Complexity of Pivot Algorithms

Qi Wang, Sean Kelley

December, 2020

Abstract

Abstract...

1 Introduction

The Simplex method was invented by Dantzig in 1947 [7] to solve the Linear Programming (LP) problem. It was a tableau based method which pivots between solutions under certain pivoting rules. Another framework of pivot algorithms is the criss-cross method, which was proposed by Zionts at 1969 [9] and then [12], [4] present a finite criss-cross version developed independently around 1980s.

Many pivot rules have been designed, the result of which is that the simplex method and criss-cross have many variants depending on which pivot rule they employ. Some pivot rules have same motivation of Dantzig's original pivoting rule, which is to find a "best possible" nonbasis variable enter the basis, but these rules could cause cycling when the problem is degenerate (when some rhs coefficients of b are zero). Thus a series pivot rules were proposed to ensure the algorithms are finite. Bland's rule is one of the them. We will present it and several other pivot rules in section 3.3.

Csizmadia and Illés in 2010 [6] proved that a general pivot rule - the s-monotone index selection rule, which include Bland's, and two other index selection rules - can ensure the algorithms are finite. They also provide guidance towards revising other pivot rules to be finite. We present their research in section 3.4.

Although simplex method was generally efficient in practice, theoretically, Klee and Minty [5] in 1970s showed the worst case complexity could happen. For a specific type of LP with n variables and n constraints, using the simplex method with Dantzig's rule, the number of iterations is $2^n - 1$. We will demonstrate this in section 4.1.

Kitahara and Mizuno in 2011 [10] analyzed the complexity of the simplex method with certain pivot rules. They came up with a bound that is a polynomial of n, m, and a special ratio when A is unimodular and b is integral. We showed their analysis in section 4.2.

2 Notations

In this paper, we consider the primal Linear Optimization (LO) problem of the standard form

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ x \in \mathbb{R}^n,\ c \in \mathbb{R}^n,\ & A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m \end{aligned} \tag{1}$$

Table 1 lists notations we will refer back to throughout the paper.

Table 1: Notation

n	number of variables
m	number of constraints (assume $m \leq n$)
B	set of index of basis solution x_B, $\lvert B \rvert = m$
N	set of index of nonbasis solution x_N, $\lvert N \rvert = n - m$

3 Pivot algorithms and pivot rules

As we will go on to see in this article, much of our conversation will revolve around advancements in pivoting rules that led to performance increases for pivot algorithms. Before getting there, however, we would like to begin by discussing a few key improvements to other aspects of pivot algorithms that contributed to research focus in this space. We will start by going all the way back to the beginning.

3.1 Dantzig's Simplex Algorithm

As mentioned in our opening, the major advancement that opened the space of pivot algorithms was the development of the Primal Simplex Algorithm (referred to hereon as Simplex) by Dantzig, and his work will serve as our baseline for the complexity of pivot algorithms as we go on to compare them

moving forward. In order to understand the complexity of the algorithm Dantzig proposed, let's take a moment to refresh ourselves on its operations. Although mentioned in greater detail in [7], the important details to recall are as follows. We begin by setting up the following system of equations where the equality exists between the first and second columns:

0	$-c_B^T x_B$	$-c_N^T x_N$	x_0
b	$A_B x_B$	$A_N x_N$	0

Then with a couple of algebraic operations, we get the following tableau:

$c_B^T A_B^{-1} b$	0	$(c_B^T A_B^{-1} A_N - c_N)^T x_N$	x_0
$A_B^{-1} b$	$I x_B$	$A_B^{-1} A_N x_N$	0

Simplex tells us to then pivot through bases until we find one that is either optimal or inconsistent. The pivoting operation for primal simplex method is as following

- If there is dual infeasibility
  - Find an index from the nonbasis elements which is dual infeasible, i.e, choose $q \in \mathcal{Q}$ where $\mathcal{Q} := \{j \in N : (c_B^T A_B^{-1} A_j - c_j) > 0\}$. If $|\mathcal{Q}| = 1$, we just choose the only index, otherwise we make a choice based on some rules we will discuss later.
  - Find an index from the basis by the ratio test to keep the primal solution feasible, i.e., choose $p \in \mathcal{P}$ where $\mathcal{P} := \{\arg\min_{i \in B} \frac{x_i}{(A_B^{-1} A_q)_i} : (A_B^{-1} A_q)_i > 0\}$. If $|\mathcal{P}| = 0$, the LO is dual infeasible and primal unbounded, else if $|\mathcal{P}| = 1$, we choose the only index, otherwise we make choice by some rules we will again discuss later.
  - Perform a pivot: $B := B \cup \{q\} \backslash \{p\}$, $N := N \cup \{p\} \backslash \{q\}$.
- Else, the current solution solves the LO problem.

