Pivoting Rules, Simplex Complexity and Ellipsoid Method

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

http://www.stanford.edu/~yyye
(LY, Chapter 5.1-5.3)
Very good on average, but the worse case ...?

When the simplex method is used to solve a linear program the number of iterations to solve the problem starting from a basic feasible solution is typically a small multiple of $m$, e.g., between $2m$ and $3m$.

At one time researchers believed—and attempted to prove—that the simplex algorithm (or some variant thereof) always requires a number of iterations that is bounded by a polynomial expression in the problem size.
Consider

\[
\begin{align*}
\max & \quad x_n \\
\text{subject to} & \quad x_1 \geq 0 \\
& \quad x_1 \leq 1 \\
& \quad x_j \geq \epsilon x_{j-1} \quad j = 2, \ldots, n \\
& \quad x_j \leq 1 - \epsilon x_{j-1} \quad j = 2, \ldots, n
\end{align*}
\]

where \(0 < \epsilon < 1/2\). This presentation of the problem emphasizes the idea (see the figures below) that the feasible region of the problem is a perturbation of the \(n\)-cube.
In the case of $n = 2$ and $\epsilon = 1/4$, the feasible region of the linear program above looks like
For the case where \( n = 3 \), the feasible region of the problem above looks like
The formulation above does not immediately reveal the standard form representation of the problem. Instead, we consider a different one, namely

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} 10^{n-j} x_j \\
\text{subject to} & \quad 2 \sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-1} \quad i = 1, \ldots, n \\
& \quad x_j \geq 0 \quad j = 1, \ldots, n
\end{align*}
\]

The problem above\(^a\) also be used is easily cast as a linear program in standard form. Unfortunately, it is less apparent how to exhibit the relationship between its feasible region and a perturbation of the unit cube.

\(^a\)It should be noted that there is no need to express this problem in terms of powers of 10. Using any constant \(C' > 1\) would yield the same effect (an exponential number of pivot steps).
Example

\[
\begin{align*}
\text{max} & \quad 100x_1 + 10x_2 + x_3 \\
\text{subject to} & \quad x_1 \quad \leq \quad 1 \\
& \quad 20x_1 + x_2 \quad \leq \quad 100 \\
& \quad 200x_1 + 20x_2 + x_3 \leq 10,000
\end{align*}
\]

In this case, we have three constraints and three variables (along with their non-negativity constraints). After adding slack variables, we get a problem in standard form. The system has \( m = 3 \) equations and \( n = 6 \) nonnegative variables. In tableau form, the problem is
The bullets below the tableau indicate the columns that are basic.
\[ T^1 \]

\[
\begin{array}{cccccccc}
-z & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & 1 \\
1 & 0 & 10 & 1 & -100 & 0 & 0 & -100 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & -20 & 1 & 0 & 80 \\
0 & 0 & 20 & 1 & -200 & 0 & 1 & 9,800 \\
\end{array}
\]
\[ T^2 \]

<table>
<thead>
<tr>
<th></th>
<th>(-z)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>100</td>
<td>-10</td>
<td>0</td>
<td>-900</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-20</td>
<td>1</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>200</td>
<td>-20</td>
<td>1</td>
<td>8,200</td>
</tr>
<tr>
<td>T³</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>-z</td>
<td>(x_1)</td>
<td>(x_2)</td>
<td>(x_3)</td>
<td>(x_4)</td>
<td>(x_5)</td>
<td>(x_6)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-100</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-10</td>
<td>0</td>
<td>-1,000</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-200</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-20</td>
<td>1</td>
<td>8,000</td>
<td></td>
</tr>
</tbody>
</table>
\[ T^4 \]

