The Geometry of the Simplex Method and Applications to the Assignment Problems

By

Rex Cheung

SENIOR THESIS

BACHELOR OF SCIENCE

in

MATHEMATICS

in the

COLLEGE OF LETTERS AND SCIENCE

of the

UNIVERSITY OF CALIFORNIA,

DAVIS

Approved:

Jesús A. De Loera

March 13, 2011

ABSTRACT.
In this senior thesis, we study different applications of linear programming. We first look at different concepts in linear programming. Then we give a study on some properties of polytopes. Afterwards we investigate in the Hirsch Conjecture and explore the properties of Santos' counterexample as well as a perturbation of this prismatoid. We also give a short survey on some attempts in finding more counterexamples using different mathematical tools. Lastly we present an application of linear programming: the assignment problem. We attempt to solve this assignment problem using the Hungarian Algorithm and the Gale-Shapley Algorithm. Along with the algorithms we also give some properties of the Birkhoff Polytope. The results in this thesis are obtained collaboratively with Jonathan Gonzalez, Karla Lanzas, Irene Ramirez, Jing Lin, and Nancy Tafolla.

Contents

CHAPTER 1

Introduction to Linear Programming

1.1. Definitions and Motivations

Optimization is a mathematical way to solve for the best solution to a problem. Its applications arise in many different areas is real life. For example, you want to minimize the labor cost in a business or to maximize the profit of a product. Or in an airline company, you want to minimize the flight time of a plane or the fuel cost. In simple words, in an optimization problem, you want to either minimize or maximize the outcome (called the objective/cost function). Of course, you cannot always optimize a situation without some specific constraints. For example, while producing a product, each material might cost a specific value, and you might only have a limited amount of money to purchase them; or during the production process, it will take an amount of time for a worker to finish a specific part, and each worker will only work for this many hours during his contract. Mathematically speaking, giving a set of constraints, we would like to find the solution in the set that either maximize or minimize a respective cost function.

Since optimization plays such a big role in many industries, finding a method that solves them accurately is one of the biggest research topics in mathematics, namely operational research. Imagine, the simple conditions above can be solved using very basic mathematics, but when it comes to multinational companies that need to make decisions involving many variables, a sight mistake in the mathematics can lead to a loss in millions of dollars.

There are many types of optimization problems. We can have constraints that only involve linear inequalities, which we call it linear programming. This we have quite a few useful and accurate algorithms in which I will slightly touch on them later. If we require the solutions to be integers as well, we have the integer linear programming. This is then very difficult to solve, and till now we still do not have an exact algorithm that can solve them efficiently. In this thesis I will focus on linear programming and some of its useful applications. One of its useful applications comes from the study of convex geometry, when we translate these linear inequalities into a mathematical objects called polytopes. But before I get into it, let me provide a brief example of a linear programming problem.

Here is the general form of a linear program:

$$\begin{aligned} &\max/\min \ \mathbf{z} = \mathbf{c^T x} \\ &\text{subject to } \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \end{aligned}$$

with $c = (c_1, ..., c_n)$, $b = (b_1, ..., b_m)$, $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$.

Example: Suppose your want to make two handcrafts A and B for sell. Both take an hour to make. A costs \$10 to produce and B costs \$20 to produce. You only have 5 hours in your hand and \$60 in your pocket. You are going to sell A for \$20 while B for \$30. How many of each should you make so you can maximize your profit?

We can translate this problem into a linear programming problem (in fact, this is an integer programming problem since you cannot have half a product). Writing them in mathematical notations, we have

$$\begin{aligned} &\max\ 2x_1 + 3x_2 = z \\ &\text{s.t. } x_1 + x_2 \leq 5 \\ &\quad\ x_1 + 2x_2 \leq 6 \end{aligned}$$

There are many ways to solve this problem. We will go for the simplest way now: using the geometric representation of the inequalities.

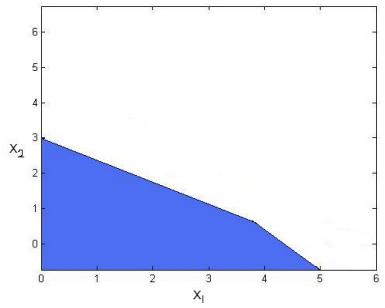


FIGURE 1. The geometric representation of the linear inequalities

First notice that the linear inequalities graph out a polygon. In convex geometry this is called a 2-dimensional polytope. This is our first step towards the study of polytopes, which I will give great detail later. Before I explain how to solve this problem, I will first state a theorem that will automatically tells us the answer, but I will not prove it here, since it will require a bit more background knowledge and that is not the main focus here.

THEOREM 1.1. *The solution to an optimization problem is always a vertex of the polytope represented by the linear inequalities.*

The vertices are (0,3), (4,1) and (5,0). The solution is (4,1) since it maximizes the profit.

1.2. Methods for Solving Linear Programs

We cannot finish off the discussion of linear programming without giving some common methods on solving them. Among all the algorithms that solve linear programs, the Simplex Method, Interior Point Method, Cutting Plane Method, and the Ellipsoid Method are the most commonly used ones. Each of them solves linear programs differently, having its own strength and weakness.

Simplex Method:
For visual learners, the Simplex Method works as follow. First note that the set of inequalities defines a polytope (to be mentioned in details later). We start the Simplex Method by first picking a vertex, then we look at the neighbors of this vertex; if there is a neighbor that gives a better solution with respect to the objective function (max/min), we pivot to that new vertex. We follow this procedure until we reach a vertex that none of its neighboring vertices will give a better solution. There we have reached the optimal solution.

Before I provide the algorithm of the Simplex Method, I will first define a few terms, as well as some equations necessary for the understanding of the algorithm. I will also provide a

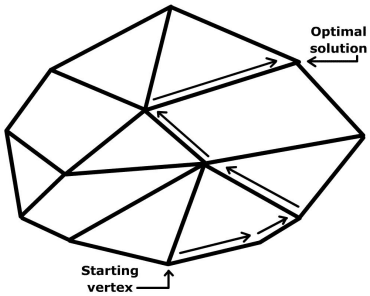


FIGURE 2. A geometric presentation on how the Simplex Method works. (picture from [**sim**])

running example along with the steps of the algorithm.

The Simplex Method solves linear programs written in standard forms, this means that the constraints will be in equalities instead of inequalities. To achieve this we simply add in slack variables to the linear inequalities. These slack variables do not affect the cost function. Below is an example of a linear program and it's standard form.

Example [**IGS09**]**:**

$$\begin{aligned} \min\ z &= x_1 - 2x_2 \\ \text{s.t. } -2x_1 &+ x_2 \leq 2 \\ -x_1 &+ 2x_2 \leq 7 \\ x_1 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Now we add in slack variables to translate the program into standard form:

$$\begin{aligned} \min\ z &= x_1 - 2x_2 \\ \text{s.t. } -2x_1 &+ x_2 + x_3 = 2 \\ -x_1 &+ 2x_2 + x_4 = 7 \\ x_1 &+ x_5 = 3 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

As we can see, x_3, x_4, x_5 are slack variables, and they do not play any role in the objective function z.

Now let us denote the cost function by $z = c^T x$ and the constraints by $Ax = b$, so that we have the following:

$$\begin{aligned} \min\ z &= c^T x \\ \text{s.t. } Ax &= b \\ x &\geq 0 \end{aligned}$$

We denote the basic variables as x_B: these are the variables with a value assigned to it (see example below), and non-basic variables as x_N: these are the variables with a 0 to it. We can then write the objective function as

$$z = c_B^T x_B + c_N^T x_N.$$

We can also write the constraints as

$$Bx_B + Nx_N = b,$$

which can be rewritten as

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

Note that we are setting $x_N = 0$. Substituting this into the objective function and x_B, we get

$$z = c_B^T B^{-1} b \text{ and } x_B = B^{-1}b.$$

The steps of the algorithm are as follow:

(1) Compute $\hat{c}_N^T = c_N^T - c_B^T B^{-1} N$. If $\hat{c}_N^T \geq 0$, then the current basis is optimal, otherwise select x_t such that $c_t < 0$ as the entering variable, where $c_t = min\{c_{t'} < 0\}$.
(2) Compute $\hat{A}_t = B^{-1}A_t$. Find the index s such that
$$\frac{\hat{b}_s}{\hat{a}_{s,t}} = \min_{1 \leq i \leq m}\{\frac{\hat{b}_i}{\hat{a}_{i,t}} : \hat{a}_{i,t} > 0\}.$$
This ratio test determines the leaving variable and the pivot entry $\hat{a_{s,t}}$. If $\hat{a}_{i,t} \leq 0$ for all i, then the problem is unbounded.
(3) Update the basis matrix B and the vector of basic variables x_B.

The following is an example on how we apply the Simplex Method to solve a linear program (taken from Griva, Nash, Sofer [**IGS09**]):

$$\text{A} = \begin{pmatrix} -2 & 1 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ b} = \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} \text{ c} = \begin{pmatrix} -1 \\ -2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

To make things simple, we usually use the slack variables as the initial basis. We have $x_B = (x_3, x_4, x_5)^T$, $x_N = (x_1, x_2)^T$, which means $B = I = B^{-1}$, $c_B^T = (0, 0, 0)$, $c_N^T = (-1, -2)$, and

$$\text{N} = \begin{pmatrix} -2 & 1 \\ -1 & 2 \\ 1 & 0 \end{pmatrix}.$$

Using the formulas above, we have $x_B = \hat{b} = B^{-1}b = (2, 7, 3)^T$ and $\hat{c}_N^T = c_N^T - c_B^T B^{-1} N = (-1, -2)$. Since both components of $\hat{c}_N^T$ are negative, we pick the more negative one, which corresponds to x_2, as the entering variable. From the ratio test, we get

$$\hat{A}_2 = B^{-1}A_2 = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

and

$$\frac{\hat{b_1}}{\hat{a}_{1,2}} = 2 \text{ and } \frac{\hat{b_2}}{\hat{a}_{2,2}} = \frac{7}{2}.$$

The first ratio is smaller, so x_3 is the leaving variable from basis.

In the second iteration we replace x_3 with x_2 in the basis, so x_B is now $(x_2, x_4, x_5)^T$, and $x_N = (x_1, x_3)^T$, and the rest are

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, B^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, N = \begin{pmatrix} -2 & 1 \\ -1 & 0 \\ 1 & 0 \end{pmatrix},$$

with $c_B^T = (-2, 0, 0)$ and $c_N^T = (-1, 0)$. The rest gives

$$x_B = \hat{b} = B^{-1}b = (2, 3, 3)^T,$$
$$\hat{c}_N^T = c_N^T - c_B^T B^{-1} N = (-5, 2).$$

The first component in $\hat{c}_N^T$ is the only negative component, which corresponds to x_1, thus x_1 is the new entering variable. The entering column is

$$\hat{A}_1 = B^{-1}A_1 = \begin{pmatrix} -2 \\ 3 \\ 1 \end{pmatrix}$$

and the ratio test gives $\hat{b_2}/\hat{a}_{2,1} = 1$ and $\hat{b_3}/\hat{a}_{3,1} = 3$, so that x_4 is the leaving variable.
At the third iteration we have $x_B = (x_2, x_1, x_5)^T$, $x_N = (x_3, x_4)^T$, and

$$B = \begin{pmatrix} 1 & -2 & 0 \\ 2 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, B^{-1} = \begin{pmatrix} 1/3 & 2/3 & 0 \\ -2/3 & 1/3 & 0 \\ 2/3 & -1/3 & 1 \end{pmatrix}, N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix},$$

with $c_B^T = (-2, -1, 0)$ and $c_N^T = (0, 0)$, and

$$x_B = \hat{b} = B^{-1}b = (4, 1, 2)^T,$$
$$\hat{c}_N^T = c_N^T - c_B^T B^{-1} N = (-\tfrac{4}{3}, \tfrac{5}{3}).$$

x_3 is the leaving basis in this case. The entering column is

$$\hat{A}_3 = B^{-1}A_3 = \begin{pmatrix} -\frac{1}{3} \\ -\frac{2}{3} \\ \frac{2}{3} \end{pmatrix}$$

and the only ratio we have is $\hat{b_3}/\hat{a_{3,1}} = 3$, thus x_5 is the leaving variable.

