dimension three to seven, then analyze various features (i.e. depth, breath, degree of interior node, etc.) of the pivot rules from the spanning trees and observe patterns.

Contents

CHAPTER 1

Introduction

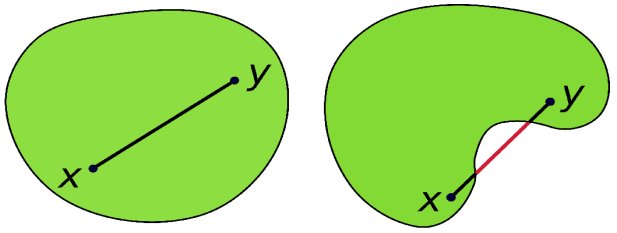
In this thesis, we will examine the behavior of Dantzig's pivot rule, Bland's pivot rule and its variations, Greatest Descent pivot rule, and Steepest Edge pivot rule on random generated cone-type polytopes from dimension three to seven. To do this, we consider the spanning trees for each simplex pivot rule.

In Chapter 1, we review some basic definitions and theorems of linear programming. Then, we establish some geometric understandings of the LP that are essential for our implementations. In Chapter 2, we discuss the software package and its usage. In Chapter 3, we present the statistical analysis of the result data. In the last chapter, we talk about the implementation of the Shadow Vertex algorithm and difficulties with it while drawing the spanning trees.

1.1. Basic Definitions

Before we give the rigorous definition a linear program, we first establish some geometric understandings of a polytope.

DEFINITION 1. A set X is called **convex** if for all $x, y \in X$ and for all $\lambda \in [0, 1]$, the points $\lambda x + (1 - \lambda y)$ belong to X.



(a) A convex set (b) A non-convex set [6]

DEFINITION 2. The **convex hull** of a set X is the smallest convex set containing X.

DEFINITION 3. The **convex cone** of a set X is the smallest set containing all non-negative linear combinations of the elements of X.

DEFINITION 4. A **hyperplane** is a set $\{x \in \mathbb{R}^n : A \cdot x = b\}$ for any given $A \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Geometrically, a n-dimensional hyperplane can be viewed as dividing $\mathbb{R}^n$ into two parts, which we call half-spaces.

DEFINITION 5. A **half-space** is a set $\{x \in \mathbb{R}^n : A \cdot x \leq b\}$ for any given $A \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

Now we can define a **polytope** as a finite intersection of half-spaces.

DEFINITION 6. A n-dimensional polytope (or a n-polytope) may be specified as the set of solutions to a system of linear inqualities

$$Ax \leq b,$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

For example, a polygon is a *2*-polytope and a polyhedron is a *3*-polytope. Then, a **convex polytope** can be characterized as a convex hull of finitely many points in the n-dimensional space $\mathbb{R}^n$.

DEFINITION 7. Let $\mathcal{C}$ be a convex polytope. We say that $x \in \mathcal{C}$ is a **vertex** (or **extreme point** of $\mathcal{C}$ if whenever $x \in \{(1-\lambda)u + \lambda v : 0 \leq \lambda \leq 1\}$ for some $u, v \in \mathcal{C}$, it must be the case that either $x = u$ or $x = v$. Namely it cannot be represented as a proper convex combination of two other (distinct) points of the convex set.

1.2. The Simplex Method

With the previous definitions, we are now ready to define an LP (Linear Program).

DEFINITION 8. A standard **Linear Program** is of the form:

$$\begin{aligned} \text{min. } & p^T x \\ \text{s.t. } & Ax \leq b, \quad x \geq 0 \end{aligned}$$

where $x, p \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$

The constraints of a linear program can each be realized as half-spaces, and so their intersections form a polytope. The most popular method to solve linear programs is the simplex method. All feasible solutions are encapsulated by the convex polytope, and the simplex method provides algorithms to examine such polytopes to an optimal solution [**?**]. The simplex method methodically examines the value at each vertex to find the value that yields the highest objective value. This is done by moving from vertex to vertex of the polytope, and checking each adjacent vertex. If every neighboring vertex decreases the objective value or does not increase the objective value, the process finishes and we have attained an optimal solution. Note that a polytope may give multiple optimal solutions, and the process by which it chooses a optimal solution is dependent on the way it chooses which vertices to examine.

To help understand how the simplex method works, we present a simple, two-dimensional example. Consider a farmer who is trying to decide what animals he should buy to raise on his farm. He wants to buy cows and chickens, and would like to purchase as many total animals as he can with the money he has. Say cows cost \$200 each and chickens cost \$20

each, and the farmer has $1050 to spend. Assume our farmer wants at least one cow and at least four times as many chickens as cows, but his coop will hold no more than 20 chickens.

We can visualize solutions to this farmer's dilemma as points (x, y) in the Cartesian plane, where x is the number of cows and y the number of chickens he will purchase. The farmer wants as many animals as possible, which means he is trying to maximize the objective function

$$x + y.$$

The farmer's financial and practical constraints can be represented as the following inequalities:

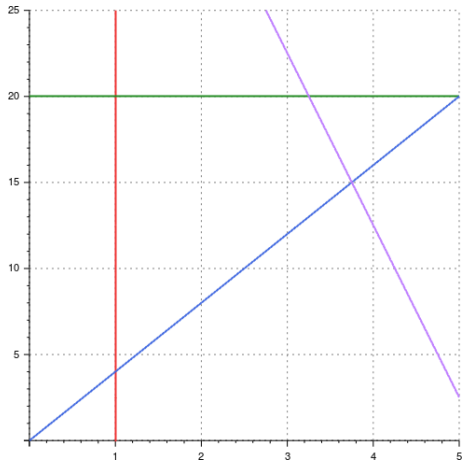
$$200x + 20y \leq 1050,$$

$$x \geq 1,$$

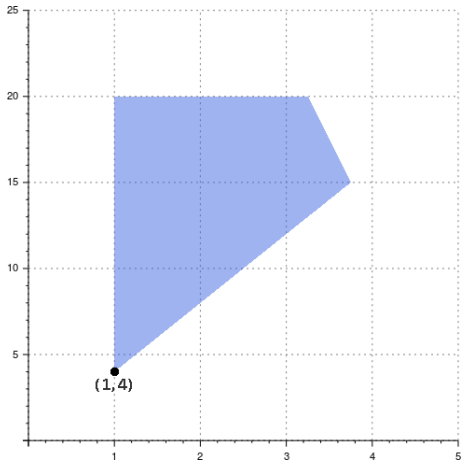
$$y \geq 4x,$$

$$y \leq 20.$$

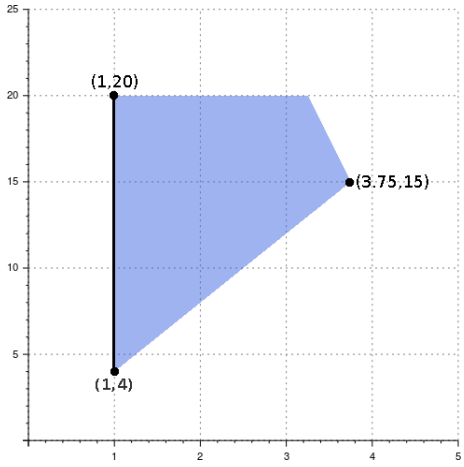
We graph the constraints below:



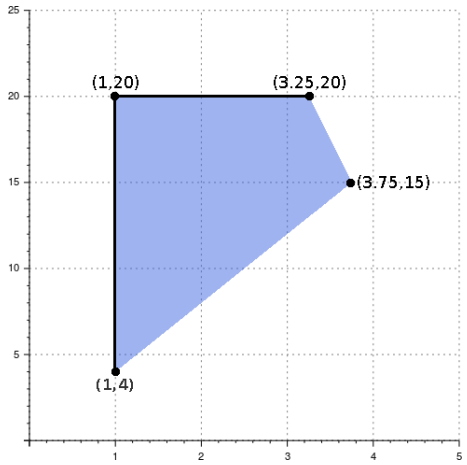
Notice that these lines cut out a polygon, which is a 2-dimensional polytope. Every point inside of the polygon is a valid solution to the farmer's problem. The question that remains is which of these points is the optimal solution. The answer can be found using the simplex method. As we said before, the central idea behind the simplex method is that optimal solutions are found at the vertices of polytopes. Thus, we can find the optimal solution by traveling between vertices until we cannot improve our objective value by traveling. In this our example, our objective is to maximize $x + y$, the total number of animals. To begin the simplex method, we need to start at an arbitrary vertex that gives a feasible solution. It is easy to check that $(1, 4)$ is a feasible solution, so we begin there.



Now we look to the neighboring vertices to see if we can improve our objective value, the total number of animals. We can move directly up to attain an objective value of $1 + 20 = 21$ or to the right diagonal to reach $3.75 + 15 = 18.75$. We choose to move up to reach the higher value.



We again look to our neighbors to see if we can improve. We see that we can move directly to the right to attain $3.25 + 20$, so we move to that point.



Now we look to our neighbors and see that they have strictly lower objective values. Thus, we have found our optimal solution, $(3.25, 20)$. Though the simplex method gets much more complicated with higher dimensions and more constraints, the basic idea remains the same. As long as our objective function and constraints are linear, we are guaranteed to find an optimal solution if one exists.

The above process can also be understood in the matrix tableau form, as we demonstrate below. With the definition of a LP in mind, we make the following definitions.

DEFINITION 9. A **slack variable** x_{n+i} is defined by

$$x_{n+i} := A_{i\cdot}x - b_i, \quad i = 1, \cdots, m. \tag{1.1}$$

Adding slack variables can turn the LP into the **canonical form**:

$$\min \quad z = p^T x \tag{1.2}$$

$$\text{s.t.} \quad Ax = b, \quad x \geq 0. \tag{1.3}$$

where $x, p \in \mathbb{R}^{m+n}, b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m\times(m+n)}$.

DEFINITION 10. A **basic solution** is a vector $x \in \mathbb{R}^{m+n}$ that satisfies $Ax = b$ where for some $B \subset \{1, 2, \cdots, l\}$, $A_{\cdot B}$ has linearly independent columns and $x_j = 0$ for $j \notin B$.

DEFINITION 11. We say that x is a **basic feasible solution** (BFS) if x is a basic solution and in addition it satisfies the nonnegativity condition $x \geq 0$.

DEFINITION 12. Given a point x in the feasible region, a constraint $A_{i\cdot}x \leq b$ is called **active** if $A_{i\cdot}x = b$ and **inactive** if $A_{i\cdot}x > b$.

Now, we introduce the first theorem.

THEOREM 1 ([2, Theorem 5.2.1]). *Let $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times(m+n)}$ be given. If $Ax = b$, $x \geq 0$ has a solution, then it has a **BFS** $x \in \mathbb{R}^{m+n}$. That is, there is a set $B \subseteq \{1, \cdots, l\}$*

such that

$$\sum_{j\in B} A_{\cdot j}x_j = b, \quad x_B \geq 0,$$

and $A_{\cdot B}$ has linearly independent columns.

PROOF (SEE [2, p. 119]). Since the system $Ax = b, b \geq 0$ has a solution, the LP

$$\begin{aligned} \min. \quad & p'(Ax - b) \\ \text{s.t.} \quad & Ax \geq b, \quad x \geq 0. \end{aligned}$$

must have optimal objective value 0. Solve the LP by setting up an initial tableau as follows:

		$x_{\cdot,1}$	$x_{\cdot,2}$	1
$y_{1,\cdot}$	$=$	$A_{1,1}$	$A_{1,2}$	$-b_1$
$y_{2,\cdot}$	$=$	$A_{2,1}$	$A_{2,2}$	$-b_2$
z	$=$	$p'A$	$p'A$	$-p'b$

Performing a block pivot on $(1,1)$ yields

		$y_{1,\cdot}$	$x_{\cdot,2}$	1
$x_{\cdot,1}$	$=$	$A_{1,1}^{-1}$	$-A_{1,1}^{-1}B$	$A_{1,1}^{-1}b_1$
$y_{2,\cdot}$	$=$	$A_{2,1}A_{1,1}^{-1}$	$A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$	0
z	$=$	p'	p'	0

where the zeros in the last column follow from $p'(Ax - b) = p'y = 0$ and $y \geq 0$, which together imply $y = 0$ at the optimum. Therefore,

$$y_{2,\cdot} = A_{2,1}A_{1,1}^{-1}y_{1,\cdot} + (A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2})x_{\cdot,2} + A_{2,1}A_{1,1}^{-1}b_1 - b_2 = 0$$

implies that

$$A_{2,1}A_{1,1}^{-1}b_1 - b_2 = 0.$$

Substituting the optimal values of x and y into the original tableau, we obtain

$$0 = \begin{bmatrix} y_{1,\cdot} \\ y_{2,\cdot} \end{bmatrix} = \begin{bmatrix} A_{1,1} \\ A_{2,1} \end{bmatrix} A_{1,1}^{-1}b_1 + \begin{bmatrix} A_{1,0} \\ A_{2,0} \end{bmatrix} 0 - b.$$

We complete the proog by setting $\bar{x} = \begin{bmatrix} A_{1,1}^{-1}b_1 \\ 0 \end{bmatrix}$ and J the columns corresponding to columns in $X_{\cdot,1}$.