<table>
<thead>
<tr>
<th></th>
<th>(-z)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>-1</td>
<td></td>
<td>(-9,000)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>-200</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-20</td>
<td>1</td>
<td></td>
<td>8,000</td>
</tr>
</tbody>
</table>
\[
\begin{array}{cccccc|c}
-z & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & 1 \\
1 & 0 & 0 & 0 & -100 & 10 & -1 & -9,100 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & -20 & 1 & 0 & 80 \\
0 & 0 & 0 & 1 & 200 & -20 & 1 & 8,200 \\
\end{array}
\]
\[
\begin{array}{ccccccc}
-z & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & 1 \\
1 & 0 & -10 & 0 & 100 & 0 & -1 & -9,900 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & -20 & 1 & 0 & 80 \\
0 & 0 & 20 & 1 & -200 & 0 & 1 & 9,800 \\
\end{array}
\]

\[T^6\]
\( T^7 \)

<table>
<thead>
<tr>
<th></th>
<th>(-z)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(x_4)</th>
<th>(x_5)</th>
<th>(x_6)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-100</td>
<td>-10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-10,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>200</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10,000</td>
</tr>
</tbody>
</table>
\[(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 0, 10^4, 1, 10^2, 0)\]

is optimal and that the objective function value is 10,000.

Along the way, we made \(2^3 - 1 = 7\) pivot steps. The objective function made a strict increase with each change of basis.

**Remark.** The instance of the linear program (1) in which \(n = 3\) leads to \(2^3 - 1\) pivot steps when the greedy rule is used to select the pivot column. The general problem of the class (1) takes \(2^n - 1\) pivot steps. To get an idea of how bad this can be, consider the case where \(n = 50\). Now \(2^{50} - 1 \approx 10^{15}\). In a year with 365 days, there are approximately \(3 \times 10^7\) seconds. If a computer were running continuously and performing \(T\) iterations of the Simplex Algorithm per second, it would take approximately

\[
\frac{10^{15}}{3T \times 10^8} = \frac{1}{3T} \times 10^8\text{ years}
\]

to solve the problem using the Simplex Algorithm with the greedy pivot selection rule.
An interesting connection

Consider the eight vectors $v^k = (v_1^k, v_2^k, v_3^k)$ where $k = 0, 1, \ldots, 7$ and

$$v_j^k = \begin{cases} 
1 & \text{if } x_j \text{ is basic in tableau } k \\
0 & \text{otherwise}
\end{cases}$$

Looking at the eight tableaus $T^0, T^1, \ldots, T^7$, we see that

$$v^0 = (0, 0, 0) \quad v^4 = (0, 1, 1)$$

$$v^1 = (1, 0, 0) \quad v^5 = (1, 1, 1)$$

$$v^2 = (1, 1, 0) \quad v^6 = (1, 0, 1)$$

$$v^3 = (0, 1, 0) \quad v^7 = (0, 0, 1)$$

Now suppose we regard these vectors as the coordinates of the vertices of the 3-cube $[0, 1]$. 
The figure above illustrates the fact that the sequence of vectors $v^k$ corresponds to a path on the edges of the 3-cube. The path visits each vertex of the cube once and only once. Such a path is said to be Hamiltonian.

There is an amusing recreational literature that connects Hamiltonian path with certain puzzles. See Martin Gardner, “Mathematical games, the curious properties of the Gray code and how it can be used to solve puzzles,” *Scientific American* 227 (August 1972) pp. 106-109. See also, S.N. Afriat, *The Ring of Linked Rings*, London: Duckworth, 1982.
In a system of rank $m$, a (basic) solution that uses fewer than $m$ columns to represent the right-hand side vector is said to be degenerate. Otherwise, it is called nondegenerate.

A basic feasible solution will be nondegenerate if and only if its $m$ basic variables are positive.