At the fourth iteration, we have $x_B = (x_2, x_1, x_3)^T$, $x_N = (x_5, x_4)^T$, and

$$B = \begin{pmatrix} 1 & -2 & 1 \\ 2 & -1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, B^{-1} = \begin{pmatrix} 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \\ 1 & -1/2 & 3/2 \end{pmatrix}, N = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix},$$

with $c_B^T = (-2, -1, 0)$ and $c_N^T = (0, 0)$, and

$$x_B = \hat{b} = B^{-1}b = (5, 3, 3)^T,$$
$$\hat{c}_N^T = c_N^T - c_B^T B^{-1} N = (1, 2).$$

Since all components in $\hat{c}_N^T$ are non-negative, this basis is optimal.

Until now we have only considered the bounded case. But life is not always as good. There are many times we have to deal with unbounded cases, and the algorithm described above will not work properly in those cases.

Example: In step 2 of the algorithm, it is possible that we encounter the unbounded problem. If $\hat{a}_{i,t} \leq 0$ for all i, then none of the basic variables will decrease as x_t increases, implying that x_t can increase without bound. The objective function will then change by any amount equal to $\hat{c}_t x_t$ as x_t increases. The following program illustrates this problem:

$$\begin{aligned} \min\ & z = -x_1 - 2x_2 \\ \text{s.t. } & -x_1 + x_2 \leq 2 \\ & -2x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

After a couple iterations, we get the basis $x_B = (x_1, x_2)^T$ and $x_N = (x_3, x_4)^T$,

$$B = \begin{pmatrix} -1 & 1 \\ -2 & 1 \end{pmatrix} \text{ and } B^{-1} = \begin{pmatrix} 1 & -1 \\ 2 & -1 \end{pmatrix}.$$

Here we have $x_B = (1, 3)^T$ and $\hat{c}_N^T = (5, -3)$, so it is not optimal, and let x_4 be the entering variable. The entering column is

$$\hat{A}_4 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}.$$

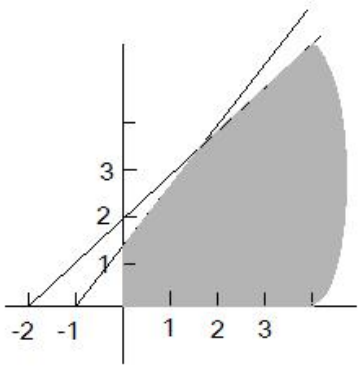


FIGURE 3. Unbounded Linear Program.

Note that both components are less than 0, so there is no candidate. This implies that this problem is unbounded, and x_4 can increase without bound. In this case, we can find a direction of unboundedness and include that into our solution. The feasible solution is $(x_1, x_2, x_3, x_4)^T = (1, 3, 0, 0)^T$. Looking at figure 3 we see that at point $(1, 3)$, the direction $(1, 1)$ is a direction of unboundedness. Thus the solution becomes

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} x_4.$$

Remark: For a $m \times n$ constraint matrix A, the simplex method typically requires at most $2m$ to $3m$ pivots (iterations) to attain optimality.

Remark: The way we travel along edges depends on the pivot rule we use (a pivot rule is a decision rule on how to travel along the polytope). In our example we move to our nearest neighbor which decreases the objective value the most. But there are many other pivot rules out there that are more useful in certain situations: there are the steepest-edge rule, best-neighbor rule, etc. Each rule has its own advantage and is preferred in certain situation. For example, the steepest-edge algorithm takes fewer iterations than other methods, but each iteration takes an excessive amount of time [**Too02**]. Bland's rule solves feasible linear programs without cycling, but it is computationally complex.

Ellipsoid Method:
The ellipsoid method was first developed by David B. Yudin and Arkady Nemirovski [**YN76**] and Naum Shor [**Sho77**] for convex nonlinear programming, and later studied by Leonid Khachiyan for solving linear programming problems with rational data. For problems with n inequalities and integral data with total bit size L, the method generates approximate solution from which an exact solution can easily be obtained in $O(n^2L)$ iterations, requiring $O(n^4L)$ arithmetic operations on numbers with $O(L)$ digits [**Kha80**]. However, the performance is typically very close to its worst-case bound, thus the method is not competitive for linear programming.

The basic idea behind the ellipsoid method can be easily understood. At each iteration an ellipsoid is constructed so that it contains all optimal points. A hyperplane is then constructed such all points lie on one side. Then a new ellipsoid is constructed to contain all the solutions and the correct side. The algorithm (as well as the mathematical formulas) is as follow:

Require: x_0 (a feasible point), A_0 (an initial ellipsoid that contains the minimizers), $\epsilon \geq 0$ (accuracy);
$k \leftarrow 0$;
repeat
$k \leftarrow k + 1$;
evaluate $\phi(x_k)$ and compute any $g_k \in \delta\phi(x_k)$;
$\tilde{g} \leftarrow g_k / \sqrt{g_k^T A_k g_k}$;
$x_{k+1} \leftarrow x_k - A_k \tilde{g}/(n+1)$;
$A_{k+1} \leftarrow \frac{n^2}{n^2-1}(A_k - \frac{2}{n-1} A_k \tilde{g}\tilde{g}^T A_k)$;
until $\sqrt{g_k^T A_k g_k} \leq \epsilon$.

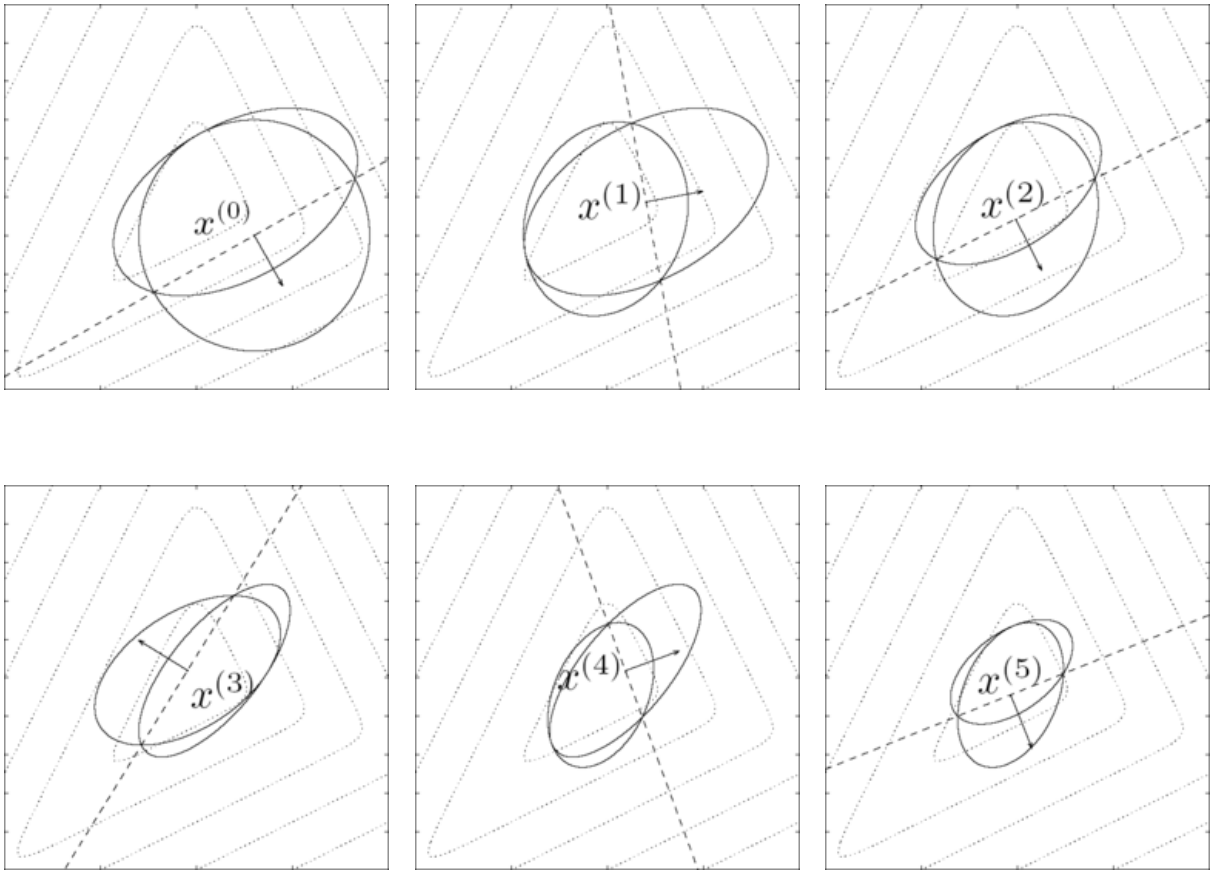


FIGURE 4. 6 steps of the ellipsoid method, from k = 0 to k = 5 [**ell**]

Remark: The Ellipsoid Method is extremely impractical when solving linear programs because of the computation cost at each step. However, its theoretical properties are used in many proofs related to linear programming.

Interior Point Methods (IPM):
Instead of traveling along the edges of the convex region, we can travel within the interior of the region to search for an optimal solution. This is called the interior-point methods. It was first implemented by Narendra Karmarkar in 1984, and the complexity bound is slightly better than the one of the ellipsoid methods. The interior-point method requires $O(n^{3.5}L)$ arithmetic operations on a problem with n inequalities and integer data of total bit length L. However, this method is not very practical: at each iteration a projective transformation is used to bring the current iterate into the center of the feasible region. This transformation is to bring the current iterate to a point far from the constraints so that a steepest descent step will give a good decrease. The following theorem is required to show how the interior point methods work.

THEOREM 1.2. *The optimal solutions of primal and dual programs x^* and y^* must be the solutions of the following system of equations and inequalities:*

- $Ax = b$
- $A^T y - s = c$
- $x_i s_i = 0$
- $x_i, s_i \geq 0$

The IPM works by relaxing the constraint $x_i s_i = 0$, and this will not satisfied until the very end. The set of equations and inequalities to solve now becomes

- $Ax = b$
- $A^T y - s = c$
- $x_i s_i = \lambda, where \lambda > 0$
- $x_i, s_i \geq 0$

We start off at the *analytic center* by setting $\lambda = \infty$. Then for each λ, we get a solution to the system above, and the set of all real solutions form the *central path.*

THEOREM 1.3. *For all $\lambda > 0$, the system above has a unique real solution $(x^*(\lambda), y^*(\lambda), s^*(\lambda))$ such that*

- *$x^*(\lambda)$ is an optimal solution to $max(c^T + \sum_{i=1}^{n} \lambda log x_i)$ with $Ax = b$ and $x \geq 0$.*
- *The limit point $(x^*(0), y^*(0), s^*(0))$ is an optimal solution of the primal problem.*

Finding the exactly central path requires a lot of computational efforts, thus most researchers use approximations of the curve when dealing with IPM. The exact equation for dimension 2 can be found in [**JADL10**]. For more information on interior-point method, readers can refer to [**Too02**].

1.3. Diameter and Pivot Bound

Diameter is defined as the largest number of edges in a shortest path joining two of its furthest vertices. Let $\triangle(d, n)$ be the largest diameter of a d-dimensional polyhedron with n facets, then this represents the worst number of iterations needed for such polyhedra in a linear program. In 1957, W. M. Hirsch conjectured that $\triangle(d, n) \leq n - d$. In May 2010, Professor Francisco Santos disproved this conjecture using a prismatoid. More about this will be discussed in Chapter II.

The best bounds available now are $\triangle(d, n) \leq 2^{d-3}n$ by Larman [**Lar70**], and $\triangle(d, n) \leq n^{1+logd}$ by Kalai and Kleitman [**KK92**].

We also want to look at the number of iterations needed for any pivot rules. Kalai, Matousek, Sharir, and Welzl found a randomized pivot rule with the expected number of pivots

being $exp(K\sqrt{dlogn})$ [**JMW96**]. Instead of looking at randomized pivot rule, we can look at the expected behavior of a deterministic rule on a random linear programming problem. Borgwardt [**Bor99**] was the first to obtain such polynomial bound. He found the bound to be $O(m^3n^{1/(m-1)})$ for the expected total number of iterations of a dimension by dimension simplex method for the dual problem. On the other hand, Adler and Megiddo [**AM85**], Adler, Karp, and Shamir [**IAS87**], and Todd [**Tod86**] showed a bound of $O(min\{d^2, m^2\})$ on the expected total number of iterations to show infeasibility, show unboundedness, or attain optimality for a lexicographic parametric method.