3.2 Criss-cross algorithm

The difference between the criss-cross algorithm and the simplex method is that criss-cross does not maintain primal or dual feasibility and thus omits

the ratio test procedure. The similarities are that both the simplex method and criss-cross algorithm are pivot algorithms and need some rules to choose the pivot element in each iteration. Such a rule is crucial for the complexity and the finiteness of both algorithms.

3.3 Pivoting rules

From now on, we will focus on the simplex method unless otherwise stated. In the simplex method, the decision of which nonbasis element enter and which basis element leaves (if more than one basis satisfy ratio test) is flexible. Thus many pivot rules have been designed and we will show some of them.

3.3.1 Dantzig's rule

To make a pivot, Dantzig said to do the following:

- Find the most positive element in the top row. Denote its index as q ($n - m$ operations)
- Find the smallest $x_i/\tau_{iq} : \tau_{iq} > 0$, where τ_{iq} is the ith element of the qth column in our tableau. If there is a tie, arbitrarily pick one of the tying elements ($3m$ operations)
- Swap i and q in their respective index sets.
- Multiply A_B^{-1} by A_N ($m^2 * (n - m)$ operations)
- Add $c_B^T A_B^{-1} A_N$ to $-c_N^T$ ($m^2 + m(n - m) + (n - m)$ operations)

We see then this takes us to a total of $m^2(n-m)+m^2+m(n-m)+m+2n = O(m^2n)$ operations for the pivot.

Now as we showed in class [13], we know that so long as there is no degeneracy in the problem, simplex cannot cycle, so the number of pivots is bounded by the number of possible bases, which was $\binom{n}{m}$. Therefore, we know Dantzig's version of simplex to have complexity $O(\binom{n}{m}m^2n)$ for nondegenerate problems.

3.3.2 Bland's Rule (Least-index)

Now for Dantzig's simplex, we assumed that a problem would have no degeneracy, which is a rather high standard for using the algorithm in practice. To further complicate the matter, degeneracy leads to cycling in that version of

simplex, which means the algorithm could run infinitely and never converge. To get around that issue then, many pivot rules have been proposed which help simplex avoid cycles. Arguably none of these such pivoting rules are more famous than Bland's Rule, which updates the above steps of simplex in the following way:

- instead of picking the most positive top row element to enter the basis, pick the first positive top row element.
- instead of arbitrarily selecting a basis row matching the minimal ratio, select the minimal ratio basis row with least index.

As shown in [3], these updates to Dantzig's simplex method ensure that we never cycle and make all linear programs converge and finite. Thus all linear programs now have complexity $O(\binom{n}{m}m^2n)$.

3.3.3 Revised Simplex Method

With simplex having a finite worst case across all problems, it is worthwhile to look at ways now to lower that upper bound. One of the first places the optimization community had success in doing this was with speeding up the process of making a pivot, which has come to be known as the Revised Simplex Method. Again, Revised Simplex differs from Dantzig's Simplex in two key ways [15]:

- instead of calculating all of the objective row for each pivot, just calculate enough columns until we get a positive entry, pivoting on that (up to $n - m$ operations)
- instead of multiplying A_B^{-1} by all of A_N, just multiply by the column corresponding to our first found positive entry (m^2 operations).

It is worth highlighting at this point that [15] shows Revised Simplex yields the same solution as Dantzig's Simplex but saves us from making many unnecessary calculations. This change brings our pivot to $2m^2 + m(n - m) + m + 2n = O(mn)$ operations (the same as in Dantzig's Simplex but swap $m^2(n - m)$ for m^2).