Why is degeneracy a problem? The Simplex Algorithm can cycling (an infinite repetition of a finite sequence of bases) when a degenerate basic feasible solution crops up in the course of executing the algorithm, unless a suitable rule is employed to break the ties. Fortunately, there are rules to overcome this problem.
**Cycling Example**

\[
\begin{align*}
\text{min} & \quad -2x_1 - 3x_2 + x_3 + 12x_4 \\
\text{s.t.} & \quad -2x_1 - 9x_2 + x_3 + 9x_4 + x_5 = 0 \\
& \quad \frac{1}{3}x_1 + x_2 - \frac{1}{3}x_3 - 2x_4 + x_6 = 0 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]

Initially, the basic variables are \(\{x_5, x_6\}\) and it is in the canonical form. The pivot sequence shown in the table below leads back to the original system after 6 pivots.

<table>
<thead>
<tr>
<th>Pivot number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic var. out</td>
<td>(x_6)</td>
<td>(x_5)</td>
<td>(x_2)</td>
<td>(x_1)</td>
<td>(x_4)</td>
<td>(x_3)</td>
</tr>
<tr>
<td>Basic var. in</td>
<td>(x_2)</td>
<td>(x_1)</td>
<td>(x_4)</td>
<td>(x_3)</td>
<td>(x_6)</td>
<td>(x_5)</td>
</tr>
</tbody>
</table>
There are several methods for resolving degeneracy in linear programming. Among these are:

1. Perturbation of the right-hand side.
2. Lexicographic ordering.
3. Application of Bland's pivot selection rule.
Bland’s Rule

It is a double least-index rule consisting of the following two parts:

(i) Among all candidates for the entering column (i.e., those with $r_j < 0$), choose the one with the smallest index, say $e$.

(ii) Among all rows $i$ for which the minimum ratio test results in a tie, choose the row $r$ for which the corresponding basic variable has the smallest index, $j_r$.

**Theorem 1** Under Bland’s pivot selection rule, the Simplex Algorithm cannot cycle.
Sketch of Proof

Let initial tableau (we omit the right-hand-side vector $\mathbf{b}$ here)

$$
\mathcal{A} = \begin{bmatrix}
1 & \mathbf{c}^T \\
0 & \mathbf{A}
\end{bmatrix},
$$

where the column index from 0 to $n$ and row index from 0 to $m$. Now if cycling occurs, there is a set $\tau$ of indices $j \in \{1, \ldots, n\}$ such that $x_j$ becomes basic during cycling. Clearly $\tau$ has only a finite number of elements, so it has a largest element which we denote by $q$. Also note that during the cycling the right-hand-side vector $\mathbf{b}$ does not change and the values of all variables in $\tau$ are fixed at 0.

Let

$$
\mathcal{A}' = \begin{bmatrix}
1 & (\mathbf{r}')^T \\
0 & \mathbf{A}'
\end{bmatrix}.
$$

denote the tableau that first specifies $q$ as the pivot column, which means that $x_q$ is the entering variable at $\mathcal{A}'$. 

24
Let $y = (1; r')$. By virtue of the definition of $q$ and the rule that results in the choice of $q$, we have

$$
y_0 = 1, \quad y_j \geq 0 \quad 1 \leq j < q, \quad y_q < 0. \tag{1}\n$$

Note that the $(n + 1)$-vector $y$ belongs to the row space of $A$ or $A'$ or any subsequent tableau.

Now $x_q$ must also leave the basis, say immediately after some tableau

$$A'' = \begin{bmatrix}
1 & (r'')^T \\
0 & \bar{A}''
\end{bmatrix}
$$

where basic variable index set $B'' = (j_1, j_2, \ldots, j_m)$ with $q = j_r$. Let $t$ denote the entering variable to replace $x_q$. We define another $(n + 1)$-vector $v = (v_0, v_1, \ldots, v_n)$ as follows:

$$v_0 = r'_t < 0, \quad v_{B''} = \bar{A}_{,t}'', \quad v_t = -1, \quad v_j = 0 \quad \text{else.} \tag{2}\n$$

Note that $v_q = \bar{A}_{rt}'' > 0$ since $x_q$ is the outgoing variable. Note that $A''v = 0$ so that it is also in the null-space of $A'$, which implies $y \cdot v = 0$. By construction $y_0v_0 = v_0 < 0$ so that $y_jv_j > 0$ for some $j \geq 1$.