1.4. Polytopes

Without further ado, let us talk about our main subject of this paper: polytope. First I will describe mathematically what polytopes are, provide some examples, then briefly talk about some special polytopes and related theorems.

A polyhedron is defined by the intersection of a finite set of half space, and a polytope is a bounded polyhedron. This representation of a polytope is called a H-representation. Each half space is defined by a linear inequality, which geometrically represents a hyperplane. We can write the set of linear inequalities as a system of inequalities as follow:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \leq & b_2 \\ \vdots & & & & & & & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & \leq & b_m \end{array}$$

The polyhedron can then be written in the matrix form $\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}$.

We have another way to define a polytope: the convex hull of a set of finite points. A convex combination of a set of points is of the form $\lambda_1x_1+\lambda_2x_2+...+\lambda_nx_n$, where $\lambda_1+\lambda_2+...+\lambda_n = 1$. So the convex hull of a set of points S will be all the possible convex combinations of the points, expressed as

$$conv(S) = \{\textstyle\sum_{k=1}^{n} \lambda_k x_k | x_k \in X, \lambda_k \geq 0, \sum_{k=1}^{n} \lambda_k = 1\}.$$

We call this the V-representation of a polytope. In dimension two, we can visualize this as placing a rubber band over a set of points on the plane. When we release the rubber band, it wraps around the points tightly, and that is the convex hull of the points.

Elements of a polytope have different names. The 0-dimensional elements are called vertices; 1-dimensional elements are called edges; and the $(n-1)$-dimensional elements are called facets. In Chapter II I will give a discussion on the Hirsch Conjecture, which relates to the number of facets of a polytope and it's dimension.

1.4.1. Special Types of Polytopes. There are many different special types of polytopes out there. Each of them has a unique structure. The following are some examples of these polytopes, and some will reappear in the later sections as running examples:

(1) The Platonic Solids.

A platonic solid is a convex polyhedron that is regular, meaning that each facet is of the same shape and size. There are 5 of these platonic solids, namely the tetrahedron, cube, octahedron, dodecahedron, and icosahedron.

THEOREM 1.4. *There are exactly 5 platonic solids.*

FIGURE 5. Dodecahedron, a platonic solid

FIGURE 6. Icosahedron, another platonic solid

(2) Transportation Polytopes.

A $(p \times q)$-transportation polytope is the set of matrices with pre-described integer row and column sums, which can be written as

$$T_{p,q}(a,b) := \{x \in \mathbb{R}^{p\times q} | \sum_j x_{ij} = a_i, \sum_i x_{ij} = b_j, x_{ij} \geq 0\},$$

where a is the vector for row sums and b is the vector for column sums. A special type of transportation polytope is the Birkhoff polytope, where the vertices are permutation matrices. Birkhoff polytopes will be the main character in Chapter III when I discuss about the assignment problem in optimization.

THEOREM 1.5. *(Hurkens) The diameter of any $p \times q$ transportation polytope is at most 3(p+q-1).*

(3) Prismatoid.

A prismatoid is a polytope having two parallel facets that contain all vertices. The diameter of a prismatoid is the number of steps it takes to go from one of these facets to the other. A cube and a prism are simple examples of prismatoids.

Why are we interested in prismatoid? It is because Santos used a specially constructed prismatoid to disprove the Hirsch Conjecture.

(4) Of course there are many more other polytopes, such as the $0-1$ polytopes, which are convex hull of subsets of $\{0,1\}^d$, the vertices of a d-cube. There is also the network-flow polytopes, which is defined by an arbitrary directed graph $G = (V, E)$ with weights given to its vertices. Kim and Santos' survey [**EK10**] provides a detailed discussion on more results of different polytopes.

CHAPTER 2

Hirsch Conjecture and Santos' Prismatoid

2.1. Introduction

In this chapter, we will focus on a famous conjecture in convex geometry: the Hirsch Conjecture. This conjecture was first stated in a letter by Warren M. Hirsch to George B. Dantzig in 1957. For 53 years, many mathematicians have tried to either prove or disproved this conjecture, yet no one has succeeded. In May 2010, Professor Francisco Santos from the University of Cantabria disproved this conjecture, putting an end to this 53 years long run.

CONJECTURE 1. *(The Hirsch Conjecture) For any d-dimensional polytope with n facets, the diameter is no more than $n - d$.*

The conjecture never stated whether the inequality holds for bounded or unbounded polytopes. The unbounded case was disproved by Klee and Walkup [**KW67**] in 1967 with the construction of a polyhedron of dimension 4 with 8 facets and diameter 5. They also proved the following theorem in the same paper:

THEOREM 2.1. *(Klee-Walkup [**KW67**])The following statements are equivalent*

- *Every d-polytope with $d + m$ facets has diameter at most m (The Hirsch Conjecture)*
- *Every m-polytope with 2m facets has diameter at most m (The d-step Conjecture)*

That is, the Hirsch Conjecture holds if and only if the d-step Conjecture holds.

We can look at the Hirsch Conjecture as saying that if we want to go from vertex u to vertex v, we do not expect to have to enter and leave the same facet several times. This implies another conjecture:

CONJECTURE 2. *(The Non-revisiting Conjecture) Let P be a simple polytope. Let u and v be two arbitrary vertices of P. Then, there is a path from u to v which at every step enters a different facet of P.*

And proven by Klee and Walkup, we have

THEOREM 2.2. *(Klee-Walkup [**KW67**]) The Non-revisiting Conjecture holds iff the d-step Conjecture holds.*

DEFINITION 2.3. A d-polytope with n facets is called *Hirsch-sharp* if it meets the Hirsch Conjecture with equality, which means the diameter is exactly $n - d$.

Let us look at some Hirsch-sharp polytopes [**Kim10**]:

- **Cubes.** The d-dimensional cube has $2d$ facets. The distance between any pair of vertices is the number of different coordinates (called the Hamming distance). Thus the diameter is $d = 2d - d$.
- **Products of polytopes.** The product of two Hirsch-sharp polytopes $P \times Q$ is also Hirsch-sharp. To see why this holds, first notice that the dimension and number of facets of $P \times Q$ are the sum of those of P and Q. The diameter of the product is the

sum of the diameters: if we want to go from vertex (u_1, v_1) to vertex (u_2, v_2), we can do this by going from (u_1, v_1) to (u_2, v_1) along $P \times \{v_1\}$, and to (u_2, v_2) along $\{u_2\} \times Q$.

- **Products of simplices.** Any product of simplices of any dimension satisfies the Hirsch Conjecture with equality.

The first non-trivial Hirsch-sharp polytope was constructed by Klee and Walkup [**KW67**] in 1967. This polytope, called Q_4, has 9 facets with diameter 5. The polar $Q_4^{\triangle}$ has the following coordinates:

a := (-3,3,1,2)	e := (3,3,-1,2)
b := (3,-3,1,2)	f := (-3,-3,-1,2)
c := (2,-1,1,3)	g := (-1,-2,-1,3)
d := (-2,1,1,3)	h := (1,2,-1,3)

w := (0,0,0,-2)

Readers can use softwares such as *POLYMAKE* [**pol**] to verify the properties of Q_4 using these coordinates.

Next I will describe two operations one can apply on polytopes. They are called *wedging* and *one-point suspension* (notice that wedging is the dual version of one-point suspension).

Wedging:
Suppose you have a d-polytope P. Pick a facet F. A wedge over F can be thought as stretching all the facets except F into one dimension higher. More formally speaking, let F be a facet of a polytope P, and let $f(x) \leq b$ be the set of inequalities corresponding to F. The wedge of P over F is the polytope

$$W_F(P) = P \times [0, \inf) \cap \{(x, t) \in \mathbb{R}^d \times \mathbb{R} : f(x) + t \leq b\}.$$

LEMMA 2.4. *Let P be a d-polytope with n facets. Let $W_F(P)$ be its wedge on a certain facet F. Then, $W_F(P)$ has dimension $d+1$, $n+1$ facets, and*

$$diam(W_F(P)) \geq diam(P)$$

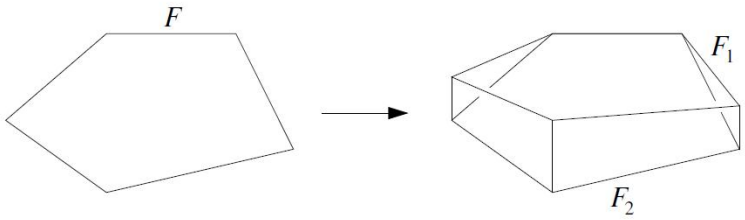


FIGURE 1. A wedge of the 5-gon over F (picture from [**Kim10**]).

One-point suspension:
Pick a vertex v in a d-polytope P. A one-point suspension over v is to take the convex hull of all the original points plus a 'raised' and 'lowered' point of v. In mathematical notation, the one-point suspension of a d-polytope P at vertex v is the polytope

$$S_w(P) = conv((P \times \{0\}) \cup (\{w\} \times \{-1, +1\})) \subset \mathbb{R}^{d+1}.$$

LEMMA 2.5. *Let P be a d-polytope with n vertices. Let $S_w(P)$ be its one-point suspension on a certain vertex w. Then $S_w(P)$ is a (d+1)-dimensional polytope with n+1 vertices, and the diameter of the polar graph $G^{\triangle}(S_w(P))$ of $S_w(P)$ is at least the diameter of the polar graph of P.*

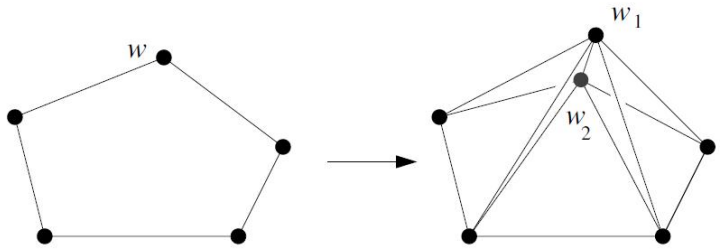


FIGURE 2. One-point suspension of w. w_1 and w_2 are the raised and lowered points of w (picture from [**Kim10**]).

Remark: The wedge of a simple polytope produces a simple polytope, and the one-point suspension of a simplicial polytope is a simplicial polytope.

Professor Santos disproved this conjecture using a special polytope called the prismatoid. In this section, I will briefly discuss some backgrounds about prismatoid, then provide some properties of this counterexample (as well as a perturbed version of this prismatoid), and lastly I will discuss some of the ways we have looked at in order to find more counterexamples to this conjecture.

2.2. Prismatoid and Santos' Counterexample

DEFINITION 2.6. A prismatoid is a polytope Q having two parallel facets Q^+ and Q^- that contain all vertices. We call Q^+ and Q^- the base facets of Q. The width (diameter) of a prismatoid is the minimum number of (dual) steps needed to go from Q^+ to Q^-. [**San10**]

A cube and a prism are examples of prismatoid in dimension 3. We won't consider a pyramid a prismatoid because, even though all vertices are contained within two facets, the top node is not a facet itself. A dual version of a prismatoid is called a spindle.

DEFINITION 2.7. A *d-spindle* is a d-polytope P having two distinguished vertices u and v such that every facet of P contains exactly one of them.