3.3.4 Best improvement rule

This rule also goes by largest improvement rule or greatest improvement rule. It aims for largest increase in objective value. It will pick the non-basic variable with index $j := \arg\max_{j \in N}(c_B^T A_B^{-1} A_j - c_j) \times t_j$ where $t_j =$

$\min_{i\in\{B:(A_B^{-1}A_j)_i>0\}} \frac{x_i}{(A_B^{-1}A_j)_i}$. In other words, the improvement to the objective value is the reduced cost times the ratio, and we choose the nonbasis element that brings the largest improvement in this iteration. For example, as in [11], the tableau is

	rhs	$-s_1$	$-s_2$	$-s_3$	$-s_4$
z	0	2	3	0	0
x_3	2	1	2	1	0
x_4	3	1	-1	0	1

Now x_1, x_2 are nonbasis elements that are dual infeasible. For x_1 we can increase it to $t_1 = \min\{\frac{2}{1}, \frac{3}{1}\} = 2$; for x_2, $t_2 = \min\{\frac{2}{2}\} = 1$. Since $2t_1 = 4 > 3 = 3t_2$, we choose x_1 to enter according to this rule. Notice this rule needs more arithmetic operations in each iteration, but it may save number of iterations, which we will talk about later.

3.3.5 Steepest edge rule

This rule builds off Best Improvement, but this time we normalize our reduced cost with respect to the norm of its column. Specifically, we choose $\max_{j\in N} \frac{c_B^T A_B^{-1} A_j - c_j}{\|A_B^{-1} A_j\|}$.

3.3.6 Last-in-first-out rule (LIFO)

In addition to Bland's rule, this rule, plus the following most-often-selected-variable (MOSV) rule are also finite pivot rules. LIFO and MOSV are proposed by Zhang in 1997 [17]. LIFO means among the candidates, choose the most recently moved variable. In the first iteration with no selection in the past, we use Bland's rule or Dantzig's rule.

3.3.7 Most-often-selected-variable (MOSV)

MOSV rule is to select the variable that has been selected the largest amount of times before, that is the "most often selected". In the first iteration with no selection in the past, we use Bland's rule or Dantzig's rule. When the choice by LIFO (or MOSV) rule is not unique, we use Dantzig's rule. This is called Hybrid-LIFO (Hybrid-MOSV) [17].

3.4 S-monotone index selection rules

Csizmadia and Illés in 2010 [6] proposed that Bland, LIFO, MOSV pivot rules are s-monotone index selection rules. They provide a explicit definition of this in [6]. The key steps of the s-monotone rule are

1. Maintain a sequence of vectors $s_k \in \mathcal{N}^n$, where k is the iteration number.
2. At iteration k, when selecting index from a couple of candidates (either a tie in the entering or leaving element), select the index $j := \arg\max_j s_k^j$, where j is the index of s_k.
3. When finishing the pivot, generate s_{k+1} by s_k and specific rules for Bland, LIFO and MOSV.
4. s_k is an nondecreasing vector from $k = 0, 1, 2, \cdots$.

We define i_k, o_k as indices of the entering variable and leaving variable respectively at iteration k. The s_k for Bland, LIFO, and MOSV is generated as follows

- **Bland**. $s_k = (n, n-1, \cdots, 1)^T$ for all k.
- **LIFO**.

$$s_{k+1}^i = \begin{cases} k, & \text{if } i \in \{i_k, o_k\}, \\ s_k^i & \text{otherwise.} \end{cases}$$

- **MOSV**.

$$s_{k+1}^i = \begin{cases} s_k^i + 1, & \text{if } i \in \{i_k, o_k\}, \\ s_k^i & \text{otherwise.} \end{cases}$$

Csizmadia and Illés [6] then proved the simplex method (as well as crisscross) with s-monotone rule is a finite algorithm. They also provided guidance of revising other pivot rules to be finite. For example, if we want to apply steepest edge rule but want to ensure it is finite, we can make the following modification:

1. At the beginning, define a sequence $\{p_k\}$ that is strictly increasing.

2. At iteration k, let

$$\gamma = \max_{j \in N} \frac{c_{B^k}^T A_{B^k}^{-1} A_j - c_j}{\|A_{B^k}^{-1} A_j\|},$$

and adjust p_k by

$$p_k = \begin{cases} p_k + \delta, & \text{if } p_{k-1} \geq \gamma, \\ \gamma, & \text{otherwise.} \end{cases}$$

where $\delta > 0$ is a given number. The $\{p_k\}$ by such adjustment is strictly increasing. Therefore we can use it to update the s_k as follows (similar to LIFO)

$$s_k^i = \begin{cases} p_k, & \text{if } i \in \{i_k, o_k\}, \\ s_{k-1}^i & \text{otherwise.} \end{cases}$$