Since $y_j \neq 0$, $x_j$ must be nonbasic at $A'$; since $v_j \neq 0$, $x_j$ must be a basic variable at $A''$ or $j = t$. 
Accordingly, $j \in \tau$, and hence $j \leq q$. But by construction again, $y_q < 0 < v_q$ which implies that $y_q v_q < 0$ so that $j \neq q$.

Furthermore, (1) implies that $y_j > 0$, so $v_j > 0$. Thus, $j \neq t$ since $v_t = -1$ from (2). Let $j = j_p$ for some $p$. Then $A''_{pt} = v_j > 0$ and $b''_p = 0$.

But these contradict the assumption that $x_q$ is outgoing at $A''$, since $j < q$ and by Bland’s rule $j$ should be the outgoing variable. This means that cycling cannot occur when Bland’s Rule is applied.
Elements of Complexity Theory

The term *complexity* refers to the amount of resources required by a computation. Computational complexity wishes to associate to an algorithm more intrinsic measures of its time requirements:

- a notion of *input size*,
- a set of *basic operations*, and
- a cost for each basic operation.

The last two allow one to associate a (total) cost of a computation.
Polynomial Time Algorithms

- Bit size (bit operations) for integers and Unit size (unit cost) for real numbers.
- The former is usually referred to as the Turing model of computation, and latter is referred as the real number arithmetic model.
- An algorithm is said to be a polynomial time algorithm if its worst-case cost of computation is bounded above by a polynomial function of the input size of the problem data.
Ellipsoid Method: the first polynomial-time algorithm for LP

The basic ideas of the ellipsoid method stem from research done in the nineteen sixties and seventies mainly in the Soviet Union (as it was then called) by others who preceded Khachiyan. The idea in a nutshell is to enclose the region of interest in each member of a sequence of ellipsoids whose size is decreasing, resembling the bisection method.

The significant contribution of Khachiyan was to demonstrate in two papers—published in 1979 and 1980—that under certain assumptions, the ellipsoid method constitutes a polynomially bounded algorithm for linear programming.
Ellipsoids are just sets of the form

\[ E = \{ \mathbf{y} \in \mathbb{R}^m : (\mathbf{y} - \tilde{\mathbf{y}})^T B^{-1} (\mathbf{y} - \tilde{\mathbf{y}}) \leq 1 \} \]

where \( \tilde{\mathbf{y}} \in \mathbb{R}^m \) is a given point (called the center) and \( B \) is a symmetric positive definite matrix of dimension \( m \). We can use the notation \( \text{ell}(\tilde{\mathbf{y}}, B) \) to specify the ellipsoid \( E \) defined above. Note that

\[ \text{vol}(E) = (\det B)^{1/2} \text{vol}(S(0, 1)). \]

where \( S(0, 1) \) is the unit sphere in \( \mathbb{R}^m \).
By a Half-Ellipsoid of $E$, we mean the set

$$\frac{1}{2}E_a := \{y \in E : a^T y \leq a^T \tilde{y}\}$$

for a given non-zero vector $a \in \mathbb{R}^m$ where $\tilde{y}$ is the center of $E$.

We are interested in finding a new ellipsoid containing $\frac{1}{2}E_a$ with the least volume.