The proof that there exists a counterexample relies on the following theorem:

THEOREM 2.8. *(Generalized d-step Theorem* [**Kim10**]*) If Q is a prismatoid of dimension d with n vertices and width l, then there is another prismatoid Q' of dimension n - d, with 2n - 2d vertices and width at least l + n - 2d. In particular, if $l > d$, then Q' violates the d-step conjecture, and also the Hirsch Conjecture.*

THEOREM 2.9. *There exists a 5-dimensional prismatoid P (with 48 vertices and 322 facets) of width 6.*

Throughout this paper we will call this prismatoid Santos' prismatoid. This prismatoid has the following vertices:

$$\begin{pmatrix} 18 & 0 & 0 & 0 & 1 \\ -18 & 0 & 0 & 0 & 1 \\ 0 & 18 & 0 & 0 & 1 \\ 0 & -18 & 0 & 0 & 1 \\ 0 & 0 & 45 & 0 & 1 \\ 0 & 0 & -45 & 0 & 1 \\ 0 & 0 & 0 & 45 & 1 \\ 0 & 0 & 0 & -45 & 1 \\ 15 & 15 & 0 & 0 & 1 \\ -15 & 15 & 0 & 0 & 1 \\ 15 & -15 & 0 & 0 & 1 \\ -15 & -15 & 0 & 0 & 1 \\ 0 & 0 & 30 & 30 & 1 \\ 0 & 0 & -30 & 30 & 1 \\ 0 & 0 & 30 & -30 & 1 \\ 0 & 0 & -30 & -30 & 1 \\ 0 & 10 & 40 & 0 & 1 \\ 0 & -10 & 40 & 0 & 1 \\ 0 & 10 & -40 & 0 & 1 \\ 0 & -10 & -40 & 0 & 1 \\ 10 & 0 & 0 & 40 & 1 \\ -10 & 0 & 0 & 40 & 1 \\ 10 & 0 & 0 & -40 & 1 \\ -10 & 0 & 0 & -40 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 18 & 0 & -1 \\ 0 & 0 & -18 & 0 & -1 \\ 0 & 0 & 0 & 18 & -1 \\ 0 & 0 & 0 & -18 & -1 \\ 45 & 0 & 0 & 0 & -1 \\ -45 & 0 & 0 & 0 & -1 \\ 0 & 45 & 0 & 0 & -1 \\ 0 & -45 & 0 & 0 & -1 \\ 0 & 0 & 15 & 15 & -1 \\ 0 & 0 & -15 & 15 & -1 \\ 0 & 0 & 15 & -15 & -1 \\ 0 & 0 & -15 & -15 & -1 \\ 30 & 30 & 0 & 0 & -1 \\ -30 & 30 & 0 & 0 & -1 \\ 30 & -30 & 0 & 0 & -1 \\ -30 & -30 & 0 & 0 & -1 \\ 40 & 0 & 10 & 0 & -1 \\ 40 & 0 & -10 & 0 & -1 \\ -40 & 0 & 10 & 0 & -1 \\ -40 & 0 & -10 & 0 & -1 \\ 0 & 40 & 0 & 10 & -1 \\ 0 & 40 & 0 & -10 & -1 \\ 0 & -40 & 0 & 10 & -1 \\ 0 & -40 & 0 & -10 & -1 \end{pmatrix}$$

Remarks:

- The vertices on the left and the vertices on the right span two facets of P, denote as P^+ and P^-. They lie in the hyperplanes $\{x_5 = 1\}$ and $\{x_5 = -1\}$, so P is indeed a prismatoid.
- The matrix on the right is obtained from the one on the left by right multiplication by the following orthogonal matrix:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

- P^+ and P^- are themselves invariant under any of the following 32 orthogonal transformations:

$$\begin{pmatrix} \pm1 & 0 & 0 & 0 & 0 \\ 0 & \pm1 & 0 & 0 & 0 \\ 0 & 0 & \pm1 & 0 & 0 \\ 0 & 0 & 0 & \pm1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & \pm1 & 0 & 0 & 0 \\ \pm1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \pm1 & 0 \\ 0 & 0 & \pm1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Using the Generalized d-step Theorem, we can conclude that there exists a prismatoid of dimension 43 with 86 vertices and width 44.

Remark: We can only prove the existence of this prismatoid using the Generalized d-step Theorem. We cannot find the explicit coordinates of the 86-dimension prismatoid yet because

at each operation (one-point suspension) we are more or less doubling the total number of vertices and at the same time multiplies four or five times the computation time. Julian Pfeifle, a mathematician that helped verify Santos' prismatoid, approximated that the final prismatoid will have about 2^{40} vertices.

The d-step property says that for any prismatoid, the distance between the two base facets is at most d, and if a prismatoid violates this property, this implies there exist a prismatoid of equal or higher dimension that violates the Hirsch Conjecture. Santos provided two proofs to show that his prismatoid violates the d-step property: one involves looking at the inequalities of the facets, while the other involves the study of geodesic maps. The second proof also gives an insight on how Santos' prismatoid is being constructed.

2.3. Properties of Santos' Prismatoid

What makes Santos' prismatoid so special? Does its graph contain any special features? Is there a combinatorial pattern in its F-vector? In this section we will explore the graph of Santos' prismatoid. Also, as we will see later, not all facets of this prismatoid are simplices, so we perturbed this prismatoid slightly and look at the 'simple' version as well.

The Hirsch Conjecture suggests that we either travel from vertex to vertex or from facet to facet when counting the diameter (width). Santos' prismatoid comes in vertex notation, this implies that when counting the distance, we will be traveling along facets, which also means that we are interested in the connectivity of the facets more than the vertices. For this purpose, we take the polar of Santos' prismatoid and look at its properties.

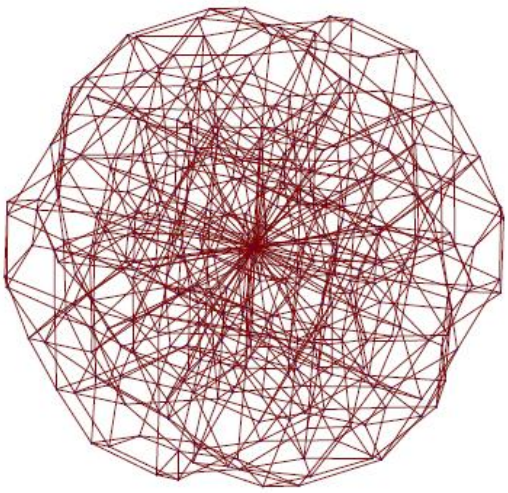
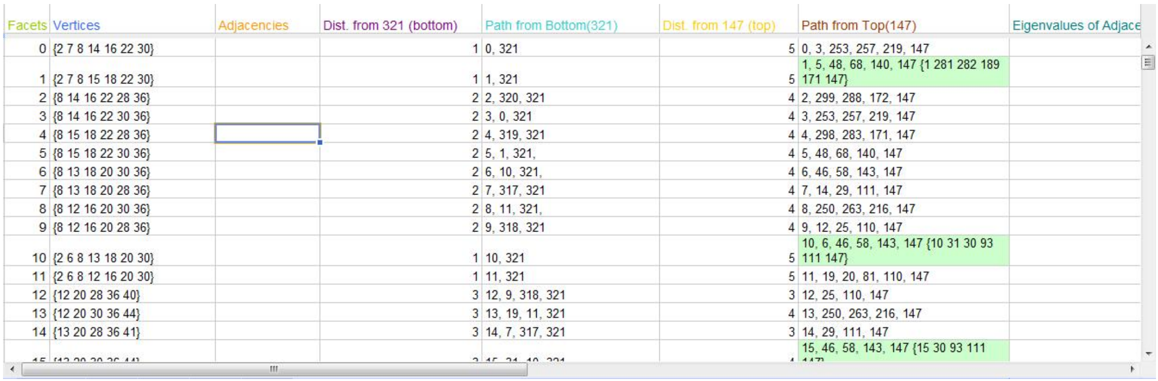


FIGURE 3. The graph of Santos' spindle

We cannot draw out the 5-dimensional spindle, thus we draw out the graph instead. Fig 3 shows the graph of this 5-spindle. We can see that the center of the graph is packed with lines connecting to one vertex. In the dual setting, this vertex corresponds to either the facet Q^+ or Q^-. We can also see that this graph is symmetric along a diagonal line slanted at about 23 degree.

Before continuing, we should label the vertices of this graph to make calculation easier. `https://spreadsheets.google.com/ccc?key=0Apy5kusjeMBkdE11Ni1rVDl4T3ZWWDZYNkpXZVFuc2c&hl=en` contains a spreadsheet of facets and paths, along with some other information on Santos' prismatoid. The vertices in the second column are labeled in the order shown in the above matrix starting at position 0. The facet Q^+ is numbered as 147 and the facet Q^- is numbered as 321. The table also contains the eigenvalues of the adjacency matrix (with respect to the facets) as well as the eigenvalues of the Laplacian matrix. These eigenvalues were used to calculate chromatic numbers.



Facets	Vertices	Adjacencies	Dist. from 321 (bottom)	Path from Bottom(321)	Dist. from 147 (top)	Path from Top(147)	Eigenvalues of Adjace
0	{2 7 8 14 16 22 30}		1	0, 321	5	0, 3, 253, 257, 219, 147	
1	{2 7 8 15 18 22 30}		1	1, 321	5	1, 5, 48, 68, 140, 147 {1 281 282 189 171 147}	
2	{8 14 16 22 28 36}		2	2, 320, 321	4	2, 299, 288, 172, 147	
3	{8 14 16 22 30 36}		2	3, 0, 321	4	3, 253, 257, 219, 147	
4	{8 15 18 22 28 36}		2	4, 319, 321	4	4, 298, 283, 171, 147	
5	{8 15 18 22 30 36}		2	5, 1, 321,	4	5, 48, 68, 140, 147	
6	{8 13 18 20 30 36}		2	6, 10, 321,	4	6, 46, 58, 143, 147	
7	{8 13 18 20 28 36}		2	7, 317, 321	4	7, 14, 29, 111, 147	
8	{8 12 16 20 30 36}		2	8, 11, 321,	4	8, 250, 263, 216, 147	
9	{8 12 16 20 28 36}		2	9, 318, 321	4	9, 12, 25, 110, 147	
10	{2 6 8 13 18 20 30}		1	10, 321	5	10, 6, 46, 58, 143, 147 {10 31 30 93 111 147}	
11	{2 6 8 12 16 20 30}		1	11, 321	5	11, 19, 20, 81, 110, 147	
12	{12 20 28 36 40}		3	12, 9, 318, 321	3	12, 25, 110, 147	
13	{12 20 30 36 44}		3	13, 19, 11, 321	4	13, 250, 263, 216, 147	
14	{13 20 28 36 41}		3	14, 7, 317, 321	3	14, 29, 111, 147	
[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	15, 46, 58, 143, 147 {15 30 93 111	

FIGURE 4. A screen shot of the webpage.

Sheet 2 and sheet 3 contains all the paths going from Q^- to Q^+. Below each path shows the vertices contained in each respective facet. Readers can refer back to the table in sheet 1 for details.

Now that we have the numberings, we can look at more details. The first thing that came to our mind was: is this prismatoid, excluding the two base facets, simple? This means that each vertex has degree d for a d-polytope. In our case, we want to see if each vertex has degree 5 or not. Using Mathematica we have the following list:

Degree sequence: 32, 32, 6, 5}.

We see that about 1/3 of the vertices (which in the actual prismatoid, facets) have degree 6. This shows that not all the facets are simplices.

Next we looked at the maximal independent edge set and maximal independent vertex set. Recall that an independent vertex set is a set S such that every edge of the graph has at least one endpoint not in S and every vertex not in S has at least one neighbor in S (an independent edge set is defined in the similar way). Please keep in mind that the following numbers are

1 more than the numberings in the spreadsheet. This is because Mathematica's index system starts at 1 while *POLYMAKE*'s index system starts at 0.