□

The above theorem shows that, the set B such that columns of $A_{\cdot B}$ are linearly independent fixes simplex tableau. Therefore, we define the following terms.

DEFINITION 13. Given the index set $B \subseteq \{1, \cdots, l\}$ such that $A_{\cdot B}$ has linearly independent columns, we call x_B the **basic** variables and x_N where $N = \{1, \cdots, l\} \setminus B$ the **nonbasic** variables, and such index set B the **basis**.

A LP then can be written as:

$$\begin{aligned} \text{min. } & z = p_B x_B + p_N x_N \\ \text{s.t. } & A_B x_B + A_N x_N = b, \quad x \geq 0. \end{aligned}$$

where the constraints can be rewritten as

$$x_B = A_B^{-1}(b - A_N x_N), \quad x \geq 0$$

which together with the objective function yields

$$z = (p_N - p_B A_B^{-1} A_N) x_N + p_B A_B^{-1} b.$$

Therefore, we can rewrite the LP in the **tableau form**:

$$\begin{array}{c|cc} & x_N & b \\ \hline x_B & A_B^{-1}A_N & A_B^{-1} \\ z & p_N - p_B A_B^{-1} A_N & p_B A_B^{-1} \end{array} \tag{1.4}$$

Note that the tableau depends entirely on the choice of the basis B, so every basis generates its unique tableau. The simplex method provides us such a way to choose the basis B so that the optimum solution z can be attained. To be specific, within each iteration, we pick a **entering variable** from the nonbasic variables to enter the set of basic variables and a **leaving variable** from the basic variables to leave, together forms a new basis; hence a new tableau that yields a improved z.

Several ways had been invented to pick the entering and leaving variables, which we call **pivot rules**. Details will be discussed in the next section.

1.3. Pivot Rules

In this thesis, we consider the following 4 pivot rules: Dantzig, Blands (and its variations), Greatest Descent, and Steepest Edge. To better understand the procedure, we demonstrate a 3-D example. Consider the LP:

$$\begin{aligned} & \text{max.} \quad x_1 + x_2 + x_3 \\ & \text{s.t.} \quad \begin{bmatrix} 569/162 & 403/111 & 380/63 \\ 299/57 & 1356/307 & 17/61 \\ 4006/463 & 448/93 & 74/65 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 1357/103 \\ 2883/290 \\ 3316/227 \end{bmatrix}, \quad x_1, x_2, x_2 \geq 0 \end{aligned} \tag{1.5}$$

As shown in Figure 2, the constraints in (1.5) correspond to the cone-type polytope and its interiors. The level set (objective function) $z = x_1 + x_2 + x_3$ spans a 2-dimensional plane in $\mathbb{R}^3$. When x_1, x_2 and x_3 take the value 0, the value of the objective function z is 0. To view this geometrically, we see that at $(0, 0, 0)$ (vertex 3), the polytope intersects the 2-D plane $x_1 + x_2 + x_3 = 0$. As we translate the 2-D plane along its normal vector $(1, 1, 1)$, the value of z increases, meaning that the value of the objective function is improved. And as long as the translations of the level set still intersects the polytope, there must exists a

feasible solution such that the objective function attains the value z.

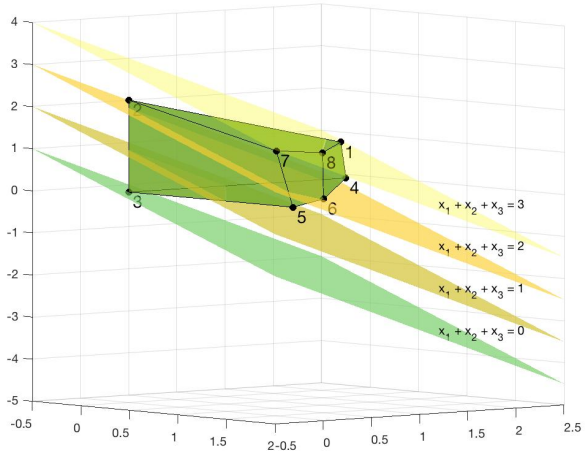


FIGURE 1. 3-D example

In this example, when z attains its optimum $\overline{z}$, the level set $\overline{z} = x_1 + x_2 + x_3$ can only intersect the polytope through a vertex, a an edge, or a face. In a more general setting, when an objective function with n variables attains its optimum, the $(n-1)$-dimensional flat surface spanned by the equation $\overline{z} = \sum_{i=1}^{n} x_i$ can only intersect the polytope in $(n-1)$ ways, namely through 0-skeletons (vertices), 1-skeletons (edges), ..., $(n-1)$-skeletons (facets).

Nonetheless, we know that if all feasible solutions are contained in a convex polytope, then there must exists a vertex on the polytope such that the objective function attains its optimum when the variables take value at such vertex. However, in many cases, the constraints might not form a convex polytope, in which case we say the polytope has **rays** (the polytope is not "closed" in certain directions). If the directions of the rays coincide with the direction of the normal vector of the objective function, the LP have no optimum solution. On the other hand, even if the convex polytope exists, it is inefficient to check the values z takes on all vertices. Therefore, more economic ways are invented to solve linear programs, and the simplex pivot rules are a subset of them.

Following the genearal procedure of pivot rules as in (1.1), we add slack variables x_4 to x_9 to the inequalities in (1.5) to turn them into equations:

$$\begin{aligned} 569/162x_1 + 403/111x_2 + 380/63x_3 + x_4 &= 1357/103 \\ 299/57x_1 + 1356/307x_2 + 17/61x_3 + x_5 &= 2883/290 \\ 4006/463x_1 + 448/93x_2 + 74/65x_3 + x_5 &= 3316/227, \quad x_1, \cdots, x_6 \geq 0. \end{aligned} \tag{1.6}$$

We can write (**??**) into the tableau form. In each column, the numbers represent coefficients of the variables in the first row. We also write $-z$ instead of z to turn the LP into a standard minimization problem. The yellow vector corresponds to the constant vector b in (1.3), the green matrix represents matrix $A = (a_{ij}) \in \mathbb{R}^{m\times(m+n)}$ in (1.3), and the purple vector is p in (1.2)

(1.7)

1	x_1	x_2	x_3	x_4	x_5	x_6
1357/103	569/162	403/111	380/63	1	0	0
2883/290	299/57	1356/307	17/61	0	1	0
3316/227	4006/463	448/93	74/65	0	0	1
$-z$	-1	-1	-1	0	0	0

Since the coefficients of x_4 to x_6 form a non-singular matrix, by Definition 13, we know that x_4 to x_6 are the basic variables and are depended on the nonbasic variables x_1, x_2 and x_3. The basis B is the index set $\{4, 5, 6\}$. We further define the set N to be the index set of the nonbasic variables, which is $\{1, 2, 3\}$ in this example. Among the nonbasic variables, we say the ones with negative coefficients the **eligible** variables, denoted Elg. Note that the LP is improved only when we increase the value of an eligible variable. In (1.7), $Elg = \{1, 2, 3\}$. Moreover, we introduce an **order set** for all variables, denoted $Order$, indicating the position of each variable in the matrix tableau. In this case, the order set is the trivial order from 1 to n. For a tableau with with variables in a different order the order set is different (i.e. $\{x_1, x_2, x_3\}$ with $Order = \{2, 1, 3\}$).

Recall that in each iteration of the simplex method, the pivot rules provide us such a way to pick one index variable from the basis to leave and one index variable from outside to enter the basis. In the essence, the pivot rules pick an element b in B and n in N and swap their positions.

1.3.1. Dantzig's Rule.

- entering variable: the one with the most negative reduced cost
- leaving variable: the one with minimum (positive) ratio

We demonstrate this process with an example of the path from $(0, 0, 0)$ to the optimum with respect to the objective function. At $(0, 0, 0)$, we have the same tableau as in (1.7). In addition, we add a ratio column on the right, which each row i of it indicates the ratio between the constant term b_i and the coefficient of the entering variables $a_{ij} \in A$ in the corresponding row, where j is the entering variable. We add the value of x_i's in the last row. Notice that the numbers here is the value of the variables in the first row instead of their coefficients. We then have:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
1357/103	569/162	403/111	380/63	1	0	0	949/253
2883/290	299/57	1356/307	17/61	0	1	0	904/477
3316/227	4006/463	448/93	74/65	0	0	1	883/523
$-z$	-1	-1	-1	0	0	0	
x	0	0	0	1357/103	2883/290	3316/227	

*Tableau form of the Dantzig pivot rule. The green columns indicate the nonbasic variables and their coefficients, the yellow cell is the entering variable and the purple cell is the leaving variable. The brown cells highlight the important steps in the computation. The gray cells refer to the corresponding vertex we attain at this iteration.

Here, since we want to minimize $-z$, we pick the nonbasic variable with the most negative coefficient in the objective function. In this case, all coefficients are the same (we say this is a **tie**) and we pick the one with the **minimum index**.

To pick the leaving variable, we look at the positive ratio between the constant term and the coefficients of x_1 (the entering variable). The (positive) ratio in each row stands for the maximum value x_1 can be increased until any variables in the equation has to go negative, in which case no longer satisfies the constraints. Therefore, the nonnegativity of x_i's impose x_1 to be increased only by the minimum ratio, which is 883/532 in this case. This forces x_6 to be zero and become the leaving variable.

Notice that we only look at the positive ratios because if the ratio between the constant terms and coefficients of the entering variable is negative, increasing the entering variables will never violate the contraints.

Until now, we have obtained the new basis $B = \{1, 4, 5\}$ and $N = \{2, 3, 6\}$. Apply the formula we derived from (1.4), we get the second tableau:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
883/523	1	103/185	5712/43411	0	0	463/4006	5248/409
3789/523	0	1521/908	2601/467	1	0	$-874/2153$	679/522
1123/1035	0	1667/1114	$-407/989$	0	1	$-1489/2456$	
$-z$	0	$-4881/11012$	$-33/38$	0	$-463/4006$	463/4006	
x	883/523	0	0	3789/523	1123/1035	0	

By reading the gray bar, we know that this tableau corresponds to vertex 5. Repeat the above process, we pick x_3 as the leaving variable because it has the most negative reduced cost $-33/38$ and pick x_4 as the entering variable because the minimum ratio between the constant and x_3 is 679/522. Swap the position of the index 3 and 4, we obtain the third tableau:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
883/582	1	301/582	0	$-289/12233$	0	375/2996	883/301
679/522	0	277/921	1	467/2601	0	$-618/8479$	1544/357
2599/1604	0	1975/1219	0	736/9961	1	$-586/921$	11275/11274
$-z$	-1	$-1382/7591$	$-653/4188$	653/4188	0	259/4954	
x	883/582	0	679/522	0	2599/1604	0	

The point $(883/582, 0, 697/522)$ is vertex 7. We pick x_2 as the entering variable and x_5 as

the leaving variable. Then:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
21156/21157	1	0	0	−534/11311	−957/2998	281/856	1453/477
11263/11262	0	1	0	193/4232	1219/1975	−97/247	
49677/49678	0	0	1	793/4782	−230/1239	4294/94945	5196/235
$-z$	−1	−1	−546/4859	325/1979	546/4859	−407/21181	
x	21156/21157	11263/11262	49677/49678	0	0	0	

This tableau corresponds to vertex 8. This time, x_6 is selected as the entering variable and x_7 as the leaving variable, which leads to:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
1320/601	569/162	403/111	380/63	1	0	0	
607/704	299/57	1356/307	17/61	0	1	0	
1453/477	4006/463	448/93	74/65	0	0	1	
$-z$	179/3058	−336/2081	−353/3768	336/2081	353/3768	0	
x	0	1320/601	607/704	0	0	1453/477	

Observe that all the coefficients of the nonbasic variables in the objective function are negative. This indicates that the value of z can no longer be improved. So we have attained the optimum solution, which is vertex 1. Therefore, starting from vertex 3, the Dantzig pivot rule generates the path $3-5-7-8-1$.

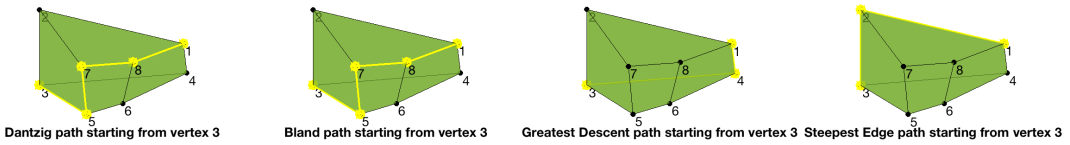


FIGURE 2. Paths from vertex 3 to the optimum for four pivot rules

1.3.2. Bland's Rule.

- entering variable: x_i such that $i = \min\{j \mid j \in Elg\}$ [5]
- leaving variable: the one with minimum (positive) ratio

Started from vertex 3 again, the first iteration of the Bland's rule pick the same entering and leaving variables, so we obtain the same second tableau as in Dantzig:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
883/523	1	103/185	5712/43411	0	0	463/4006	1028/339
3789/523	0	1521/908	2601/467	1	0	−874/2153	717/397
1123/1035	0	1667/1114	−407/989	0	1	−1489/2456	3194/4405
$-z$	0	−4881/11012	−3/38	0	−463/4006	463/4006	
x	883/523	0	0	3789/523	1123/1035	0	

Here, we look at the eligible terms, namely the indices of nonbasic variables with negative coefficients in the objective function, and pick the minimum index. In this case, we pick 2 from $\{2, 3\}$. Computing the minimum ratio yield x_5 as the leaving variable. With

the new basis $B = \{1, 2, 4, 7, 8, 9\}$ and $N = \{3, 5, 6\}$, we obtain the new tableau. Repeating the above process yields the path $3 - 5 - 6 - 8 - 1$, shown in Figure 2.