- How small could it be?
- How easy could it be constructed?
The New Containing Ellipsoid

The new ellipsoid $E^+ = \text{ell}(\bar{y}^+, B^+)$ can be constructed as follows. Define

\[ \tau := \frac{1}{m + 1}, \quad \delta := \frac{m^2}{m^2 - 1}, \quad \sigma := 2\tau. \]

And let

\[ \bar{y}^+ := \bar{y} - \frac{\tau}{(a^T B a)^{1/2}} B a, \]

\[ B^+ := \delta \left( B - \sigma \frac{B a a^T B}{a^T B a} \right). \]
Theorem 2  Ellipsoid $E^+ = \operatorname{ell}(\bar{y}^+, B^+)$ defined as above is the ellipsoid of least volume containing $\frac{1}{2} E^+_a$. Moreover,

$$\frac{\operatorname{vol}(E^+)}{\operatorname{vol}(E)} = \left(\frac{m^2}{m^2 - 1}\right)^{\frac{m-1}{2}} \cdot \frac{m}{m+1} = \left(1 + \frac{1}{m^2 - 1}\right)^{\frac{m-1}{2}} \cdot \left(1 - \frac{1}{m+1}\right)$$

$$< \exp \left(\frac{1}{m^2 - 1} \cdot \frac{m - 1}{2}\right) \exp \left(-\frac{1}{m+1}\right)$$

$$\leq \exp \left(-\frac{1}{2(m+1)}\right)$$

$$< 1.$$ 

Here we used

$$1 + a < e^a \quad \text{and} \quad 1 - a < e^{-a}$$

for $a > 0$. 

Figure 1: The least volume ellipsoid containing a half ellipsoid
Affine Transformation

Assume that $E = \text{ell}(\bar{y}, B)$, where the positive definite matrix $B$ has the factorization $B = JJ^T$. Now consider the affine transformation $y \mapsto \bar{y} + Jz$.

Let $y \in E$. Then $y - \bar{y} = Jz$ for some vector $z \in \mathbb{R}^m$. Now since $y \in E$,

$$1 \geq (y - \bar{y})^T B^{-1} (y - \bar{y})$$

$$= (Jz)^T (JJ^T)^{-1} Jz$$

$$= z^T J^T (J^T)^{-1} J^{-1} J z$$

$$= z^T z$$

so $z \in S(0, 1)$, the unit sphere. Conversely, every such point maps to an element of $E$. 


The ellipsoid method discussed here is really aimed at finding an element of a polyhedral set $Y$ given by a system of linear inequalities.

$$Y = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{a}_j^T \mathbf{y} \leq c_j, \quad j = 1, \ldots, n\}$$

Finding an element of $Y$ can be thought of as being equivalent to solving a LP problem, though this requires a bit of discussion.
Two Important Assumptions

(A1) There is a vector $y^0 \in R^m$ and a scalar $R > 0$ such that the closed ball $S(y^0, R)$ with center $y^0$ and radius $R$

$$S(y^0, R) := \{ y \in R^m : \| y - y^0 \|_2 \leq R \}$$

contains $Y$.

(A2) There is a known scalar $r > 0$ such that if $Y$ is nonempty, then it contains a ball of the form $S(y^*, r)$ with center at $y^*$ and radius $r$.

Note that this assumption implies that if $Y$ is nonempty then it has a nonempty interior.
At each iteration of the algorithm, we will have $Y \subset E_k$. It is then possible to check whether $y^k \in Y$. If so, we have found an element of $Y$ as required. If not, there is at least one constraint that is violated.

Suppose $a_j^T y^k > c_j$. Then

$$Y \subset \frac{1}{2} E_k := \{y \in E_k : a_j^T y \leq a_j^T y^k\}$$

This set is a “half ellipsoid” of $E_k$ cut through its center.
The Ellipsoid Algorithm

Input: $A \in R^{m \times n}$, $c \in R^n$, $y^0 \in R^m$ such that $Y$ (as defined on Slides 3-4) satisfies (A1) and (A2).

Output: $y \in Y$.

Initialization: Set $B_0 = \frac{1}{R^2}I$, $K = \lceil 2m(m + 1) \log(R/r) \rceil + 1$.