3. Select the candidates (of indices) that has largest value of s_k.

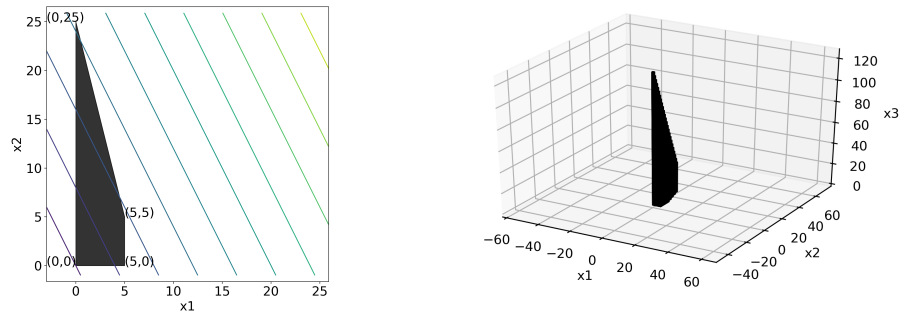
In such way the steepest edge method is combined with LIFO, and is s-monotone index rules (finite). It can also be combined with MOSV similarly. Consequently, they state an possible research direction is that to analyze the complexity of s-monotone index selection rules.

4 Contemporary complexity analysis

4.1 When will worst case happen

Given m constraints and n variables, there are at most $\binom{n}{m}$ possible basis. When $m = \frac{n}{2}$, $\binom{n}{m}$ get its maximum $\binom{n}{\frac{n}{2}} \approx \sqrt{\frac{2}{\pi n}} 2^n$, which is the worst complexity. Klee and Minty defined a type of LO problems that primal simplex with Dantzig's rule may visit all vertices before finally finding the optimal solution. Detailed introduction of Klee Minty cube can be referred to [14]. Here we demonstrate some examples. Klee Minty cube is defined as following LO problem 2. The coefficients might be different in other articles.

$$\begin{aligned} &\max \sum_{j=1}^{n} 2^{n-j} x_j \\ &\text{s.t. } 2 \sum_{j=1}^{i-1} 2^{i-j} x_j + x_i \leq 5^i, \quad i = 1, \cdots, n \\ &\qquad x_j \geq 0, \quad j = 1, \cdots, n \end{aligned} \tag{2}$$



(a) (b)

Figure 1: (a) Feasible region of 2-d Klee Minty Cube with contour lines (b) Feasible region of 3-d Klee Minty Cube

When $n = 2$ and $n = 3$ the 2-d and 3-d Klee Minty LO problems are

$$\begin{aligned} \max \quad & 2x_1 + x_2 \\ \text{s.t.} \quad & x_1 \leq 5 \\ & 4x_1 + x_2 \leq 25 \\ & x_1 \geq 0,\ x_2 \geq 0. \end{aligned} \qquad \begin{aligned} \max \quad & 4x_1 + 2x_2 + x_3 \\ \text{s.t.} \quad & x_1 \leq 5 \\ & 4x_1 + x_2 \leq 25 \\ & 8x_1 + 4x_2 + x_3 \leq 125 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0. \end{aligned}$$

The feasible region of 2-d and 3-d Klee Minty problems are shown in Figure 1. They look like "squashed" square and cube. We apply primal simplex method (a) least-index and (b) Dantzig pivot rules to demonstrate the number of iterations. Table 2 shows the tableau of least-index primal simplex method. Dantzig's rule coincides with result of least-index rule in this example.