1.3.3. Leftmost of the Eligible Variables Rule. Another way to view the Bland's rule is that we set the order of indices as $Order = \{1, 2, \cdots, n\}$ and each time we pick the **leftmost** element in it. Therefore, different **Order** set yield different choices of entering and leaving variables. That is,

- entering variable: x_{i_k} such that $k := \min\{j \mid x_{i_j} \in \text{Elg}, j \in \text{Order}\}$.
- leaving variable: the one with minimum (positive) ratio

With the same example, if we rewrite the trivial order set to be $\{1, 3, 2, \cdots\}$, which swaps the position of x_2 and x_3. Then we have to pick x_{3_2} from $Elg = \{x_{3_2}, x_{2_3}\}$, which makes the entering variables to be x_3 instead of x_2 as in Bland's.

1.3.4. Greatest Descent.

- entering variable: for all eligible variables, perform the ratio test and pick the variable that has the most negative product of minimum ratio and reduced cost.
- leaving variable: the one with minimum (positive) ratio

The Greatest Descent pivot rule picks the direction which improves the objective function the most at the current iteration. It picks the entering variable $j \in Eli$ such that

$$\min_{j \in Eli} \left\{ p_j \cdot \min \left\{ \frac{b_i}{a_{ij}} \mid j \in Eli \right\} \right\}$$

is attained. In this example:

1	x_1	x_2	x_3	x_4	x_5	x_6	$x_1 -$ ratio	$x_2 -$ ratio	$x_3 -$ ratio
1357/103	569/162	403/111	380/63	1	0	0	949/253	3001/827	2217/1015
2883/290	299/57	1356/307	17/61	0	1	0	904/477	2280/1013	4459/125
3316/227	4006/463	448/93	74/65	0	0	1	883/523	1028/339	4183/326
$-z$	-1	-1	-1	0	0	0			
x	0	0	0	1357/103	2883/290	3316/227			
$\times$	$-883/523$	$-2280/1013$	$-2217/1015$						

*Tableau form of the Greatest Descent pivot rule. In addition to the previous tableau, we add several ratio columns on the right representing the ratio tests. In addition, we add a row on the bottom, with each value indicate the product of the minimum ratio w.r.t such variable and its coefficient in the objective function.

Observe that here $Eli = \{1, 2, 3\}$. The ratio tests of x_1, x_2, and x_3 yield 883/523, 2280/1013 and 2217/1015, respectively. Multiplying the minimum ratio with its corresponding p_j's leads to the most negative term $-2280/1013$. Therefore, x_2 is the entering variable and x_5 the leaving variable. Repeat the above process yields the path $3 - 4 - 1$, as shown in Figure 2.

1.3.5. Steepest Edge.

- entering variable: for all eligible terms, compute the 2-norm spanned by the vector composed of its coefficients, then pick the one with the maximum ratio between its coefficients in the objective function and its 2-norm.
- leaving variable: the one with minimum ratio.

Since each inequality in the LP represents a half space, the vector $(a_{1j}, a_{2j}, \cdots, a_{mj})^T$ can be viewed as the displacement in such direction . And the steepest edge pivot rule finds the direction which improves the objective function the most with respect to displacement in a unit length [7]. For example, at vertex 3, we have:

1	x_1	x_2	x_3	x_4	x_5	x_6	ratio
1357/103	569/162	403/111	380/63	1	0	0	2217/1015
2883/290	299/57	1356/307	17/61	0	1	0	4459/125
3316/227	4006/463	448/93	74/65	0	0	1	4183/326
$-z$	-1	-1	-1	0	0	0	
x	0	0	0	1357/103	2883/290	3316/227	
$\|\|\cdot\|\|_2$	1869/173	3736/491	4111/652				
$\cdot/\cdot$	$-173/1869$	$-491/3736$	$-652/4111$				

*Tableau form of the Steepest Edge pivot rule. We add another row under the x which represent the 2-norm of the vector composed of the coefficients of each eligible terms. In addition, the last row indicates the ratios between the b_j's and the 2-norm of the x_j's.

Following the Steepest Edge pivot rule, we aim to find

$$\min_{j \in Eli} \frac{p_j}{||x_j||_2} \quad \text{where} \quad ||x_j||_2 = \left(\sum_{i=1}^{n} a_{ij} \right)^{\frac{1}{2}}$$

In this case, $-652/4111$ is the most negative among the three, so x_3 is the entering variable, and x_4 becomes the leaving variable after computing the minimum ratio. In the end, it generates the path $3 - 2 - 1$ (Figure 2).

1.4. Non-degeneracy and Simple Polytopes

It is worth noticing that throughout the computations of the 3-D example, the minimum ratio is always unique. It is natural to ask if that is always the case. At the same time, we see that at each iteration, the matrix tableau has its unique solution $x \in \mathbb{R}^{m+n}$, with the first n terms a vertex on the polytope. It leads to the question that can a vertex on the polytope generate a unique simplex tableau? The answer to both question is yes only if the corresponding convex polytope of the LP is **simple**, or the LP is **non-degenerate** [1].

Definition 14. A BFS, or vertex x, is said to be **degenerate** if one or more of the basic variables is assigned the value zero.

Definition 15. An LP is **degenerate** if in a BFS, one of the basic variables takes on a zero value.

From (1.1), it means that more than n constraints in the $m + n$ equations are active at vertex x. This corresponds to the geometric definition of simple polytope [3].

Definition 16. A d-dimensional **simple polytope** is a d-dimensional polytope each of whose vertices are adjacent to exactly d edges (also d facets).

THEOREM 2 ([1, Theorem 1.8]). *(**Fundamental Representation Theorem for Vertices**) A point x in the convex polytope $C = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, where $A = (a_{ij})_{m\times n}$ and $b \in \mathbb{R}^m$, is a vertex of this polytope if and only if there exist an index set $\mathcal{I} \subset \{1, \cdots, n\}$ with such that x is the unique solution to the system of equations*

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i \in \mathcal{I}, \tag{1.8}$$

Moreover, if x is a vertex, then one can that $|\mathcal{I}| = n$ in (1.8), where $|\mathcal{I}|$ denotes the number of elements in $\mathcal{I}$.

From Theorem 1, the proof is reduced to show that x is a vertex (extreme point) of the convex polytope $\mathcal{C}$ if and only if x is a BFS.

COROLLARY 1. *The set of extreme points, E, of the feasible region $\mathcal{C}$ is exactly the set, F of all BFS of the LP.*

PROOF. Refer to [4]. □

Therefore, we've answered the questions from the beginning of this section. We also present an immediate result followed from Theorem 2 here.

THEOREM 3. *If the constraint set, P, is bounded, then the objective function $z = p^T x$ achieves its minimum on P at an extreme point of P.*

PROOF. Refer to [1]. □

1.5. The Spanning Tree

Recall from the Dantzig example (2) that the path from vertex 3 to the optimum is $3 - 5 - 7 - 8 - 1$. Similarly, we can compute paths from other vertices to the optimum. Notice that for all of the four pivot rules we introduced, each vertex has it unique tableau and the choice of the entering and leaving variables depend only on the current tableau. This is not always the case. In fact, for many other pivot rules, such as the history-based pivot rules, the same tableau might yield different choices of entering and leaving variables. Nonetheless, this property is essential for us to draw the spanning tree.

DEFINITION 17. An **undirected graph** is graph, i.e., a set of vertices (or nodes) that are connected together, where all the edges are bidirectional.

The second graph in Figure 3 (with both black and red eges) is an example of an undirected graph.

DEFINITION 18. A **tree** is an undirected graph in which any two vertices are connected by exactly one path. Every acyclic connected graph is a tree, and vice versa.

DEFINITION 19. A **subgraph** S of a graph G is a graph whose set of vertices and set of edges are all subsets of G.

DEFINITION 20. A **spanning tree** T of an undirected graph G is a subgraph that is a tree which includes all of the vertices of G, with minimum possible number of edges.

The third graph in Figure 3 is an example of a spanning tree .

DEFINITION 21. An **oriented tree** is a directed acyclic graph whose underlying undirected graph is a tree.

The last graph in Figure 3 is an example of an oriented tree.

FIGURE 3. Complete path for the Dantzig pivot rule

Consider the last graph in Figure 3. The **root** is defined as the top node of the tree. The **child** is the node directly connected to another node when moving away from the root. The **parent** is the converse notion of a child. The **external node** (or the leaf) is a node with no children. The **internal node** is the node with at least one child. For a given node, the **degree** is its number of children. A leaf is necessarily degree zero. A **path** is sequence of nodes and edges connecting a node with a descendant. The **level** of a node is defined as: 1 + the number of edges between the node and the root. The **depth** of a node is defined as: the number of edges between the node and the root. The **height of a node** is the number of edges on the longest path between that node and a leaf. The **height of a tree** is the height of its root node [8].

Because of the uniqueness of the tableau for each vertex, each vertex has only one parent node. Therefore, the graph containing all path is acyclic and hence can form a oriented spanning tree, with vertices with less reduced cost in the objective function pointing toward vertices with improved value of the objective function. The process is shown in Figure 3.

CHAPTER 2

Visualizing the Four Pivot Rule Trees

In this section, we present the set-up of our experiments and the usage of the package for examining behaviors of the four pivot rules.

2.1. Experiment Set-Up

From Chapter 1 we known that given a non-degenerate LP and one of the four pivot rules, its complete paths from all vertices to the optimum vertex can be drawn as an oriented spanning tree. In our experiments, we:

- randomly generate 500 cone-type polytopes from dimension three to seven
- compute their complete paths and converse them into oriented spanning trees
- extract various features from the spanning trees and analyze the result data

The aim of this experiment is to observe patterns from the performance of the pivot rules, and look for the possible improvement of current simplex pivot rules.

2.1.1. the Cone-Type Polytope.

DEFINITION 22. A **spindle** is a polytope with two special vertices u, v such that every facet contains u or v but not both.

For example, the vertex 3 and 8 in Figure 1 do not have any facet in common.

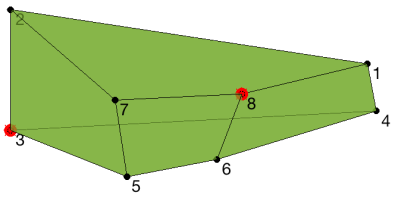


FIGURE 1. Vertices that do not have common facet

The examined polytopes satisfy the above property, with the two special vertices $(0, 0, \cdots, 0)$ and $(1, 1, \cdots, 1)$. The polytopes are characterized by $2n$ half-spaces, with

n of them $x_i \geq 0$ for $i = 1, \cdots, n$, which contains the oringin. The other n half-spaces are randomly generated to satisfy the inequalities:

$$\sum_{i=1}^{n} a_i(x_i - 1) \leq 0$$
$$\Leftrightarrow \sum_{i=1}^{n} a_i x_i \leq \sum_{i=1}^{n} a_i$$

The result matrix will be of the form:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n_1} & a_{n1} & \vdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} \sum_{j=1}^{n} a_{1j} \\ \sum_{j=1}^{n} a_{2j} \\ \\ \sum_{j=1}^{n} a_{nj} \end{bmatrix}, \ x_1, \cdots, x_n \geq 0$$

where in order to avoid repetitive constraints, the matrix $A = (a_{ij})_{n \times n}$ is forced to be nonsingular. The generated polytopes are simple.

2.1.2. CDD/CDD+. We know that each convex polytope can be represented either as the intersection of a class of half-spaces, or as the convex hull that contains a set of points. In order to examine the behaviors of the four pivot rules, the convex hull expression is needed. Hence, the software package *CDD/CDD+* is utilized, with input the class of half-spaces that generate the LP and output the corresponding vertices, edges, and adjacency vertices of each vertex.

2.1.3. Examine the Pivot Rules. Since the cone-type polytopes are simple, by the uniqueness of the tableau, once we know a path we simultaneously know the path of all vertices on the path. Therefore, it is more efficient to start from the vertex x that $z = p^T x$ attains the minimum each time we compute the path. We examine the values of the objective function takes on the set of vertices we get from *CDD/CDD+*, and compute path from the vertex that the has the greatest cost to the optimum, eliminate all vertices on the path from the vertex set, and then compute the path from the vertex with the greatest cost from the remaining set. This way we can compute all paths and then draw their corresponding spanning trees.