Maximal Independent Edge Set: {{1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10}, {11, 12}, {13, 17}, {14, 16}, {15, 29}, {18, 19}, {20, 21}, {22, 23}, {24, 35}, {25, 26}, {27, 28}, {30, 112}, {31, 32}, {33, 34}, {36, 119}, {37, 38}, {39, 40}, {41, 43}, {42, 45}, {44, 46}, {47, 57}, {48, 50}, {49, 65}, {51, 52}, {53, 54}, {55, 63}, {56, 58}, {59, 60}, {61, 62}, {64, 66}, {67, 68}, {69, 70}, {71, 72}, {73, 80}, {74, 87}, {75, 210}, {76, 77}, {78, 81}, {79, 83}, {82, 85}, {84, 86}, {88, 91}, {89, 133}, {90, 132}, {92, 99}, {93, 94}, {95, 96}, {97, 102}, {100, 101}, {103, 104}, {105, 106}, {107, 108}, {109, 110}, {111, 113}, {114, 148}, {115, 116}, {117, 122}, {118, 123}, {120, 121}, {124, 125}, {126, 127}, {128, 129}, {130, 131}, {134, 135}, {136, 137}, {138, 139}, {140, 141}, {142, 145}, {143, 144}, {146, 147}, {149, 150}, {151, 163}, {152, 159}, {153, 156}, {154, 186}, {155, 162}, {157, 158}, {160, 161}, {164, 165}, {167, 168}, {169, 246}, {170, 171}, {172, 173}, {174, 175}, {176, 177}, {178, 179}, {180, 181}, {182, 183}, {184, 189}, {185, 192}, {187, 188}, {190, 191}, {193, 198}, {194, 196}, {195, 202}, {197, 222}, {199, 201}, {200, 203}, {204, 207}, {205, 209}, {206, 297}, {211, 212}, {214, 215}, {216, 218}, {217, 220}, {219, 221}, {223, 225}, {224, 226}, {227, 242}, {228, 229}, {230, 231}, {232, 233}, {234, 235}, {236, 237}, {239, 240}, {241, 245}, {243, 244}, {247, 272}, {248, 249}, {250, 255}, {251, 252}, {253, 254}, {256, 257}, {258, 259}, {260, 261}, {262, 263}, {264, 265}, {266, 267}, {268, 269}, {270, 271}, {273, 274}, {275, 276}, {278, 286}, {279, 280}, {281, 287}, {282, 283}, {284, 285}, {288, 289}, {290, 291}, {292, 300}, {293, 294}, {295, 302}, {296, 301}, {303, 304}, {305, 307}, {306, 308}, {309, 310}, {311, 312}, {313, 315}, {314, 316}, {317, 321}, {318, 319}, {320, 322}}.

Maximal Independent Vertex Set: {1, 3, 6, 8, 9, 11, 13, 16, 19, 20, 22, 27, 30, 35, 37, 40, 41, 44, 50, 51, 53, 56, 59, 62, 65, 66, 71, 73, 75, 76, 82, 83, 86, 88, 94, 97, 98, 99, 100, 104, 107, 113, 116, 117, 118, 119, 124, 127, 136, 140, 145, 149, 153, 159, 161, 163, 166, 168, 170, 173, 175, 176, 178, 180, 184, 188, 190, 192, 193, 194, 195, 200, 204, 205, 208, 214, 216, 217, 221, 226, 228, 231, 232, 236, 240, 242, 243, 246, 248, 252, 254, 261, 265, 267, 272, 278, 279, 282, 284, 288, 290, 302, 304, 310, 314, 320}.

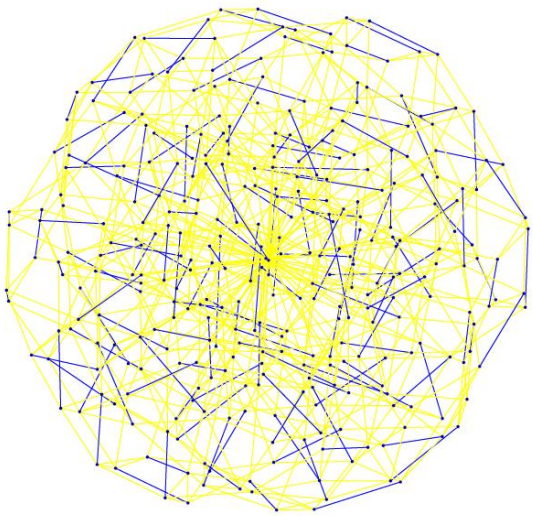


FIGURE 5. Independent Edge Set (blue lines)

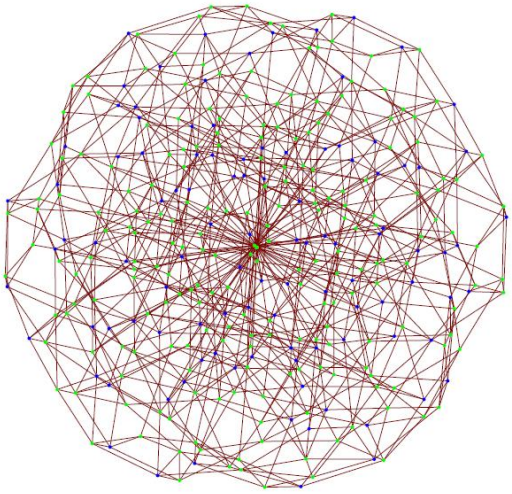


FIGURE 6. Independent Vertex Set (blue dots)

We also calculated the characteristic polynomial using *MATHEMATICA*. To those who are not familiar with linear algebra, a characteristic polynomial of a matrix A is defined by $p_A(t) = det(A - tI)$, where I denotes the n-by-n identity matrix. Due to the fact that this is a polynomial of degree 322, I will only list out the first few terms that are of interest:

Characteristic Polynomial: $\chi = x^{322} - 896x^{320} - 448x^{319} + 394880x^{318} + 388480x^{317} - ...$

From algebraic graph theory, we have the following theorem:

THEOREM 2.10. [**Big74**] *Suppose the characteristic polynomial of a graph G is*

$$\chi(G) = x^n + c_1x^{n-1} + c_2x^{n-2} + c_3x^{n-3} + ... + c_n$$

then we have

- $c_1 = 0$*;*
- $-c_2$ *is the number of edges of* G*;*
- $-c_3$ *is twice the number of triangles in* G*.*

From this theorem, we found that Santos' graph has 896 edges and 224 triangles.

Lastly we looked at the chromatic number for this graph. We cannot calculate the exact number for this graph because of the huge size, thus we find a bound for the chromatic number c.

PROPOSITION 2.11. *The chromatic number c of a graph G is bounded below by the following inequality:*

$$|V(G)| \leq |\textit{maximal independent vertex set}| \times c$$

where $|V(G)|$ denotes the size of the vertex set of G.

Thus c is bounded below by 4. Also, from algebraic graph theory, we have the following theorem:

THEOREM 2.12. *(Wilf 1967* [**Wil67**] *) For any graph G, we have*

$$c \leq 1 + \lambda_{max}(G)$$

where λ_{max} is the largest eigenvalue of the adjacency matrix.

Looking back at our table, $\lambda_{max}(G) = 7.6527$, so c is bounded above by 8. So our bound for c is $4 \leq c \leq 8$.

2.4. A Perturbed Version of Santos' Prismatoid

We like to work with simplices, because they have nice properties that are easy to work with, thus we slightly perturbed Santo's prismatoid. In this perturbed prismatoid, all the facets (except the base facets) are 4-dimensional simplices.

We perturbed the prismatoid in the following manner. The coordinates are as follows: We first multiply each coordinate by 1000 (except for x_5), then we randomly add/subtract 1 to 3 to the multiplied coordinates. We also add/subtract 1 to 3 to some of the 0 coordinates. The coordinates are as follow:

$$
\begin{pmatrix}
18002 & 0 & 0 & 0 & 1 \\
-17999 & 0 & 0 & 0 & 1 \\
0 & 18003 & 0 & 0 & 1 \\
0 & -17997 & 0 & 0 & 1 \\
0 & 0 & 45000 & 0 & 1 \\
0 & 0 & -45003 & 0 & 1 \\
0 & 0 & 0 & 45002 & 1 \\
0 & 0 & 0 & -45000 & 1 \\
15002 & 15000 & 1 & -1 & 1 \\
-15000 & 15002 & -1 & 1 & 1 \\
14998 & -15001 & 0 & 0 & 1 \\
-15002 & -14999 & 0 & 0 & 1 \\
1 & 0 & 30000 & 30000 & 1 \\
0 & -1 & -30000 & 30000 & 1 \\
0 & 0 & 30001 & -29999 & 1 \\
-1 & 0 & -30000 & -30002 & 1 \\
0 & 10002 & 40000 & -1 & 1 \\
0 & -10000 & 40002 & 1 & 1 \\
2 & 10000 & -40000 & -1 & 1 \\
0 & -9998 & -40001 & 0 & 1 \\
10002 & -1 & 3 & 39998 & 1 \\
-9999 & 2 & -1 & 40001 & 1 \\
10000 & 0 & 0 & -40002 & 1 \\
-10001 & 0 & -2 & -40000 & 1
\end{pmatrix}
\begin{pmatrix}
1 & -1 & 17999 & 0 & -1 \\
-2 & -1 & -18001 & 1 & -1 \\
1 & 0 & 0 & 18003 & -1 \\
0 & 0 & 0 & -17998 & -1 \\
45002 & 0 & -1 & 0 & -1 \\
-44998 & 1 & 0 & 1 & -1 \\
-1 & 45002 & 1 & 0 & -1 \\
-2 & -45001 & 0 & 3 & -1 \\
-1 & -1 & 15003 & 14999 & -1 \\
2 & 2 & -14999 & 14998 & -1 \\
1 & -2 & 15002 & -14997 & -1 \\
-1 & 2 & -14998 & -15003 & -1 \\
30000 & 30000 & 0 & 0 & -1 \\
-30000 & 30000 & 0 & 0 & -1 \\
30000 & -30000 & 0 & 0 & -1 \\
-30000 & -30000 & 0 & 0 & -1 \\
40000 & 0 & 10000 & 0 & -1 \\
40000 & 0 & -10000 & 0 & -1 \\
-40000 & 0 & 10000 & 0 & -1 \\
-40000 & 0 & -10000 & 0 & -1 \\
0 & 40000 & 0 & 10000 & -1 \\
0 & 40000 & 0 & -10000 & -1 \\
0 & -40000 & 0 & 10000 & -1 \\
0 & -40000 & 0 & -10000 & -1
\end{pmatrix}
$$

Readers can again use *POLYMAKE* to verify that this perturbed polytope is indeed still a prismatoid and still has width 6.

We see from Figure 7 that the graph is less symmetric now.

Again, we take the dual graph of this prismatoid. To show that all the facets are simplices, we look at the degree sequence:

Degree Sequence: {110, 105, 5}.

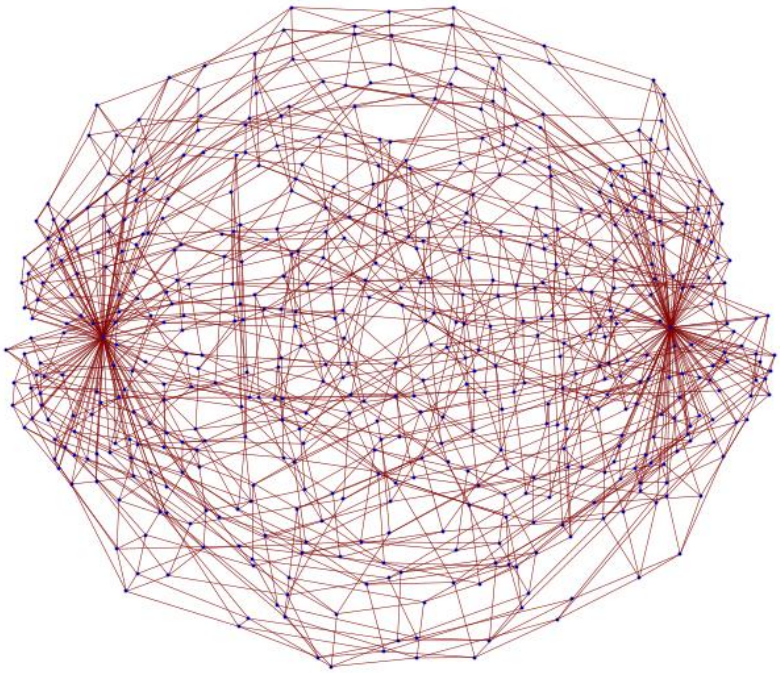


FIGURE 7. Perturbed Santos' Prismatoid

We see every entry is a 5 (except the first two, which are the top and bottom facets). This proves that every facet is a simplex. Next, using the same labeling method we used above (also recorded in sheet 4 of the link), we can obtain a maximal independent edge set and a maximal independent vertex set.