Table 2: Tableau of primal simplex method (least index version), the element with * is the pivot element.

	rhs	x_1	x_2	s_1	s_2	
z	0	2	1	0	0	at vetex (0,0)
s_1	5	1*	0	1	0	
s_2	25	4	1	0	1	
z	-10	0	1	-2	0	at vetex (5,0)
x_1	5	1	0	1	0	
s_2	5	0	1*	-4	1	
z	-15	0	0	2	-1	at vetex (5,5)
x_1	5	1	0	1*	0	
x_2	5	0	1	-4	1	
z	-25	-2	0	0	-1	at vetex (0,25)
s_1	5	1	0	1	0	
x_2	25	4	1	0	1	

By taking a look at the constraints, the first constraint says that $x_1 \leq 5$. The second constraint says that $x_2 \leq 25$ and x_2 is smaller depending on how big x_1 is. Similarly, the third constraint says that $x_3 \leq 125$ (again, smaller depending on x_1 and x_2). Therefore, the constraints are approximately a set of upper bounds, which means that the feasible region is virtually a stretched n-dimensional hypercube with 2^n vertices. For such problem, primal simplex method with Dantzig's rule will iterate $2^n - 1$ pivot steps.

4.2 Kitahara and Mizuno analysis

In 2011, Kitahara and Mizuno [10] proved the complexity can be bounded by a polynomial of m, n and a special ratio. They utilized the analysis of Ye (2010) [16] for Markov Decision Problem. We summarize the key results of Kitahara and Mizuno [10] in this section.

We define some new notations. A BFS is said to be a basic feasible solution for primal LO problem (1). Let δ and γ be the minimum and the maximum values of all the positive elements of all BFSs. That is, for any BFS $\hat{x} \in \mathbb{R}^n$, if $\hat{x}_j \neq 0,\ j \in \{1, \cdots, n\}$ where the subscript j represent the jth entry of $\hat{x}$, we have

$$\delta \leq \hat{x}_j \leq \gamma. \tag{3}$$

The most important result of Kitahara and Mizuno [10] is the following theorem and corollary.

Theorem 1. *When we apply the simplex method with the Dantzig's rule or the best improvement rule for LO 1 having optimal solutions, we encounter at most*

$$n\lceil m\frac{\gamma}{\delta}\log(m\frac{\gamma}{\delta})\rceil \tag{4}$$

different basic feasible solutions.

Corollary 1. *If the primal problem is nondegenerate, the simplex method finds an optimal solution in at most* $n\lceil m\frac{\gamma}{\delta}\log(m\frac{\gamma}{\delta})\rceil$ *iterations.*

The proof can be referred to Kitahara and Mizuno [10]. Therefore the bound (4) is a polynomial of n, m and $\frac{\gamma}{\delta}$ with assumptions that the primal LO 1 has optimal solutions and is nondegenerate. Therefore, for Klee-Minty problem 2, $\gamma = 5^n$ while $\delta = 5$ and there ratio is exponential to n. This theoretically explains the complexity of Klee-Minty cube problem.

In practical, the ratio of $\frac{\gamma}{\delta}$ is not easy to get prior to solving the problem. However, for some specific LO, $\frac{\gamma}{\delta}$ can be bounded using LO coefficients.

4.2.1 LO with a totally unimodular matrix A and integral b

Definition 1. *A matrix A is totally unimodular if every square submatrix has determinant 0, -1 or 1. In particular, this implies that all entries are 0, -1 or 1.*

Totally unimodular matrices are very well behaved, because they always define polytopes with integer vertices, as long as the right-hand side is integer-valued. Many practical problem can be modelled using totally unimodular matrices, such as the node arc incidence matrix of a directed graph.

With totally unimodular A and integral b, all the elements of any BFS are integers, so $\delta \geq 1$. And for a BFS $x = (x_B, x_N)$, $x_N = 0$ and $x_B = A_B^{-1}b \geq 0$, since A_B is totally unimodular, A_B^{-1} is unimodular [10] and all the elements of A_B^{-1} are $-1, 0$ or 1. Thus for any $j \in B$ we have $x_j \leq \|b\|_1$, which implies $\gamma \leq \|b\|_1$. Thus $\frac{\gamma}{\delta} \leq \frac{\|b\|_1}{1}$. Thus by Theorem 1, we have

Corollary 2. *Assume that the constraint matrix A of LO (1) is totally unimodular and the constraint vector b is integral. When we apply the simplex method with the Dantzig's rule or the best improvement rule for LO (1), we encounter at most* $n\lceil m\|b\|_1\log(m\|b\|_1)\rceil$ *different basic feasible solutions. Moreover if the LO (1) is nondegenerate, this is the most iterations to find optimal solution.*

4.3 Del Pia and Michini Analysis

Earlier this year, another paper was released by Del Pia and Michini further refining the upper bounds on required pivots to solve certain subclasses of linear programs [1]. In their case, they focus on the simplex paths for LP's defined over lattice polytopes (i.e. polytopes with only integer vertices). Their analysis of these paths hinge on the following four algorithms.