2.2. Output Graphs and Tables

2.2.1. Oriented Graphs. The oriented graphs include information of all vertices and edges of the polytope. For any two adjacent vertices, the one that the objective function takes less negative value on points toward the more improved vertex. In addition, the complete paths is highlighted in red, indicating the edges that are actually traveled through. Figure 2 represents the oriented graphs the four pivot rules and variations for the 3-D example we demonstrated in (1.5).

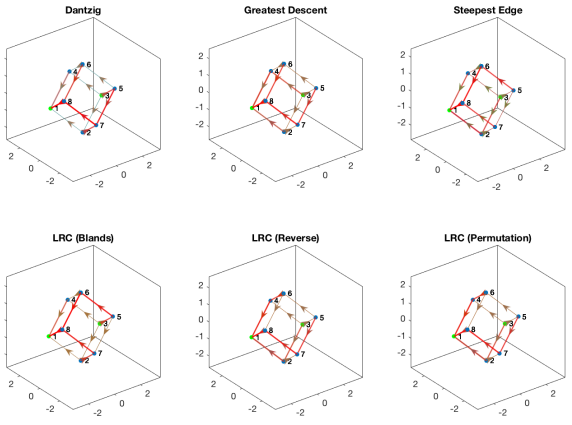


FIGURE 2. Oriented Graphs

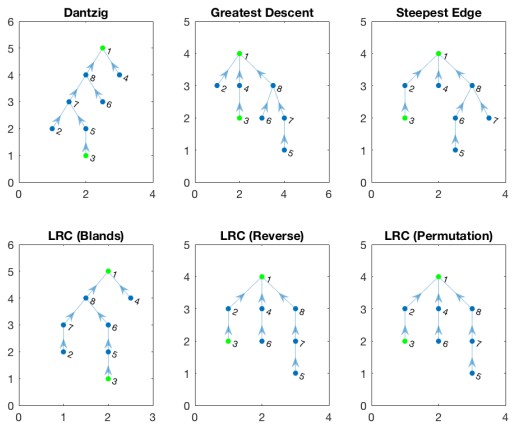


FIGURE 3. Spanning Trees

	breadth	depth	# internal nodes	# levels	width of levels	node degrees
Dantzig	3	4	4	5	1 2 2 2 1	1 0 0 1 0 1 2 2
Greatest Descent	3	3	4	4	1 3 3 1	1 1 0 0 0 1 3 1
Steepest Edge	3	3	4	4	1 3 3 1	1 1 0 0 0 1 3 1
Blands	4	4	3	5	1 2 2 1 2	0 0 0 2 0 1 2 2

FIGURE 4. Output File from the Spanning Tree

2.2.2. Spanning Trees. Spanning trees represent the complete paths of of each pivot rule, with each vertex pointing toward the next one we obtain when applying the pivot rules. In Figure 3, the green nodes indicate the sink and source of the polytope, namely the ones the objective function takes the lowest and the highest values on, respectively.

2.2.3. Extracted Features from the Spanning Trees. With each spanning tree, we examine its breadth, depth, number of internal nodes, number of levels, width of levels, and node degrees (defined in Chapter 1). The output file of the 3-D example we examine in Chapter 1.5 is shown in Figure 4.

Here, the width of levels of Dantzig is 1 2 2 2 1, which means the width of level 1 is 1, the width of level 2 is 2, etc. We read the internal node degrees in the similar fashion. That is, for the Dantzig pivot rule, the degree of node 1 is 1, the degree of node 2 is 0, etc.

2.3. Usage of the Package

This package is mostly written by Maira Hurtado and me during the summer REU at UC Davis, under the guidance of Professor De Loera. Thank you the National Science Foundation for support provided via the NSF grant 1818969 for Prof. De Loera. I would also like to thank Maira for her kind support during her free time after the REU is finished. A detailed version of the package usage is done by Maira, which will be available upon inquiry. Here I demonstrate the basic usage of the it.

```
./newScript.sh dim1 dim2 #expr rules.txt order.txt option.txt
```

For example, if we want to test 500 experiments of polytope from dimension three to seven, the we have dim1 = 3, dim2 = 7, and #expr = 500. The rule.txt indicates the specific pivot rules we want to test, the order.txt contains the Order set that is used for the Leftmost pivot rules, and the option.txt gives us the option to draw the spanning tree and the oriented graph and the options to save the adjacent matrix data and the spanning tree data. In this thesis, we run 500 experiments from dimension three and seven, with three variations of the Bland's rule. One is the trivial order set $\{1, 2, \cdots, n\}$, the second is the reverse order set $\{n, n-1, \cdots, 1\}$, and the last one the a random permutation generated by Matlab. For the output file, we include the spanning tree graphs and data.

CHAPTER 3

Statistical Analysis

In this chapter, we present the result data from the tests. Among the six features extracted from the spanning trees, we analyze the behaviors of breadth, depth, and number of internal nodes with respect to each pivot rule and each dimension.

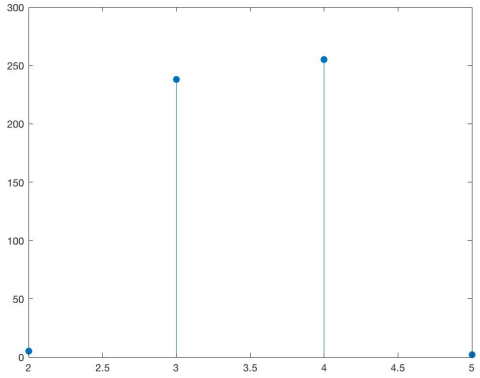
Roughly speaking, breadth represents the number different paths within the spanning tree. The value of breadth equals to the number of nodes that have no child. Under a specific pivot rule, such node can only be improved by traveling to another node but not vice versa.

On the opposite, the number of interior nodes equals to the total number of nodes minus the number of breadth and minus one. Interior nodes represent the nodes that has at least one child node that has worse objective values and has a parent node that it is improved into. An interior node is neither the start point nor the end point of a path.

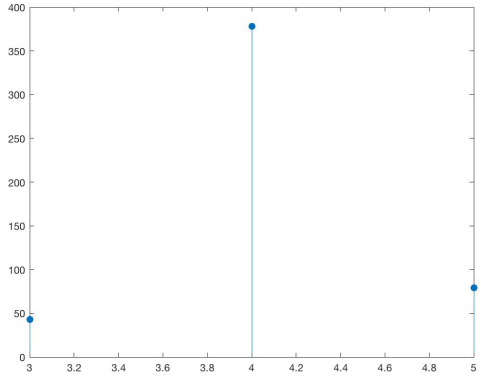
Depths represents the longest path of the spanning tree. It equals to the number of nodes in the longest path minus two, so the depth is the number of interior nodes in the longest path. It indicates the worst case that can happen under a pivot rule.

Begin from the next page, the result data will be presented with respect to the three features from dimension three to seven. The x-axis represents the value of the feature and the y-axis is the number of experiments out of 500 that attain such value. For example, in Figure 2, we see that 180 out of 500 experiments have a breadth of eight under the Dantzig pivot rule, and 200 out of 500 experiments have an breadth of eight under the Greatest Descent pivot rule.