Maximal Independent Edge Set: {{1, 3}, {2, 8}, {4, 5}, {6, 7}, {9, 10}, {11, 12}, {13, 15}, {14, 16}, {17, 18}, {19, 20}, {21, 402}, {22, 23}, {24, 25}, {26, 27}, {28, 34}, {29, 30}, {31, 32}, {33, 36}, {35, 51}, {37, 38}, {39, 40}, {41, 43}, {42, 44}, {45, 46}, {47, 96}, {48, 49}, {50, 156}, {52, 53}, {54, 55}, {56, 58}, {57, 297}, {59, 61}, {60, 62}, {63, 87}, {64, 70}, {65, 67}, {66, 79}, {68, 69}, {71, 239}, {72, 73}, {74, 75}, {76, 85}, {77, 78}, {80, 82}, {81, 222}, {83, 84}, {86, 282}, {88, 89}, {90, 91}, {92, 94}, {93, 284}, {95, 98}, {97, 100}, {99, 168}, {101, 285}, {102, 103}, {104, 105}, {106, 110}, {107, 108}, {109, 112}, {111, 113}, {114, 115}, {116, 117}, {118, 119}, {120, 122}, {121, 123}, {124, 126}, {125, 128}, {127, 130}, {129, 216}, {131, 132}, {133, 134}, {135, 144}, {136, 137}, {138, 140}, {139, 141}, {142, 143}, {145, 146}, {147, 275}, {148, 149}, {152, 165}, {153, 154}, {155, 159}, {157, 158}, {160, 173}, {161, 162}, {163, 164}, {166, 167}, {169, 170}, {171, 174}, {172, 183}, {175, 176}, {177, 178}, {179, 180}, {181, 182}, {184, 187}, {185, 186}, {188, 195}, {189, 190}, {191, 409}, {192, 194}, {193, 197}, {198, 442}, {199, 211}, {200, 201}, {202, 203}, {204, 207}, {205, 206}, {208, 209}, {210, 295}, {212, 213}, {214, 215}, {217, 256}, {218, 219}, {220, 221}, {223, 224}, {225, 226}, {227, 228}, {229, 230}, {231, 232}, {233, 236}, {235, 251}, {237, 238}, {240, 250}, {241, 242}, {243, 246}, {244, 245}, {247, 248}, {252, 253}, {254, 255}, {257, 258}, {259, 260}, {261, 262}, {263, 264}, {267, 387}, {268, 271}, {269, 274}, {270, 462}, {272, 273}, {276, 331}, {277, 329}, {278, 281}, {279, 280}, {283, 324}, {286, 342}, {287, 288}, {289, 291}, {290, 441}, {292, 298}, {293, 294}, {296,

299}, {301, 303}, {302, 305}, {304, 308}, {306, 307}, {309, 312}, {310, 311}, {313, 314}, {315, 316}, {317, 319}, {318, 320}, {321, 322}, {323, 340}, {325, 334}, {326, 327}, {328, 345}, {330, 468}, {332, 339}, {333, 346}, {335, 336}, {338, 343}, {341, 500}, {344, 347}, {348, 355}, {349, 353}, {350, 363}, {351, 367}, {352, 359}, {354, 356}, {357, 481}, {358, 365}, {360, 362}, {361, 364}, {366, 479}, {368, 369}, {370, 374}, {371, 372}, {373, 383}, {375, 482}, {376, 420}, {377, 379}, {378, 382}, {380, 381}, {384, 401}, {385, 386}, {388, 452}, {389, 392}, {390, 391}, {393, 400}, {394, 399}, {395, 396}, {397, 398}, {403, 405}, {404, 412}, {406, 407}, {408, 410}, {411, 413}, {414, 443}, {415, 416}, {417, 418}, {419, 537}, {421, 422}, {423, 424}, {425, 426}, {427, 433}, {428, 429}, {430, 431}, {432, 434}, {436, 460}, {437, 438}, {439, 440}, {444, 445}, {446, 461}, {447, 448}, {449, 455}, {450, 458}, {451, 453}, {456, 457}, {463, 467}, {464, 465}, {466, 485}, {469, 470}, {471, 472}, {473, 474}, {475, 476}, {477, 486}, {478, 480}, {483, 484}, {488, 489}, {490, 491}, {492, 497}, {493, 496}, {494, 495}, {499, 501}, {502, 503}, {504, 506}, {505, 507}, {508, 509}, {510, 511}, {512, 513}, {514, 515}, {516, 517}, {518, 520}, {519, 529}, {521, 522}, {523, 525}, {524, 528}, {526, 527}, {530, 531}, {532, 533}, {534, 535}}.

Maximal Independent Vertex Set: {1, 2, 4, 6, 10, 14, 18, 22, 24, 27, 28, 29, 31, 33, 37, 40, 43, 52, 55, 59, 60, 66, 67, 68, 71, 72, 75, 77, 83, 85, 87, 89, 91, 96, 97, 101, 102, 104, 106, 107, 109, 111, 114, 117, 118, 122, 124, 127, 128, 131, 133, 136, 139, 140, 143, 144, 145, 149, 152, 153, 155, 157, 166, 168, 170, 172, 175, 177, 179, 181, 186, 187, 191, 192, 193, 199, 200, 203, 206, 212, 215, 218, 221, 222, 223, 225, 231, 234, 237, 242, 243, 245, 249, 253, 256, 259, 264, 268, 273, 274, 277, 280, 281, 282, 284, 289, 292, 295, 299, 301, 302, 304, 306, 309, 310, 314, 316, 319, 322, 326, 331, 337, 339, 340, 342, 345, 348, 353, 360, 361, 365, 373, 377, 378, 384, 385, 387, 390, 395, 399, 403, 408, 412, 413, 422, 426, 429, 430, 435, 439, 441, 443, 449, 451, 454, 458, 462, 464, 468, 469, 471, 475, 478, 482, 484, 486, 488, 495, 499, 502, 507, 511, 514, 517, 520, 525, 527, 529, 531, 532, 535}.

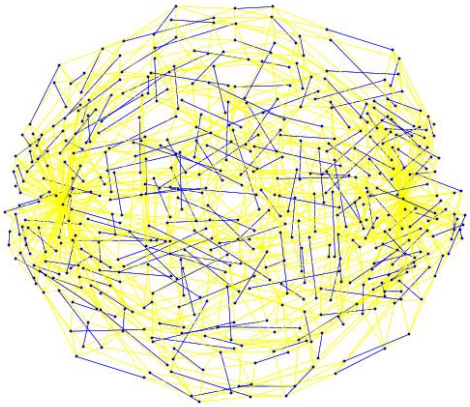


FIGURE 8. Perturbed Independent Edge Set (blue lines)

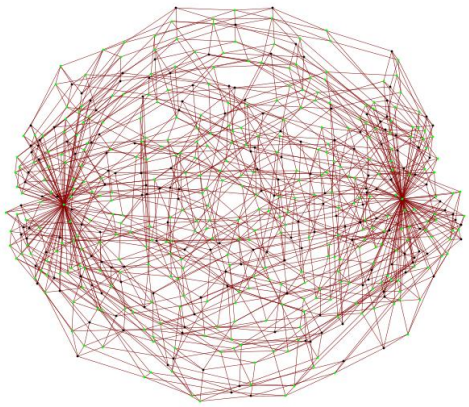


FIGURE 9. Perturbed Independent Vertex Set (blue dots)

We attempt to calculate the characteristic polynomial, but now the adjacency matrix has increased by a lot. Using *POLYMAKE*, we calculated the F-vector as (48, 471, 1333, 1445, 537). Notice that we used to have only 322 facets, but now we have 537. We can still obtain a bound for the chromatic number though. The upper bound is $1 + 12.1762 \approx 13$, and the lower bound is $537/181 \approx 4$, so the chromatic number bound is $4 \leq c \leq 13$.

2.5. A Search for Counterexamples

It took 53 years for us to find one counterexample for the Hirsch Conjecture. Can we find more? Santos' prismatoid is good, but computationally it is not. As discussed above, we still cannot enumerate all 86 vertices due to the large dimension and computation cost needed, thus we wish to search for a 'smaller' counterexample so we can actually attempt to calculate all the vertices in the final polytope. In addition, Santos' prismatoid lies in dimension 5. We know that all polytopes in dimension 3 have diameter at most 3 (one can prove this easily), but does a counterexample exist in dimension 4? In this section we will discuss some methods on attempting to search for another counterexample.

Cayley Embedding (via POLYMAKE software)
Our first attempt to create prismatoids is to use the build-in command *cayley-embedding* in *POLYMAKE*. We first create two $(d-1)$-dimensional polytopes, then use the command to create a d-dimensional prismatoid. *POLYMAKE* uses the two $(d-1)$-polytopes as the two base facets, thus the constructions of these two polytopes are critical. We defined the vertices of the $(d-1)$-polytopes in the following ways:

- Random process
- Trigonometric functions
- Cyclic polytopes
- Using preset or modified polytopes within Polymake

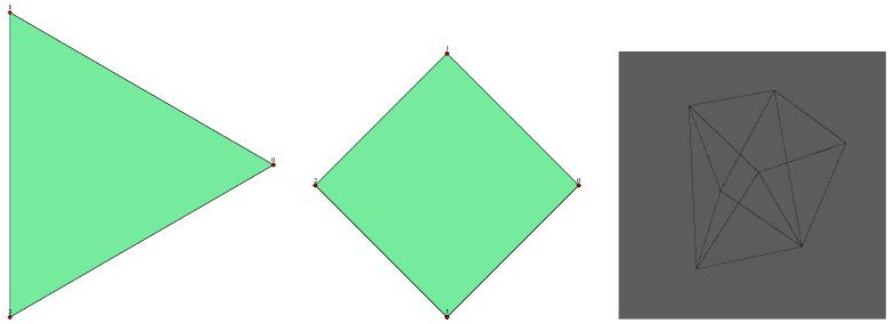


FIGURE 10. Here we show an example of a triangle (left) Cayley Embedded with a square (middle), to form the 3-dimensional prismatoid shown above (right). The top and bottom facets are the square and triangle respectively.

"Kiwi" method
A visual method we used in dimensions less than 6 to help construct prismatoids that violate the Hirsch Conjecture that rely on the d-step property as stated in Santos' paper. Within our investigations we applied this technique for the case when the dimension is equal to 4 for then we only had to deal with the construction of the 2-dimensional geodesic maps of the top and bottom facets of our prismatoid. What this essentially entailed was looking back at Santos example, taking the fundamental shape that violated the d-step property, and then attempting to tile the 2-sphere with this piece. See Fig 11.

To this end, it was necessary, but not sufficient, that each geodesic map on the 2-sphere represented the normal mapping of a 3-polytope, namely each mapping had to abide by Steinitzs theorem. However, quite soon there after we were notified by Professor De Loera that Professor

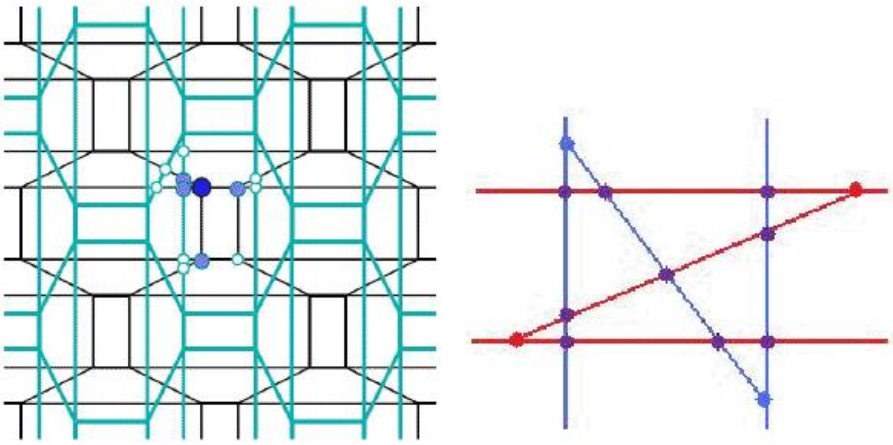


FIGURE 11. On the left is the map Santos constructed that violates the d-step property. On the right is the fundamental kiwi shape taken from Santos map that we used in attempting to construct a prismatoid in dimension 4.

Santos had given a sketch of a proof of the impossibility of a 4-dimensional prismatoid that violates the d-step property.

Graph Theoretic Approach
We took a step back and ask ourselves the question, what makes Santos counter-example so special? With this question in mind we set out to see whether the basic graph of Santos original prismatoid had any notable characteristics, e.g. what is the clique number, what is the chromatic number, what are the eigenvalues of the adjacency matrix, what is the degree sequence, etc. The results are listed above in the previous section.

Kissing Numbers
As we can see in Santos' prismatoid's graph, the distribution of the facets are very symmetric, thus we turn our direction into looking at kissing numbers. The "kissing number problem" asks for the maximal number of spheres that can touch a center sphere of the same size in n-dimensional space [**Mus04**]. For our problem, we looked at the arrangement of the two set of vertices from the two base facets projecting onto the same surface (this is equivalent to the Minkovski-Sum of the to base facets). But very soon we switch our direction.