4.3.1 Algorithm Review and Complexity

- A Basic Algorithm (A simplex method that avoids cycling)
  Input: Lattice Polytope P, vector c, vertex x^0 of P
  Ouput: Vertex x^* maximizing P

- A Scaling Algorithm
  Input: Lattice Polytope P, vector c, vertex x^0 of P
  Ouput: Vertex x^* maximizing P

  1. Let $l = \lceil \log \|c\|_\infty \rceil$. Define approximations $c^i = \lceil c/2^{l-i} \rceil$.
  2. For $i \in [l]$, let $x^{i+1} = \text{Basic Algorithm}(P, c^i, x^i)$.
  3. Return x^{l+1} optimal.

- A Preprocessing and Scaling Algorithm
  Input: Lattice Polytope P, vector c, vertex x^0 of P
  Ouput: Vertex x^* maximizing P

  1. Perturb c as described in Section 3 of [8]. Denote the output as $\bar{c}$
  2. Let $x^* = \text{Scaling Algorithm}(P, \bar{c}, x)$
  3. Return x^* optimal.

- An Iterative Scaling Algorithm
  Input: Lattice Polytope P, vector c, vertex x^0 of P
  Ouput: Vertex x^* maximizing P

  1. Let $x^* = x^0$.
  2. Let $\bar{c}$ be projection of c onto one active constraint from each previous iteration. If $\bar{c} = 0$, return x^*
  3. Let $\tilde{c}$ be the perturbation of $\bar{c}$ as described in Section 4 of [1].
  4. Let $x^* = \text{Scaling Algorithm}(P, \tilde{c}, x^*)$. Return to 2.

With these four algorithms, Del Pia and Michini go on to prove the following results.

- Basic Algorithm length $\leq nk\|c\|_\infty$
- Scaling Algorithm length is $O(nk \log\|c\|_\infty)$
  - Basic Algorithm is called $\log\|c\|_\infty) + 1$ times.
  - For scaled c each run of Basic Algorithm has $\leq nk$ pivots.
- Preprocessing and Scaling Algorithm length is $O(n^4k \log nk))$
  - Perturbed c reduces the number of times the Scaling Algorithm has to call the Basic Algorithm from $\log\|c\|_\infty$ to $n^3 \log nk$.
  - For each scaling of perturbed c, the run of Basic Algorithm still has $\leq nk$ pivots.
- Iterative Scaling Algorithm length is $O(n^2k \log nk\alpha)$, where α is the largest element in A.
  - Iterative Scaling Algorithm has at most n iterations.
  - The perturbation of $\bar{c}$ makes $\|\tilde{c}\|_\infty \leq n^3k\alpha$.
  - Recall that the Scaling Algorithm's length is $O(nk \log\|\tilde{c}\|_\infty)$, which in this case becomes $O(nk \log nk\alpha)$

Furthermore, for each algorithm, the authors show for a given set of inputs that the solution is the same as if those inputs had been run directly against simplex.

Now when looking at the last two results above, we appear to see a polynomial complexity for solving a problem with simplex (since we showed pivot calculations earlier to be polynomial). Now at first glance this might be exciting, but let's take a moment to recall their hypothesis. Del Pia and Michini assume our feasible vertices form a lattice polytope, which can rule out many LP's. Additionally, they state that the complexity is in terms of the number of vertices simplex visits, which in the case of a problem with degeneracy, is not the same as the number of pivots. So although this complexity, as we'll see next, is a nice improvement over the previous case of solving this class of LP, this class is a rather small subspace of all LP's. This isn't to say this work is insignificant, however, as much as it is to say there would be great value in further research to generalize this method to solve a wider variety of problems.

4.4 Relationship to Previously Discussed Works

The work of Del Pia and Michini builds on and relates to multiple other topics in this paper, and we would like to take a moment now to call out some of the highlights. Most noticeably, at the base of all of the algorithms developed by Del Pia and Michini was their Basic Algorithm, which finds an optimal solution over a lattice polytope for a cost vector. At its core, this Basic Algorithm runs an arbitrary non-cycling simplex method, which could be any one of the pivot algorithms mentioned before this section.