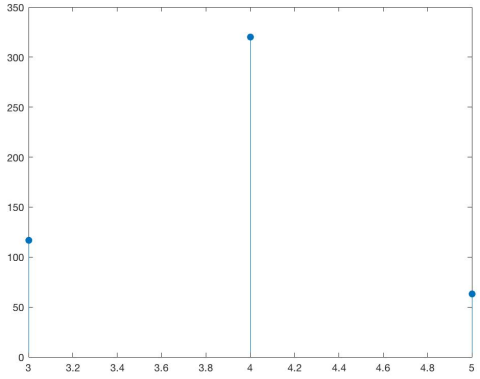
FIGURE 1. Breadth 3-D



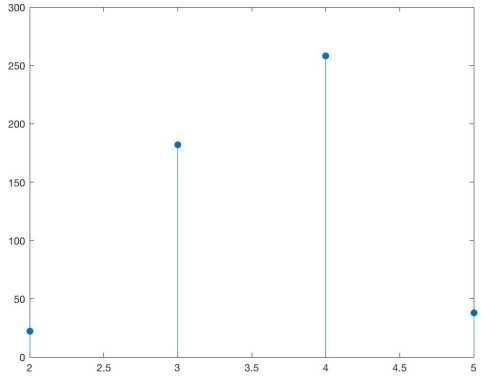
Dantzig



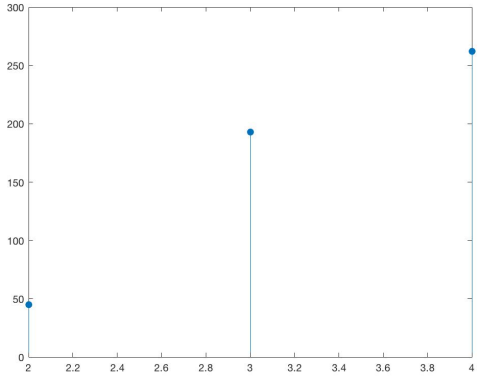
Greatest Descent



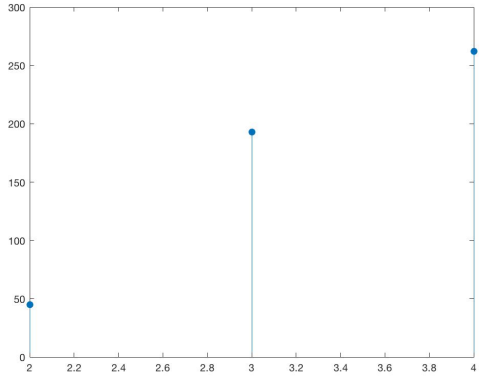
Steepest Edge



Bland's

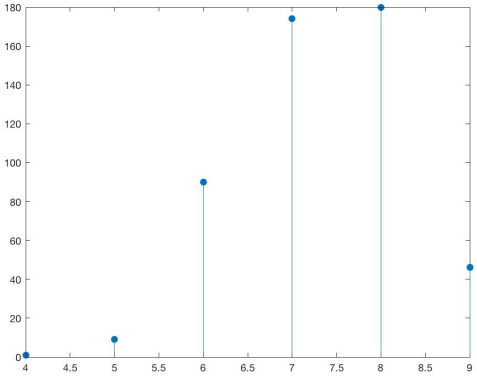


Leftmost-Reverse

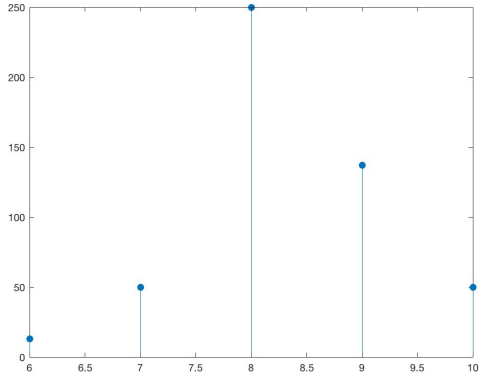


Leftmost-Permutation

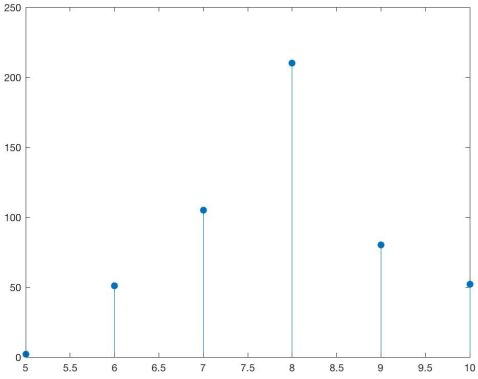
FIGURE 2. Breadth 4-D



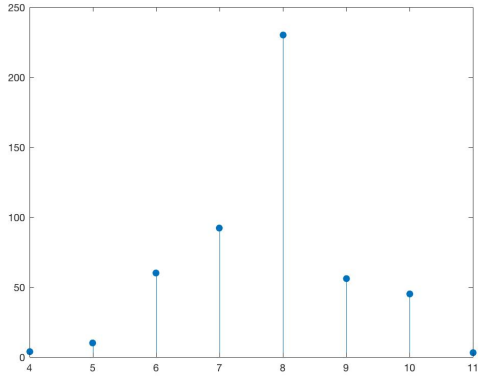
Dantzig



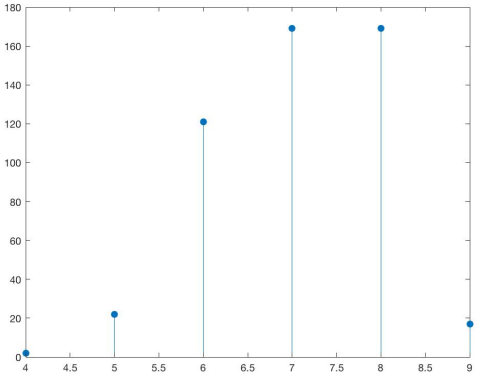
Greatest Descent



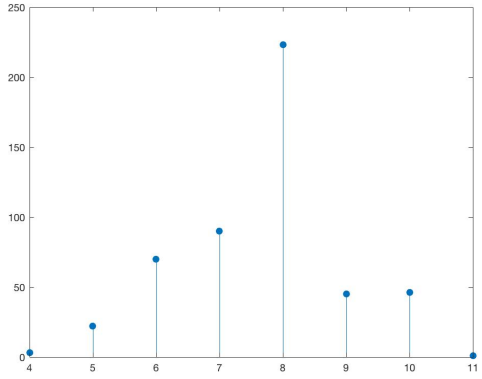
Steepest Edge



Bland's

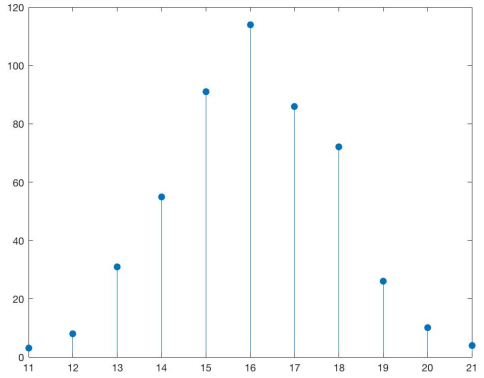


Leftmost-Reverse

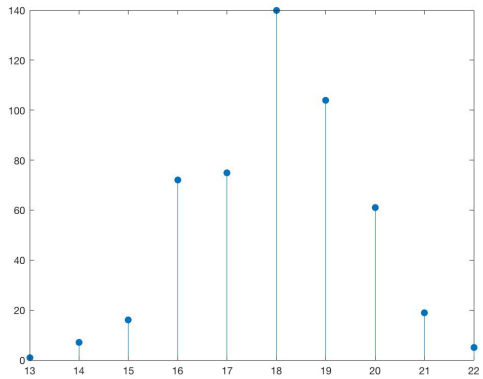


Leftmost-Permutation

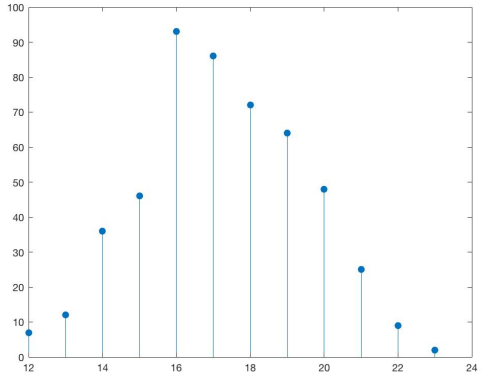
FIGURE 3. Breadth 5-D



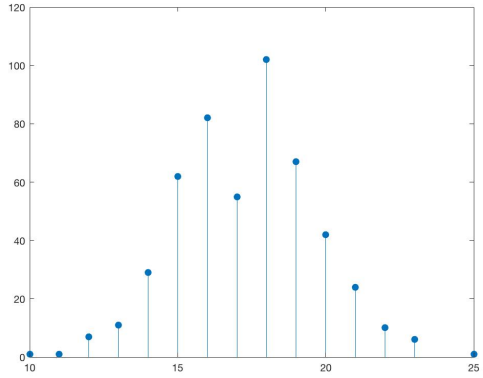
Dantzig



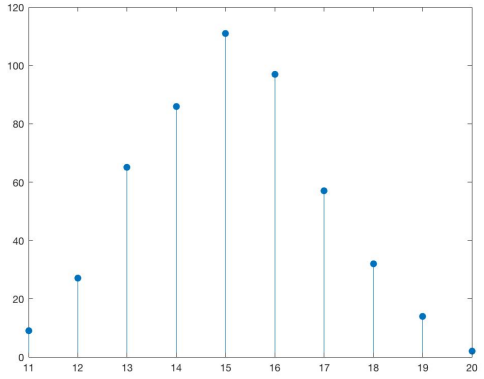
Greatest Descent



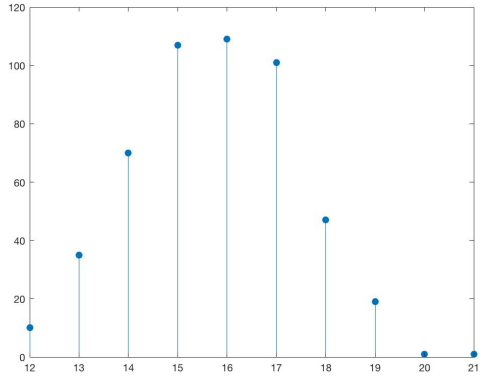
Steepest Edge



Bland's

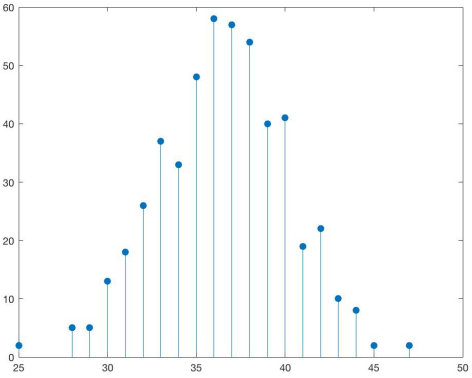


Leftmost-Reverse

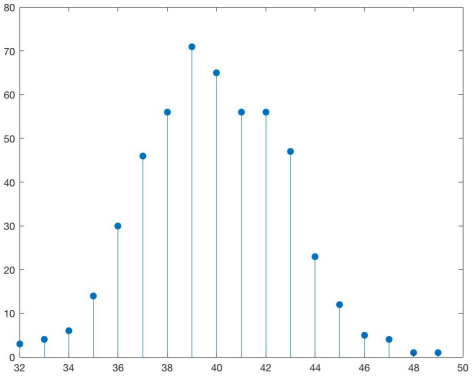


Leftmost-Permutation

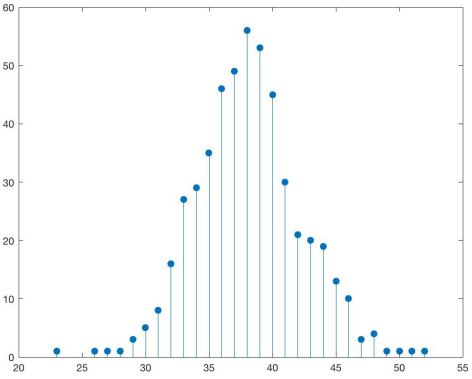
FIGURE 4. Breadth 6-D



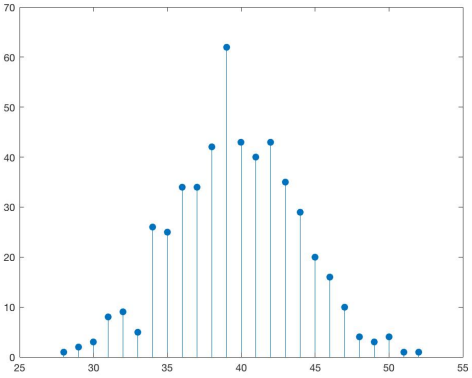
Dantzig



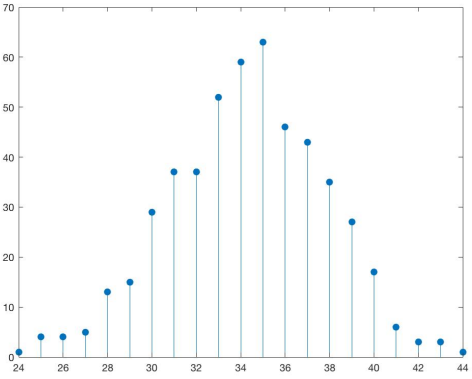
Greatest Descent



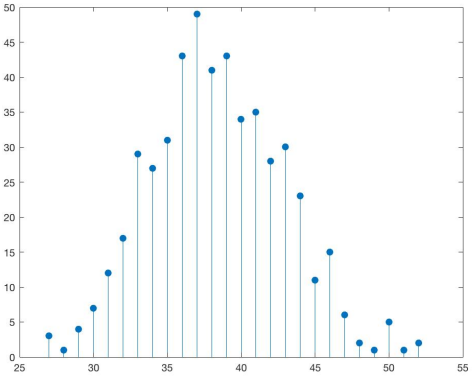
Steepest Edge



Bland's

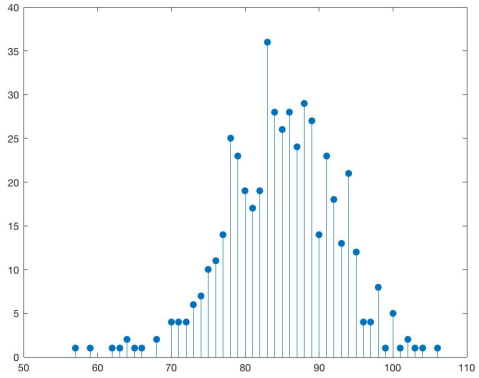


Leftmost-Reverse

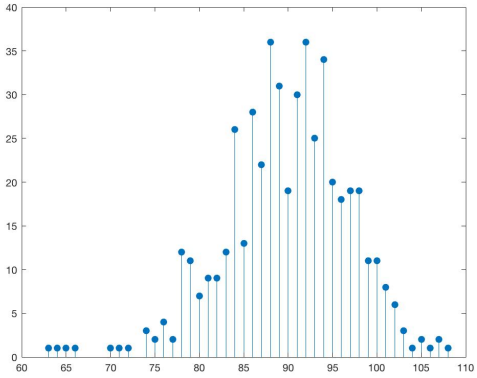


Leftmost-Permutation

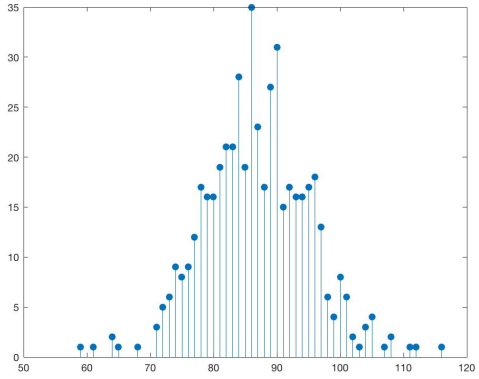
FIGURE 5. Breadth 7-D



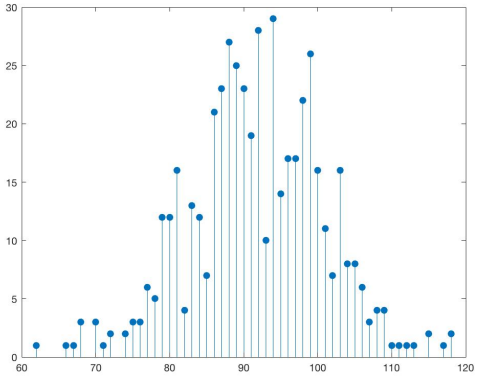
Dantzig



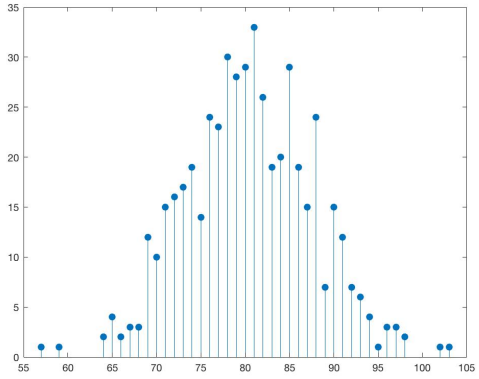
Greatest Descent



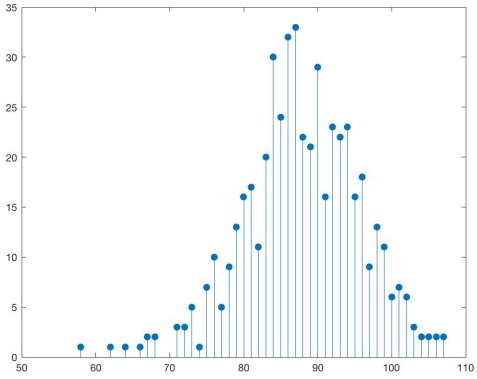