Abstract Polytope
If we cannot explicitly find another counterexample, can we at least classify them abstractly? We invested our time into the study of abstraction and connected layer families, hoping to come up with some conditions that match with Santos' prismatoid and classify a class of abstract polytopes that violates the Hirsch Conjecture.

Despite of having all these powerful methods to construct polytopes, we are still hunting for a smaller counterexample. Does another smaller example exist in dimension 5 or higher? Can we construct another prismatoid following Santos' approach? Can we have some other polytope other than prismatoid that also violates the Hirsch Conjecture? All these questions remain open until now, and this will be the next step in the investment of the Hirsch Conjecture.

Finally, Santos (December 2010) has announced a smaller example in dimensional 5. Unfortunately this counterexample is again too large that we still cannot recover a direct Hirsch counterexample.

CHAPTER 3

An Application of Linear Programming: The Assignment Problem

3.1. Introduction

Suppose you are the manager of a small company, and there is a list of tasks you need to assign to your workers. Each person can only get one job, and each job can only be assigned to one person. You also know the efficiency of each worker for each task (call it the cost of the task). Your job is to assign these tasks to the workers such that the total cost of the assignment is minimized. In other words, you want to assign these jobs to workers so that the efficiency of the overall performance is maximized.

This is called an assignment problem. It is a fundamental combinatorial optimization problem in operational research or in optimization. In general, workers are called agents and jobs are called tasks. If the number of agents and jobs is equal, and the total cost of the tasks is equal to the total cost of the agents, we call this the Linear Assignment Problem.

3.2. Mathematical Theory

Let us discuss the mathematics behind this problem. In Chapter I I slightly mentioned a special type of polytope: the Birkhoff polytope (denoted as B_n). We also call it the assignment polytope, the polytope of doubly stochastic matrices, or the perfect matching polytope of $K_{n,n}$.

DEFINITION 3.1. A *doubly stochastic matrix* is an n×n matrix whose entries are non-negative real numbers and whose rows and columns each sum to 1. The Birkhoff polytope B_n is a convex polytope in $\mathbb{R}^{n^2}$whose points are doubly stochastic matrices.

THEOREM 3.2. *(Birkhoff von Neumann Theorem) The extreme points of the Birkhoff polytope are permutation matrices.*

PROPOSITION 3.3. [**RBM09**] *Any doubly stochastic matrix can be written as a convex combination of at most* $n^2 - 2n + 2$ *permutation matrices.*

Example: Consider the doubly stochastic matrix

$$X = \begin{pmatrix} 1/2 & 1/3 & 1/6 \\ 1/3 & 1/3 & 1/3 \\ 1/6 & 1/3 & 1/2 \end{pmatrix}.$$

We choose the first permutation matrix as

$$P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and λ_1 such that $\lambda_1 = min(1/2, 1/3, 1/2) = 1/3$ to multiply with P_1, we get

$$X - \lambda_1 P_1 = \begin{pmatrix} 1/6 & 1/3 & 1/6 \\ 1/3 & 0 & 1/3 \\ 1/6 & 1/3 & 1/6 \end{pmatrix}.$$

Next we choose

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and the corresponding $\lambda_2 = min(1/6, 1/3, 1/3)$ to get

$$X - \lambda_1 P_1 - \lambda_2 P_2 = \begin{pmatrix} 0 & 1/3 & 1/6 \\ 1/3 & 0 & 1/6 \\ 1/6 & 1/6 & 1/6 \end{pmatrix},$$

which can be decomposed into

$$1/6 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 1/6 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} + 1/6 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

So $X = 1/3P_1 + 1/6P_2 + 1/6P_3 + 1/6P_4 + 1/6P_5$.

In mathematical expression, we can express $B_n \in \mathbb{R}^{n^2}$ as

$$B_n = \left\{ x \in \mathbb{R}^{n^2} : \sum_{j=1}^{n} x_{ij} = 1; \sum_{i=1}^{n} x_{ij} = 1; x \geq 0 \right\}$$

and it has coefficient matrix

$$A = \begin{pmatrix} 1 & 1 & ... & 1 & & & & & & & & & & & \\ & & & & 1 & 1 & ... & 1 & & & & & & & \\ & & & & & & & & ... & & & & & & \\ & & & & & & & & & 1 & 1 & ... & 1 & & \\ 1 & & & & 1 & & & & & 1 & & & & & \\ & 1 & & & & 1 & & & & & 1 & & & & \\ & & ... & & & & ... & & ... & & & ... & & & \\ & & & 1 & & & & 1 & & & & & & 1 & \end{pmatrix}.$$

The coefficient matrix A is the incidence matrix of a complete bipartite graph $K_{n,n}$. It is totally unimodular, meaning that every regular $k \times k$ submatrix has determinant 1 or -1. If the right-hand side vector b of an equation system $Ax = b$ is integer valued and the regular matrix A is unimodular, then Cramer's rule implies immediately that the solution x of this equation system is also integer valued. This is a very important note in our assignment problem.

One interesting point to note is that the diameter of B_n is 2 for $n \geq 4$.

THEOREM 3.4. [**RBM09**] *dim P = d - rank(A), where $d = n^2$. The assignment polytope has rank 2n - 1, so the dimension is $(n-1)^2$.*

One of the problems in convex geometry is to find the volume of B_n. A combinatorial formula for all n was given in 2007 by De Loera, Liu, and Yoshida [**JADLY07**]. An asymptotic formula was found by Rodney Canfield and Brendan Mckay [**CM07**]:

$$vol(B_n) = exp(-(n-1)^2 lnn + n^2 - (n-1/2)ln(2\pi) + 1/3 + O(1)).$$

Enough of the theory behind this problem, now let us get to the main point of this section. A matching is defined when the set of agents is matched to the set of tasks. A problem can have many matchings (or solutions), depending on the criteria we have for our optimal match. For some problems, we might want to have an optimal matching, which means the cost is minimized, but for some problems, we might want to have a stable matching, meaning that switching two

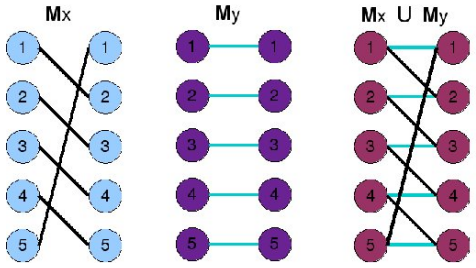


FIGURE 1. Adjacent vertices.

tasks between two agents will not improve the current solution. This has less involvement in linear programming, but finding a stable match is a very big problem in the stable marriage problem.

In the theorem stated above, the vertices of B_n are permutation matrices, and in Chapter I I said that the solutions to a linear program is actually the vertices of the polytope defined by the linear inequalities. This implies that a matching corresponds to a permutation matrix (and by theorem 3.2, for each n, we have $n!$ feasible solutions). This is not hard to imagine: suppose we have the rows as agents and columns as tasks. If we have a 1 in the ij^{th} spot, that means there is an assignment between that agent and that task, and otherwise if there is a 0. Obviously we cannot have anything other than 0 and 1, because we cannot assign a fraction of a task to an agent and vice versa. We call the solution a perfect matching if the number of agent equals to the number of tasks. Similar idea applies in the case where we have unequal numbers, except we will not have a square matrix anymore.

THEOREM 3.5. [**RBM09**] *Two distinct vertices X and Y of the assignment polytope are adjacent if and only if the edges of $M_x \cup M_y$ contain only one cycle, where M_x and M_y are perfect matchings of X and Y respectively.*

To visualize this, let us pick X and Y as

$$X = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and X' and Y' as

$$X' = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad Y' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

In the last graph of Figure 1, note that we only have one cycle, thus X and Y are adjacent, and in Figure 2, we see there are two cycles in the last graph. Thus X and Y are nonadjacent.

PROPOSITION 3.6. [**RBM09**] *Every vertex of the assignment polytope has*
$\sum_{k=0}^{n-2} \binom{n}{k}(n-k-1)!$

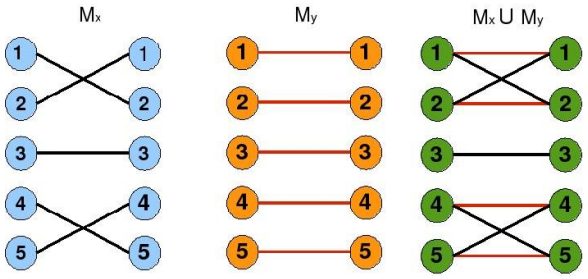


FIGURE 2. Non adjacent vertices.

neighbors.

Now back to linear programming. Our problem here is a special type of linear program, called the integer program. It has the form

$$\begin{aligned} &\min \textstyle\sum_{i,j=1}^{n} c_{ij}x_{ij} \\ &\text{s.t. } \textstyle\sum_{i=1}^{n} x_{ij} = 1 \\ &\quad \textstyle\sum_{j=1}^{n} x_{ij} = 1 \\ &\quad x_{ij} \in 0, 1 \end{aligned}$$

where $x_{ij} = 1$ means the agent takes the job, and $x_{ij} = 0$ otherwise. Solving this is equivalent to finding a permutation matrix that minimized the cost function.

As mentioned above, there are many ways to solve this problem, depending on what type of solution we want. I will describe the two algorithms that we used during our research project for the mathematics department here: the Hungarian Algorithm and the Gale-Shapley Algorithm [**SMP10**].

3.3. Our project and the Hungarian Algorithm

In summer 2009, the office of mathematics department at UC Davis requested the mathematics department to come up with a better way to assign graduate students to teach discussion sections. By better here I mean a more efficient and optimal way compared to the old method used by the staff. We found out that the staff used to do the assignment by hand, thus taking a very long time and might create possible errors, such as the need to swap already assigned students because of conflicts, or the students not satisfied with their assignment. Our job was to come up with a computer program that incorporates some mathematical algorithms to help solve this assignment problem in a more efficient way with a more optimal solution.

To solve the problem, we have the following initial setups:

- Each graduate student supplies a (strict) preference ordering of at most 10 classes to teach, i.e. rating of 10 classes with 1-10 with 1 being most like;
- The staff also supplies a (strict) preference ordering of each graduate students who applies to the same class, i.e. rating of 1-5 with 1 being the most suitable.

We require both parties to supply a list for the following reason: We need to know what classes the graduate students like to teach, which is equivalent to the efficiency of different tasks mentioned above. We require the staff to supply a list as well, and this is equivalent to the manager knowing which workers excel in each particular job.

The Hungarian Algorithm solves the assignment problem by returning an optimal assignment. In our problem, the cost will be the preference, and finding the minimized cost is essentially minimizing the overall preference, thus creating the 'happiest' solution. I will first provide some remarks to it, then describe the algorithm. I will also give an example on how it works, and followed by some theorems related to it.

Algorithm 1 The Hungarian Algorithm

Input: An $n \times n$ matrix (c_{ij})
Output: An optimal assignment (x_{ij})
Step 1. (Initialization)
Step 1.1: For each i, let p_i be the minimum element in the ith row of (c_{ij}).
Step 1.2: Let (c'_{ij}) be computed from (c_{ij}) by subtracting p_i from each element of the ith row (for all i).
Step 1.3: For each j, let q_j be the minimum element in the jth column of (c'_{ij}).
Step 1.4: Let $(\bar{c}_{ij})$ be computed from (c'_{ij}) by subtracting q_j from each element of the jth column (for all j).
Step 2.
Step 2.1: Find a minimum collection of lines covering all the 0's of $(\bar{c}_{ij})$.
Step 2.2: If this collection of lines has fewer than n elements, go to step 3. Otherwise, go to step 4.
Step 3. (Modification of Reduced Matrix)
Step 3.1: Using the covering obtained in Step 2.1, let p be the smallest uncovered element in $(\bar{c}_{ij})$.
Step 3.2: Change the reduced matrix $(\bar{c}_{ij})$ by subtracting p from each uncovered element and adding p to each twice-covered element. Return to Step 2.
Step 4.
Step 4.1: Find a set of n independent 0's in $(\bar{c}_{ij})$.
Step 4.2: Let x_{ij} be 1 if the i, j entry of $(\bar{c}_{ij})$ is one of the independent 0's, and let x_{ij} be 0 otherwise. Output this solution (x_{ij}) and stop.