The second comparison worth calling out is the relationship between this result and those of Kitahara. Although Del Pia and Michini do the work of their paper with a general form of an LP, they mapped their problem to standard form to compare it to Kitahara. After doing so, they proved for the case the problem exists in a [0,k] polytope that Kitahara's bound was exponential for the number of vertices in the simplex path, whereas shown above, their bound is polynomial. In the case we just consider an LP in a [0,1]-polytope (thus the results of Kitahara apply since we have unimodular A), Del Pia and Michini show Kitahara's upper bound to simplify to $O(n^3 \log n)$ while theirs reduces to $O(n^2 \log n)$. We see then their result is a nice improvement in terms of complexity, and, since it applies to a wider array of problems, it is even much more general as well.

4.5 Software Implementations

After taking a bit of an account of the history of pivot algorithms, all the way from Dantzig to advances this year, we thought it may be of interest to the reader to point out what in our review is leveraged in practice. In his 2015 [2] talk, Bob Bixby goes into detail the implementation details of Gurobi and Cplex. In the early parts of this paper, we find a lot of overlap with what is used. In both solvers we find implementations of primal simplex using revised simplex to calculate each pivot. He goes on to detail that for a pivoting rule both softwares use steepest-edge since computational experiments have shown it to both find short paths around the feasible region as well as paths that are easy to calculate. The surprising point in the talk was that neither software actually uses a pivot rule with an anti-cycling scheme, which led us to conclude many of our mentioned pivot rules are actually not that useful in practice. Instead, both softwares have a way of perturbing the problem (by adjusting lower bounds slightly) so that degeneracy goes away but yet we still end up with the same optimal solution. Since there were a recent advancement and theory not yet uncovered at

the time of Bixby's talk, we decided to do a little outside searching for implementations of Del Pia's and Kitahara's papers. We were not able to find implementations for either, and we think this to be the case because the results are not general enough to be useful to LP solvers.

5 Conclusion

References

[1] Carla Michini Alberto Del Pia. Short simplex paths in lattice polytopes. 2020.

[2] Robert Bixby. Robert bixby: Solving linear programs: The dual simplex algorithm (3/3): Implementing the algorithm, 2020. URL: `https://youtu.be/uccbVoamiUM`.

[3] Robert Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977.

[4] Yow-Yieh Chang. Least-index resolution of degeneracy in linear complementarity problems. Technical report, STANFORD UNIV CA DEPT OF OPERATIONS RESEARCH, 1979.

[5] Wikipedia Contributors. Klee–minty cube, 11 2020. URL: `https://en.wikipedia.org/wiki/Klee%E2%80%93Minty_cube`.

[6] Zsolt Csizmadia, Tibor Illés, and Adrienn Nagy. The s-monotone index selection rules for pivot algorithms of linear programming. *European journal of operational research*, 221(3):491–500, 2012.

[7] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1951.

[8] A. Frank and E Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987.

[9] K. Fukuda and T. Terlaky. Criss-cross methods: A fresh view on pivot algorithms. *Mathematical programming*, 79(1-3):369–395, 1997.

[10] Tomonari Kitahara and Shinji Mizuno. A bound for the number of different basic solutions generated by the simplex method. *Mathematical Programming*, 137(1-2):579–586, 2013.

[11] University of Waterloo. Lecture note of co350 linear programming chapter 6: The simplex method, 2015. URL: `https://www.math.uwaterloo.ca/~hwolkowi//henry/teaching/f05/350.f05/L16.pdf`.

[12] T. Terlaky. A finite crisscross method for oriented matroids. *Journal of Combinatorial Theory, Series B*, 42(3):319–327, 1987.

[13] Tamas Terlaky. Introduction to mathematical optimization, 2015.

[14] Robert J Vanderbei. *Linear programming: foundations and extensions*, volume 285. Springer Nature, 2020.

[15] Harvey Wagner. A comparison of the original and revised simplex methods. *Operations Research*, 5(3):103–107, 1957.

[16] Yinyu Ye. The simplex method is strongly polynomial for the markov decision problem with a fixed discount rate, 2010.

[17] Shuzhong Zhang. New variants of finite criss-cross pivot algorithms for linear programming. *European Journal of Operational Research*, 116(3):607–614, 1999.