Steepest Edge



Bland's

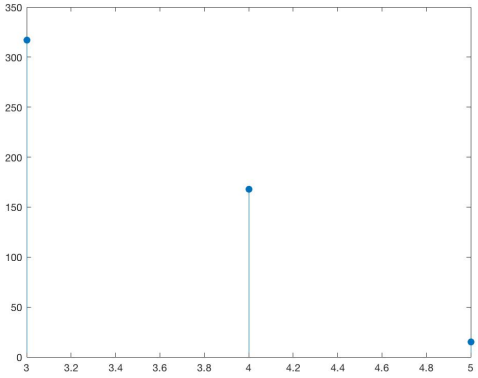


Leftmost-Reverse

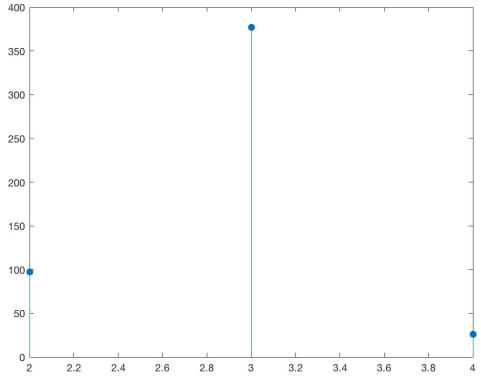


Leftmost-Permutation

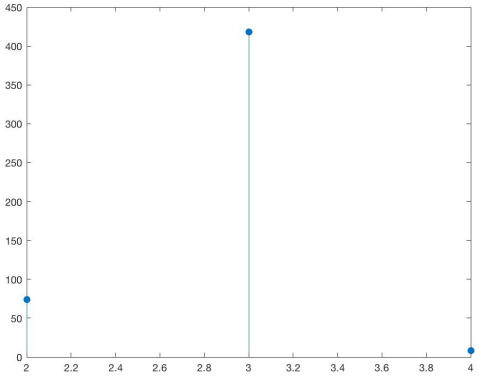
FIGURE 6. Depth 3-D



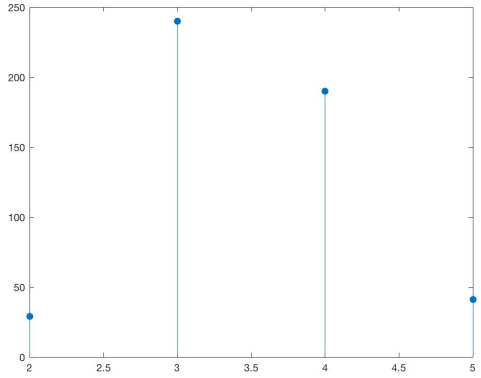
Dantzig



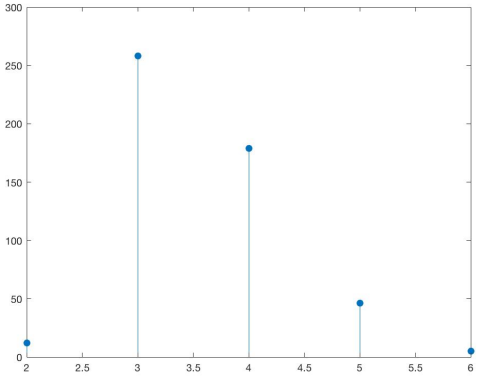
Greatest Descent



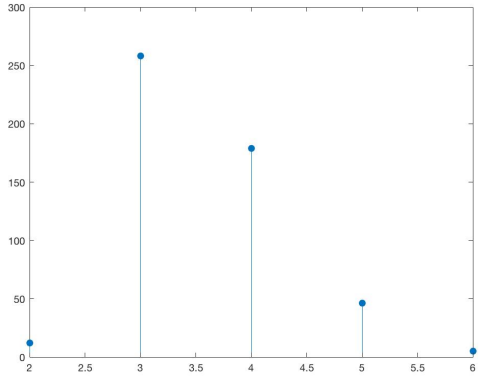
Steepest Edge



Bland's

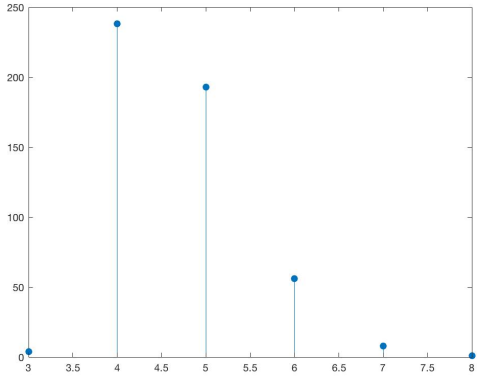


Leftmost-Reverse

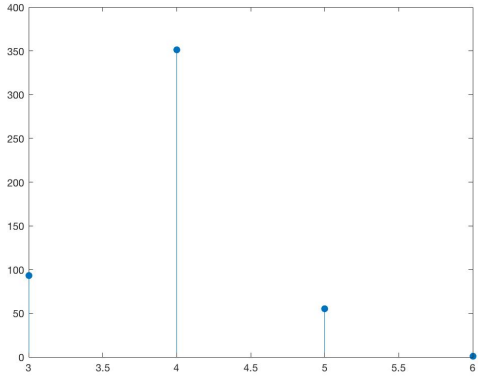


Leftmost-Permutation

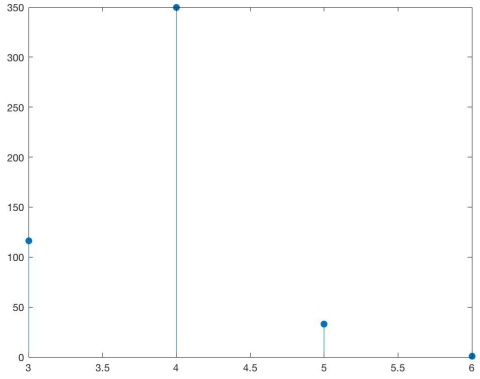
FIGURE 7. Depth 4-D



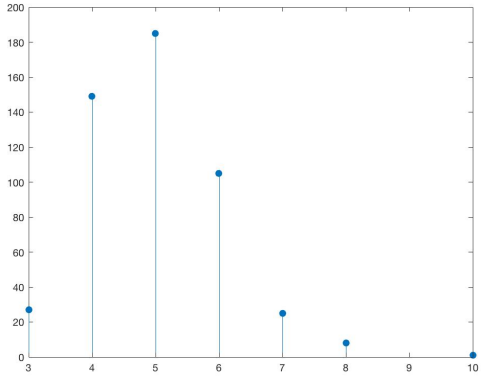
Dantzig



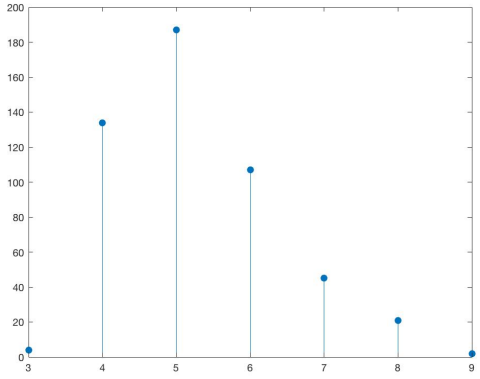
Greatest Descent



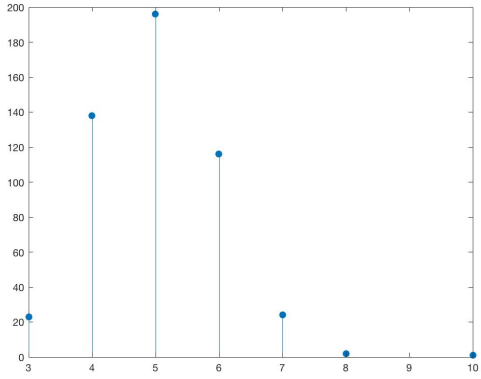
Steepest Edge



Bland's

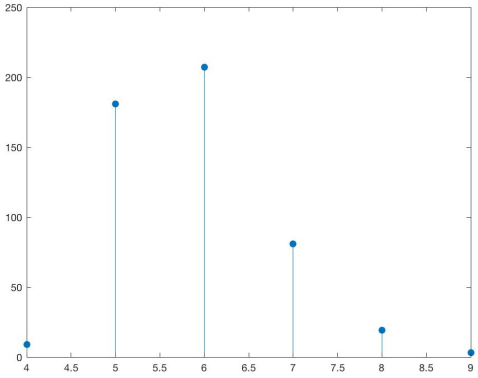


Leftmost-Reverse

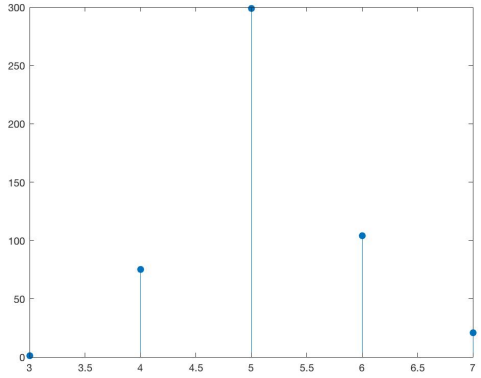


Leftmost-Permutation

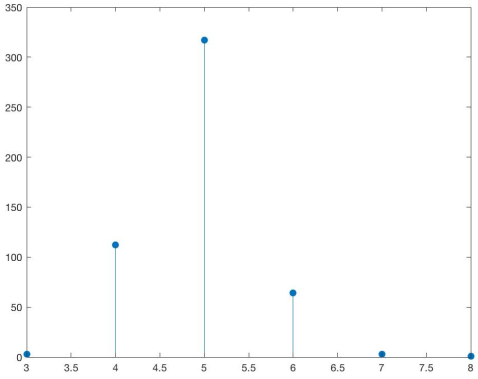
FIGURE 8. Depth 5-D



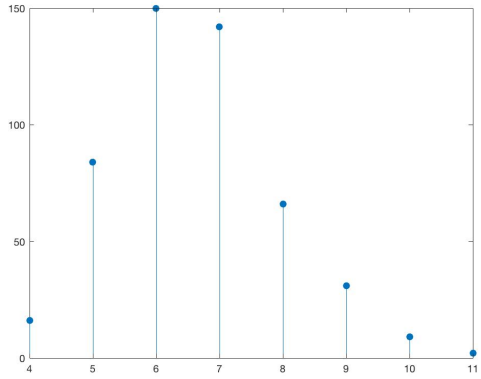
Dantzig



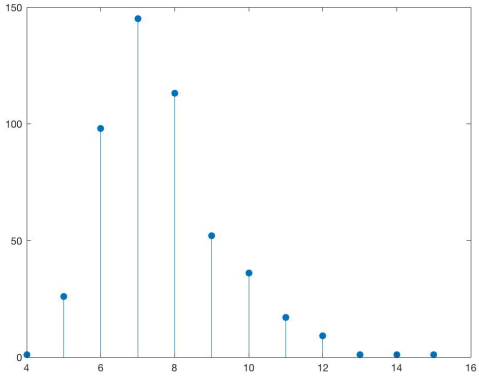
Greatest Descent



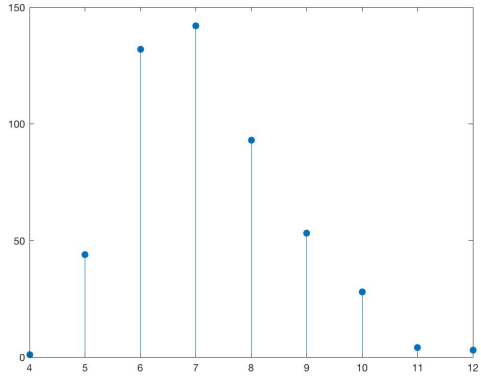
Steepest Edge



Bland's

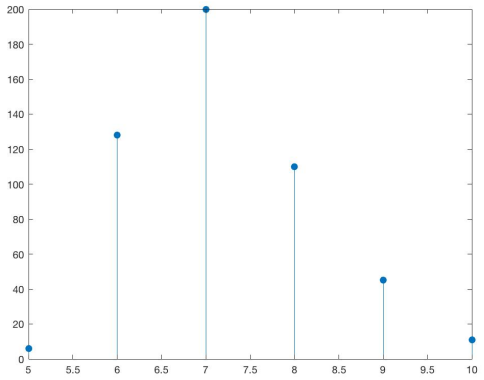


Leftmost-Reverse

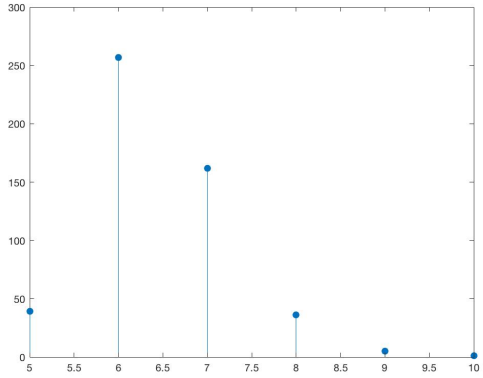


Leftmost-Permutation

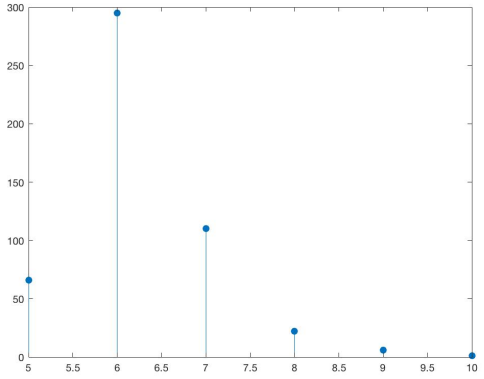
FIGURE 9. Depth 6-D



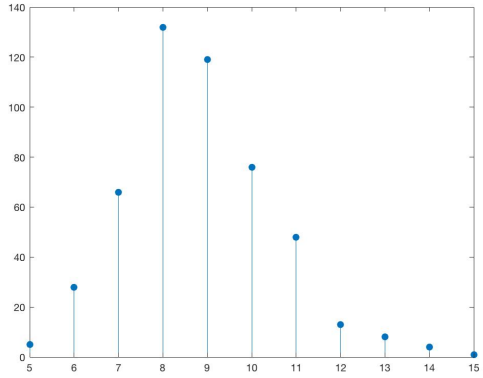
Dantzig



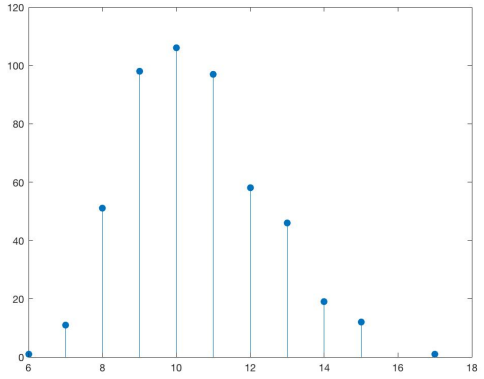
Greatest Descent



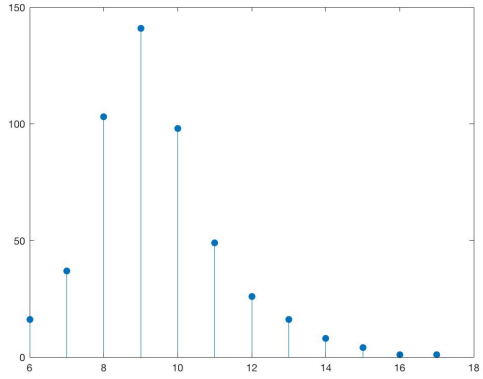
Steepest Edge



Bland's

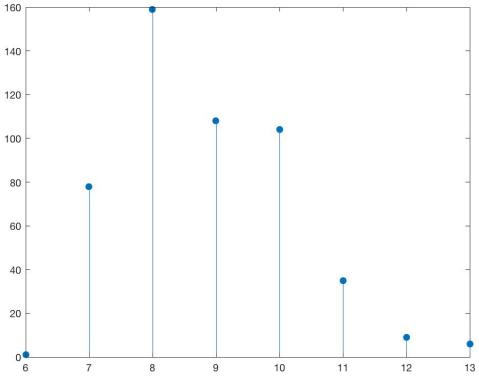


Leftmost-Reverse

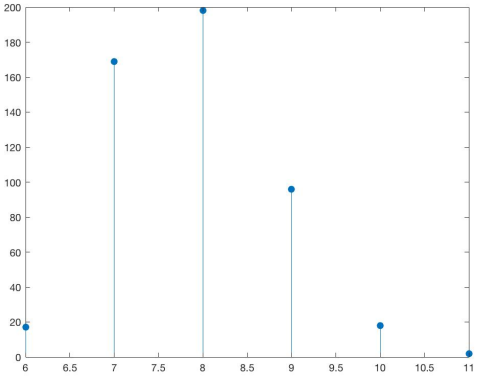


Leftmost-Permutation

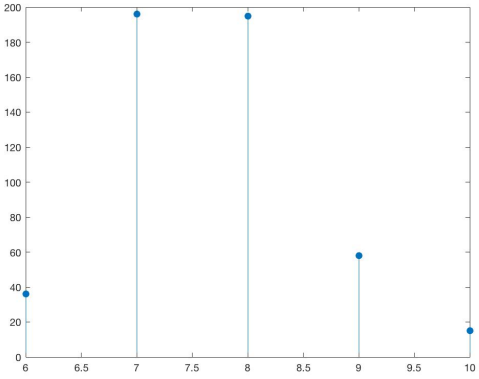
FIGURE 10. Depth 7-D