Remarks:

- In the input, the matrix (c_{ij}) is the $n \times n$ cost value matrix. The ij^{th} entry represents the cost for i to complete j.
- In step 1, despite the change of matrices, we are still minimizing the same objective function:
$$min \sum_{i,j} c'_{ij} x_{ij} = min \sum_{i,j} c_{ij} x_{ij} - p_k \sum_j x_{kj} = min \sum_{i,j} c_{ij} x_{ij} - p_k.$$
This implies that the assignment x_{ij} that minimizes $\sum_{i,j} c_{ij} x_{ij}$ also minimizes $\sum_{i,j} c'_{ij} x_{ij}$.
- In step 2, to find a minimum collection of lines covering the 0's means to find a set of lines that covers all the 0's in the matrix using the least number of lines.
- In step 4.1, we want to find a set of 0's such that no two zeros lie on the same row or column.
- The Hungarian Algorithm can be implemented in $O(n^3)$ time [**Law76**].

Example: Let us now see an example on how this algorithm works. Suppose $n = 4$ with
$$(c_{ij}) = \begin{pmatrix} 12 & 14 & 15 & 14 \\ 9 & 6 & 11 & 8 \\ 10 & 9 & 16 & 14 \\ 12 & 13 & 13 & 10 \end{pmatrix}.$$

Then $p_1 = 12$, $p_2 = 6$, $p_3 = 9$, $p_4 = 10$, and

$$\left(c'_{ij}\right) = \begin{pmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{pmatrix}.$$

Now $q_1 = 0$, $q_2 = 0$, $q_3 = 3$, $q_4 = 0$, so

$$(\bar{c}_{ij}) = \begin{pmatrix} 0 & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & 0 & 4 & 5 \\ 2 & 3 & 0 & 0 \end{pmatrix}.$$

Note that $(\bar{c}_{ij})$ always has nonnegative entries. For step 2, we get

$$(\bar{c}_{ij}) = \begin{pmatrix} \emptyset & \not{2} & \emptyset & \not{2} \\ 3 & \emptyset & 2 & 2 \\ 1 & \emptyset & 4 & 5 \\ \not{2} & \not{3} & \emptyset & \emptyset \end{pmatrix}.$$

Since we only have 3 lines (and $n = 4$), we need to go to step three. Here $p = 1$, so we get

$$(\bar{c}_{ij}) = \begin{pmatrix} 0 & 3 & 0 & 2 \\ 2 & 0 & 1 & 1 \\ 0 & 0 & 3 & 4 \\ 2 & 4 & 0 & 0 \end{pmatrix}.$$

Now it is easy to see that we need 4 lines to cover all the 0's, so we can go to step 4. We can now find an optimal solution by letting $x_{13} = 1$, $x_{22} = 1$, $x_{31} = 1$, $x_{44} = 1$, and the rest of x_{ij} be 0.

THEOREM 3.7. [**Rob03**] *If all the c_{ij} are integers, the Hungarian Algorithm gives an optimal assignment.*

Notice that the Hungarian Algorithm only takes into account of the preference of one side, thus in our project we have to generate two solutions optimizing each side, and let the office picks its choice.

3.4. Gale-Shapley Algorithm

The stable marriage problem can be stated as follow: Given n men and n women, where each person has ranked all members of the opposite sex with a unique number between 1 and n in order of preference, marry the men and women together such that there are no two people of opposite sex who would both rather have each other than their current partners. If there are no such people, all the marriages are "stable".

In 1962, David Gale and Lloyd Shapley presented an algorithm that guarantees a stable solution with everyone married. We call it the Gale-Shapley Algorithm [**SMP10**].

The main idea of the algorithm is as follow: One side will do the "proposing", while the other side either accepts or rejects the proposal, and the algorithm creates a stable solution favoring the proposing side. We do not use the word "optimal" to describe the solution, because both side may not be as happy as they want, but at least switching up the solution will not make the situation better.

DEFINITION 3.8. **Male-Optimal:** A stable matching in which each man is paired with the highest ranked woman possible.

THEOREM 3.9. *The Gale-Shapley Algorithm generates either the male-optimal solution or the female-optimal solution.*

The algorithm is as follow:

Algorithm 2 Gale-Shapley Algorithm

Input: Preference list of both parties.
Output: A stable match between the two parties.
(In our algorithm we let the males to do the proposals to the females.)
Step 1: Each male proposes to his most desired woman in order.
Step 2: If the woman is not yet matched, she will accept the proposal. If the woman is already matched, she will decide whether to break the engagement and accept the new proposal, or reject the new proposal according to her preference list.
Step 3: If the male is being rejected, he will propose to his next most desired woman on his list.
Step 4: The algorithm terminates when everyone is matched, and this matching is stable.

DEFINITION 3.10. A **blocking pair** exists if there exists a male and a female who are not assigned as a pair during the matching, and each prefers the other over their respective matched partner. A matching is stable if there does not exist any blocking pair.

Remarks:

- The complexity of this algorithm is $O(n^2)$.
- This algorithm generates a solution that favors only one party. Even though the solution is stable, it is not necessary optimal. Thus in our project we generate two solutions, having each side "propose" once.

3.5. Results of the project

We implemented the Hungarian Algorithm and the Gale-Shapley Algorithm into Matlab. Our program takes in as input a preference matrix from an excel file and returns the optimal and stable solutions.

We have three codes for this project: assignment, optimal, and GS.

- **Assignment:** First we require each graduate student to rate 10 classes. Then we ask each of them to provide a time sheet for time conflicts, i.e. the times when they cannot teach a class for varies reasons. Lastly we obtain a spreadsheet with times of the discussion sections on it. The assignment code takes in these three spreadsheets, apply some matrix multiplications, and output a new spreadsheet with the graduate students' preferences for classes with time conflicts taken into account. This step is necessary because the graduate students do not know when are classes being offered, thus they might have signed up for a class that they cannot teach because of time conflicts.
- **Optimal:** This code runs the Hungarian algorithm. It takes in the preference matrix generated by assignment, as well as the preference list from the office as input. We generate two optimal solutions by running the program twice, one optimizing the students' preferences, the other optimizing Mathematics department's preferences.

Teaching Assistant	Hungarian (TA Optimal)	Hungarian (Math Dept. Optimal)	Gale-Shapley
L.A.	21C LeadB	21C LeadB	21C-LeadB
S.B.	21B-Lead	21B-Lead	21B-Lead
B.F.	146	25	25
M.L.1	21C-B04	21D-A01	21C-A03
A.R.	135B	22A-1	22AL-1
A.V.	21A-A02	21D-A03	22A-A02
Q.W.1	108	21C-B03	
M.S.	128C	128C	128C
M.R.	129	141	129
M.R.	147	21A-A02	141
J.C.	22B-1	21C-C01	148

FIGURE 3. Sample Result From our Project.

- **GS:** This code runs the Gale-Shapley algorithm. Since the algorithm considers both preference matrices as input, it suffices to produce only one table, though a second table favoring the other side might be necessary for comparison by running the code again with the input in reversed order.

As you can see in Figure 3, the two results generated by the Hungarian Algorithm are not necessary the same. Also note that the results from the Hungarian Algorithm and the Gale-Shapley are different for some entries. If you look close enough, you will see a blank space under Gale-Shapley. This is because we do not require the graduate students to rate all the classes, thus there is only a limited number of class choices for each graduate student. If all the classes that the graduate student proposes to are filled, then an unmatched student will appear. In this case hand-made justification is necessary to guarantee that every class is being matched with a graduate student.

The research on the Stable Marriage Problem is still on going. In October 2010, Ashlagi, Braverman and Hassidim presented a new algorithm that solves a special type of the matching problem. For more information, please refer to [**IAH10**].

Bibliography

[AM85] I. Adler and N. Megiddo, A simplex algorithm where the average number of steps is bounded between two quadratic functions of the smaller dimension, Journal of the ACM (1985).

[BB08] S. Boyd and C. Barratt, Ellipsoid method.

[Big74] Norman Biggs, Algebraic graph theory, Gambridge University Press, 1974.

[Bor99] K. H. Bordwardt, A sharp upper bound for the expected number of shadow vertices in lp-polyhedra under orthogonal projection on two-dimensional planes., Mathematics of operations Research (1999).

[bp10] Birkhoff polytope, `http://en.wikipedia.org/wiki/Birkhoff_polytope#cite_ref-5`, November 2010.

[CM07] E. Rodney Canfield and Brendan D. McKay, The asymptotic volume of the birkhoff polytope, 330–332.

[DB09] W. Hua L. Schewe D. Bremner, A. Deza, More bounds on the diameters of convex polytopes.

[EK10] F. Santos E. Kim, An update on the hirsch conjecture, Jahresbericht der Deutschen Mathematiker-Vereinigung (2010).

[ell] `http://en.wikipedia.org/wiki/Ellipsoid_method`.

[FE09] A. Razborov T. RothvoßF. Eisenbrand, N. Háhnle, Diameter of polyhedra: Limits of abstraction.

[GR01] Chris Godsil and Gordon Royle, Algebraic graph theory, Springer, 2001.

[IAH10] Mark Braverman Itai Ashlagi and Avinatan Hassidim, Matching with couples revisited.

[IAS87] R. M. karp I. Adler and R. Shamir, A simplex variant solving an m x d linear program in $o(min(m^2, d^2))$ expected number of steps, Journal of Complexity (1987).

[IGS09] Stephen G. Nash Igor Griva and Ariela Sofer, Linear and nonlinear optimization, second ed., Society for Industrial and Applied Mathematics, 2009.

[JADL10] Cynthia Vinzant Jess A. De Loera, Bernd Sturmfels, The central curve in linear programming.

[JADLY07] Fu Liu Jesus A. De Loera and Ruriko Yoshida, A generating function for all semi-magic squares and the volume of the birkhoff polytope, 330–332.

[JMW96] m. Sharir J. Matousek and E. Welzl, A subexponential bound for linear programming, Algorithmica (1996).

[Kar84] N. K. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica **4** (1984), 373–395.

[Kha80] L. G. Khachiyan, Polynomial algorithms in linear programming.

[Kim10] E. Kim, Geometric combinatorics of transportation polytopes and the behavior of the simplex method, Ph.D. thesis, University of California, Davis, 2010.

[KK92] G. Kalai and D. J. Kleitman, A quasi-polynomial bound for diameter of graphs of polyhedra, Bulletin of the American Mathematical Society (1992).

[Kle67] V. Klee, Diameters of polytopes, Convex Polytopes (1967).

[KW67] V. Klee and D. W. Walkup, The d-step conjecture for polyhedra of dimension d ¡ 6., Acta Math. (1967).

[Lar70] D. G. Larman, Paths on polytopes, Proceedings of the London Mathematical Society (1970).

[Law76] E. L. Lawler, Combinational optimization: Networks and matroids, 330–332.

[Mus04] O. R. Musin, The kissing problem in three dimensions.

[pol] `http://polymake.org/doku.php/start`.

[RB81] M.Tood R. Bland, D.Goldfarb, The ellipsoid method: A survey, Operations Research **29** (1981), 1039–1091.

[RBM09] Mauro Dell'Amico Rainer Burkard and Silvano Martello, Assignment problems, Society for Industrial and Applied mathematics, 2009.

[Rob03] Fred Roberts, Applied combinatorics, Prentice Hall, 2003.

[San10] F. Santos, A counterexample to the hirsch conjecture.

[Sho77] N. Z. Shor, Cut-off method with space extension in convex programming problems.

[sim] `http://upload.wikimedia.org/wikipedia/commons/7/78/Simplex_description.png`.

[SMP10] Stable marriage problem, `http://en.wikipedia.org/wiki/Stable_marriage_problem`, December 2010.

[Tod86] M. J. Todd, Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems, Mathematical Programming (1986).

[Too02] M. J. Tood, The many facets of linear programming, Mathematical Programming **91** (2002), 417–436.

[Wil67] H. S. Wilf, The eigenvalues of a graph and its chromatic number, J. London Math. Soc., (1967), 330–332.

[YN76] D. B. Yudin and A. S. Nemirovski, Informational complexity and efficient methods for the solution of convex extremal problems.