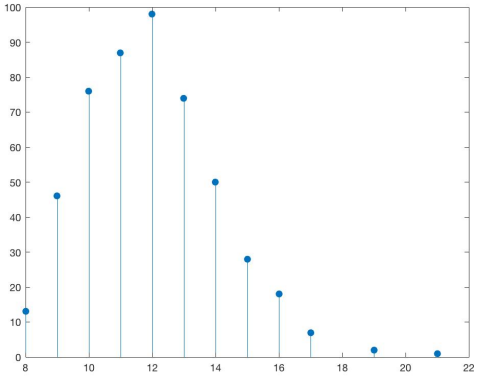
Dantzig



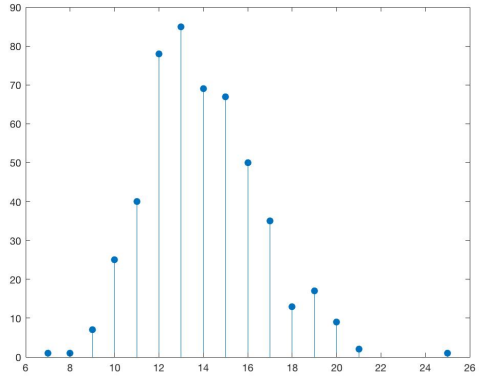
Greatest Descent



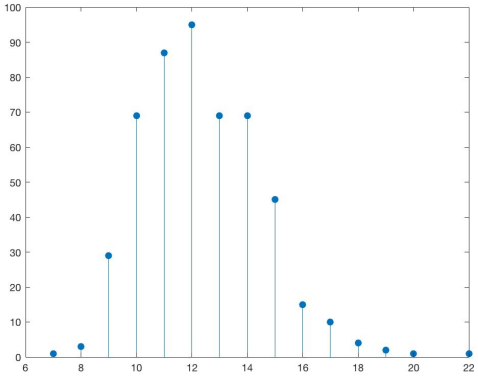
Steepest Edge



Bland's

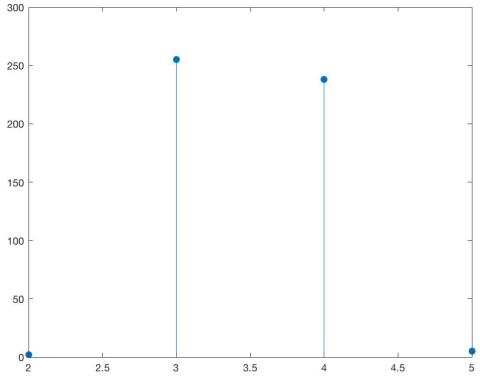


Leftmost-Reverse

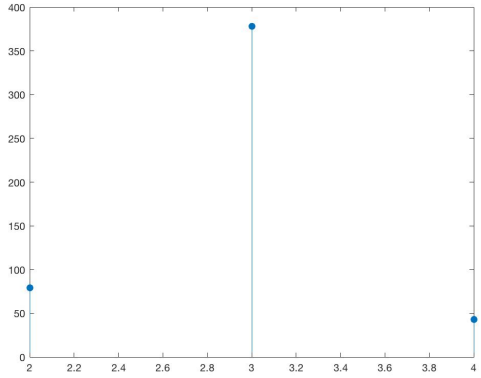


Leftmost-Permutation

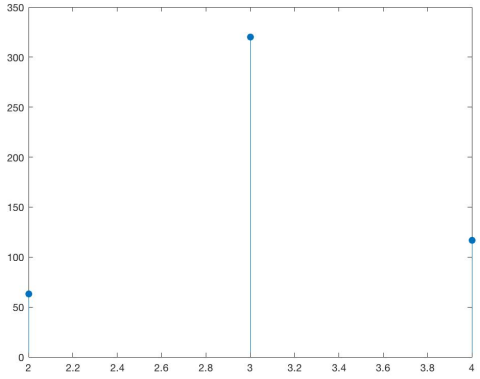
FIGURE 11. Number of Internal Nodes 3-D



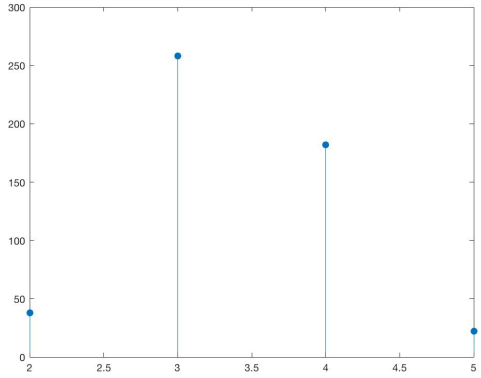
Dantzig



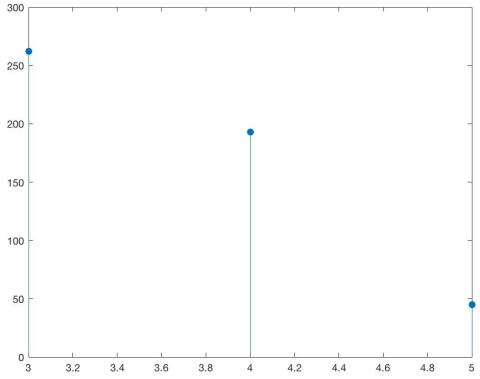
Greatest Descent



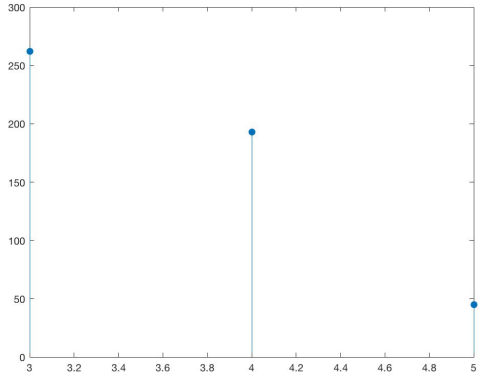
Steepest Edge



Bland's

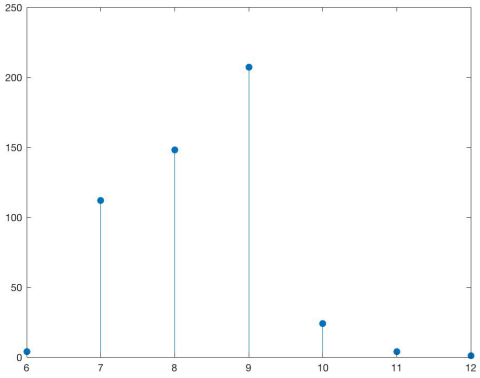


Leftmost-Reverse

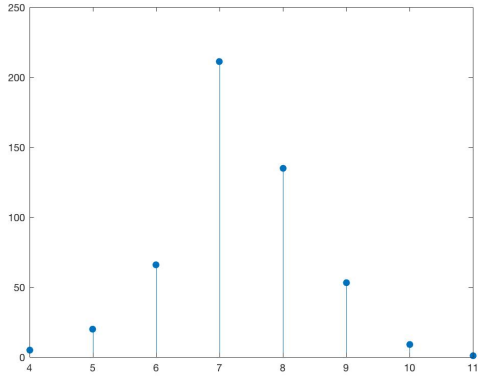


Leftmost-Permutation

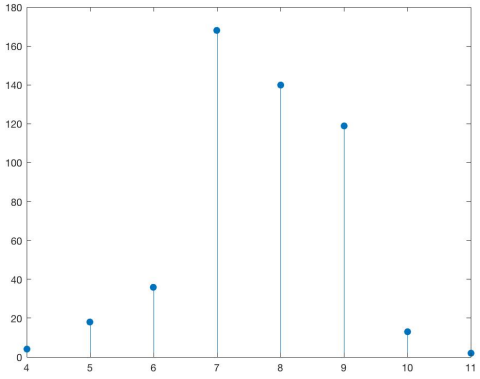
FIGURE 12. Number of Internal Nodes 4-D



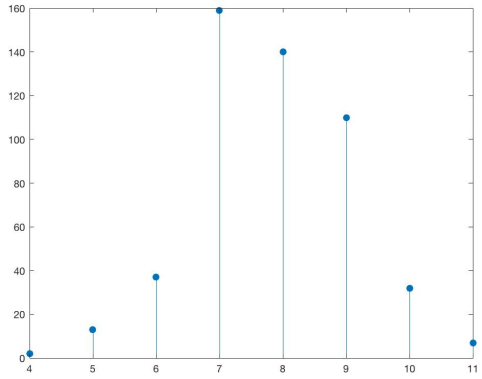
Dantzig



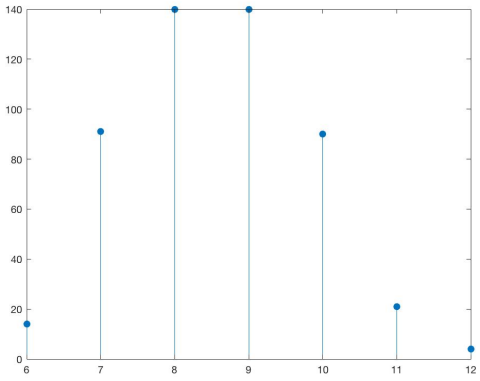
Greatest Descent



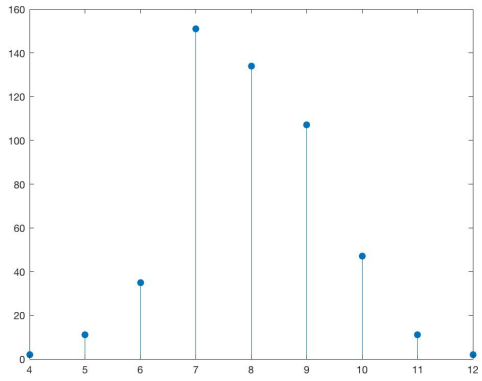
Steepest Edge



Bland's

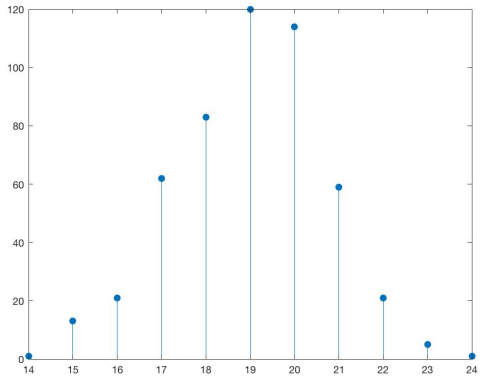


Leftmost-Reverse

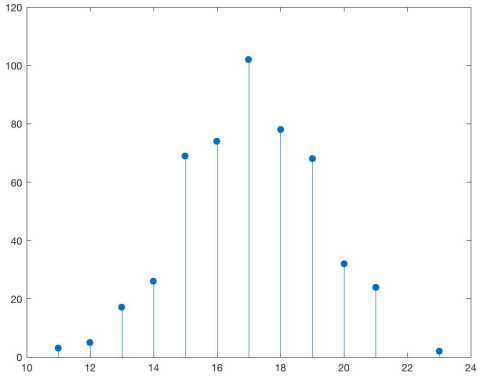


Leftmost-Permutation

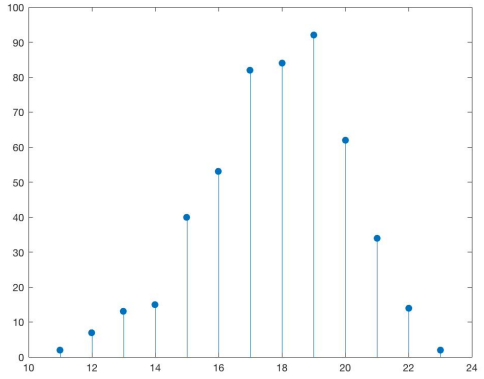
FIGURE 13. Number of Internal Nodes 5-D



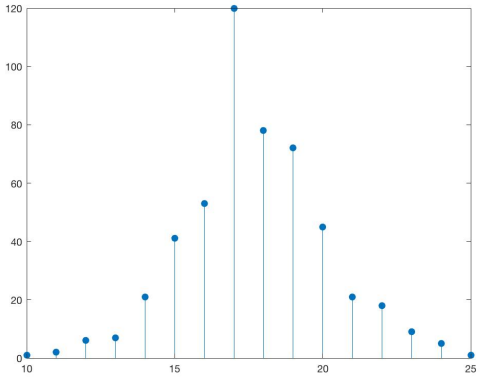
Dantzig



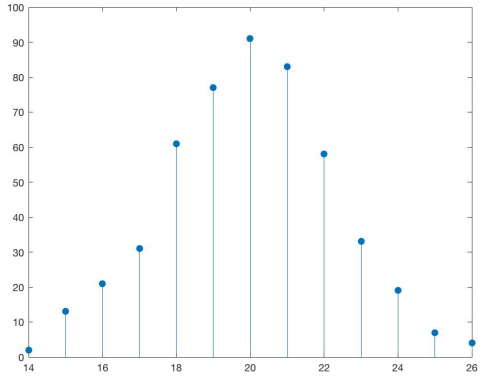
Greatest Descent



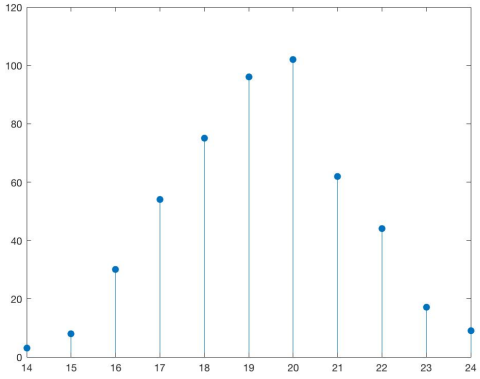
Steepest Edge



Bland's

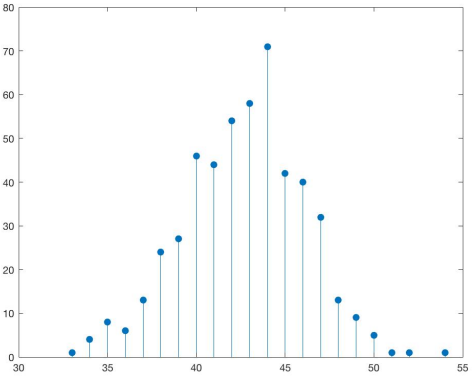


Leftmost-Reverse

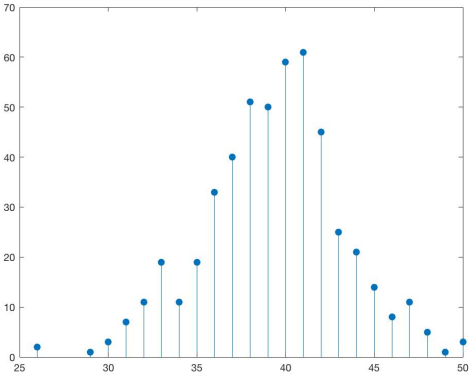


Leftmost-Permutation

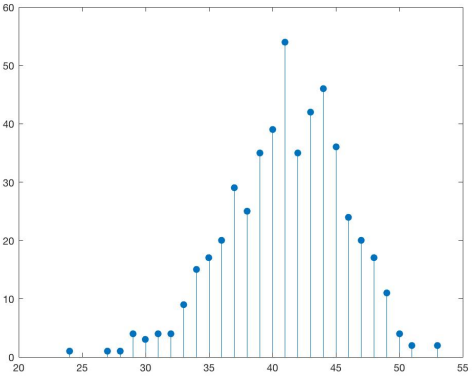
FIGURE 14. Number of Internal Nodes 6-D



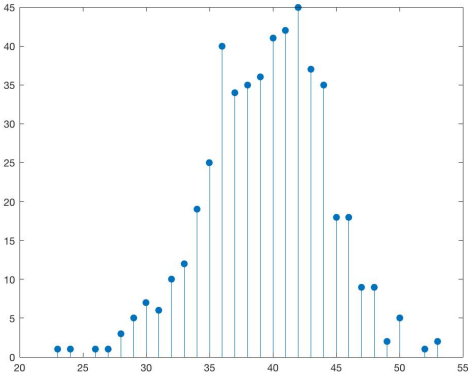
Dantzig



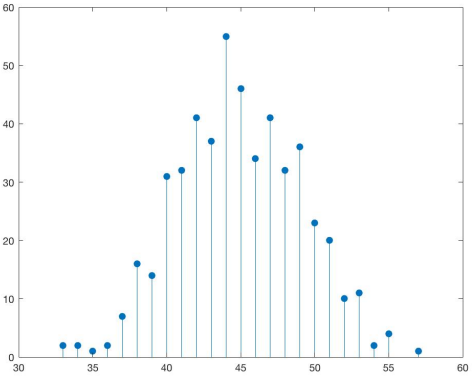
Greatest Descent



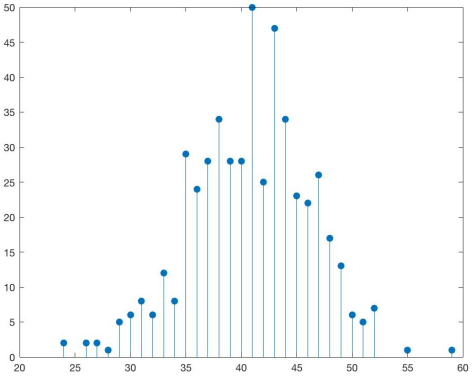
Steepest Edge



Bland's

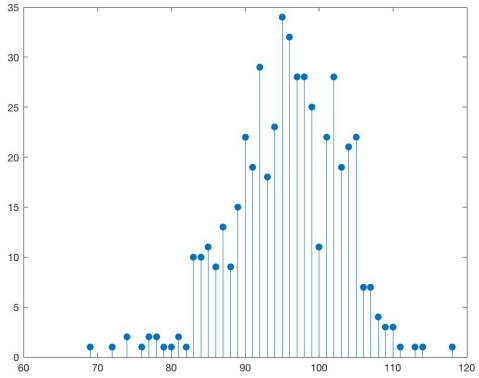


Leftmost-Reverse

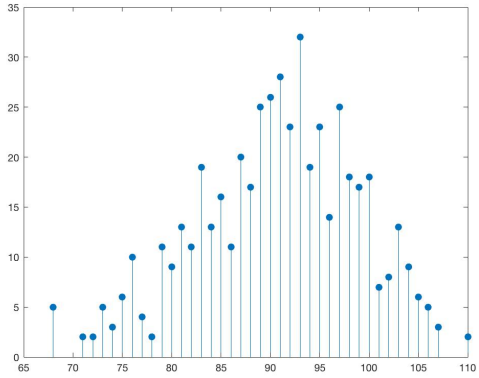


Leftmost-Permutation

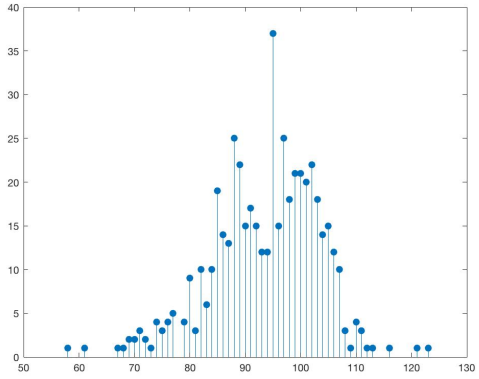
FIGURE 15. Number of Internal Nodes 7-D



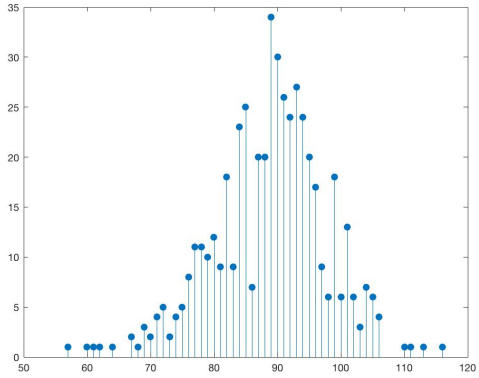
Dantzig



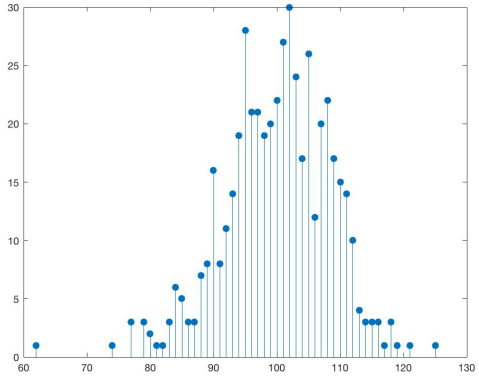
Greatest Descent



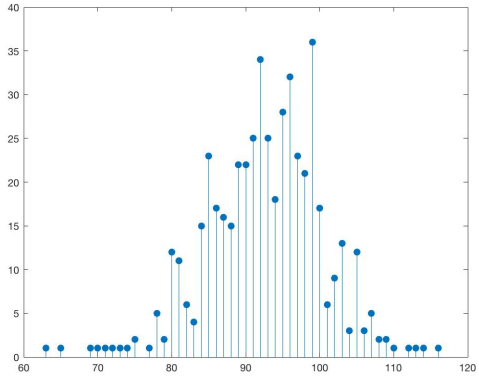
Steepest Edge



Bland's



Leftmost-Reverse



Leftmost-Permutation

Bibliography

1. James Burke, *University of Washington Math 407, Class Notes: Section 5: LP Geometry*, URL: https://sites.math.washington.edu/~burke/crs/407/notes/section5.pdf.
2. Michael C. Ferris, Olvi L. Mangasarian, and Stephen J. Wright, *Linear programming with matlab (mps-siam series on optimization)*, 1 ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.
3. Gil Kalai, *Linear programming, the simplex algorithm and simple polytopes*, Math. Program. **79** (1997), no. 1-3, 217–233.
4. CS 149 Staff, *Brown University CS 149, Notes: Extreme Points, Corners, and Basic Feasible Solutions*, 2008, URL: http://cs.brown.edu/courses/cs149/notes/day5.pdf.
5. Tams Terlaky and Shuzhong Zhang, *Pivot rules for linear programming: A survey on recent theoretical development*, Annals of Operations Research **46-47** (1993), no. 1, 203.
6. Garrett Thomas, *Mathematics for machine learning*, 2017.
7. Robert J. Vanderbei, *Linear programming : foundations and extensions*, 4 ed., International Series in Operations Research Management Science, Springer, 2014.
8. Wikipedia, *Tree (data structure) — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Tree%20(data%20structure)&oldid=887849607, 2019, [Online; accessed 18-March-2019].