For $k = 0, 1, \ldots, K - 1$ do

Iteration $k$: If $y^k \in Y$, STOP: result is $y = y^k$. Otherwise, choose $j$ with $a_j^T y^k > c_j$ and form the half ellipsoid; and update $y^k$ and $B_k$ as described earlier.
Performance of the Ellipsoid Method

Under the assumptions stated above, the ellipsoid method solves linear programs in a polynomially bounded number of iterations bounded by $O \left( m^2 \log \left( \frac{R}{r} \right) \right)$ and each iteration uses $O(m^2)$ arithmetic operations.

Computational experience shows that the number of iterations required to solve a LP problem is very close to the theoretical upper bound. This means that the method is inefficient in a practical sense.

In contrast to this, although the simplex method is known to exhibit exponential behavior on specially constructed problems such as those of Klee and Minty, it normally requires a number of iterations that is a small multiple of the number of linear equations in the standard form of the problem.
Linear Programming (LP)

\[
\begin{align*}
\text{(P)} & \quad \max & c^T x \\
& \text{subject to} & Ax \leq b \\
& & x \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{(D)} & \quad \min & b^T y \\
& \text{subject to} & A^T y \geq c \\
& & y \geq 0
\end{align*}
\]

By the Weak Duality Lemma\(^a\), we have

\[c^T x \leq b^T y.\]

\(^a\)LY Chapter 4.2
Next, we assume that the data for the problem are all integers. As a measure of the size of the problem above we let $c_j = a_{0j}$ and define

$$L = \sum_{i=0}^{m} \sum_{j=1}^{n} \left\lceil \log_2 ( \mod a_{ij} + 1 ) + 1 \right\rceil.$$

In our discussion above, we made two assumptions about $Y$. One of the assumptions, (A2), effectively says that if $Y$ is nonempty, then it possesses a nonempty interior. The linear inequalities are relaxed to

$$a_j^T y < c_j + 2^{-L} \quad j = 1, \ldots, n. \quad (3)$$

It was shown by Gács and Lovasz (1981) that if the inequality system (3) has a solution, then so does

$$a_j^T y \leq c_j, \quad j = 1, \ldots, n.$$
Therefore, we can bound

\[ r \geq 2^{-L}. \]

On the other hand, we can bound

\[ R \leq O(2^L). \]

Thus,

\[ \log(R/r) \leq O(L), \]

which is linear (polynomial) in \( L \).
The Sliding Objective Hyperplane Method

Consider

$$\min b^T y$$

(D) subject to $$A^T y \geq c$$
$$y \geq 0$$

At the center $$y^k$$ of the ellipsoid, if a constraint is violated then add the corresponding constraint hyperplane as the cut; otherwise, add objective hyperplane

$$b^T y \geq b^T y^k$$

as the cut.
Desired Theoretical Properties

- **Separation Problem**: either decide $x \in P$ or find a vector $d$ such that $d^T x \leq d^T y$ for all $y \in P$.
- **Oracle** to generate $d$ without enumerating all hyperplanes.

**Theorem 3** If the separating (oracle) problem can be solved in polynomial time of $m$ and $\log(R/r)$, then we can solve the standard linear programming problem whose running time is polynomial in $m$ and $\log(R/r)$ that is independent of $n$, the number of inequality constraints.
LP with an Exponentially Large Number of Inequalities: TSP

Travelling Salesman Problem (TSP): given an undirected graph $G = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of $n$ nodes and length $c_e$ for every edge $e \in \mathcal{E}$, the goal is to find a tour (a cycle that visits all nodes) of minimal length.

To model the problem, we define for every edge $e$ a variable $x_e$, which is 1 if $e$ is in the tour and 0 otherwise. Let $\delta(i)$ be the set of edges incident to node $i$, then

$$\sum_{e \in \delta(i)} x_e = 2, \ \forall i \in \mathcal{N}.$$  

Let $S \subset \mathcal{N}$ and

$$\delta(S) = \{ e : e = (i, j), i \in S, j \notin S \}.$$  

Then,

$$\sum_{e \in \delta(S)} x_e \geq 2, \ \forall S \subset \mathcal{N}, S \neq \emptyset, \mathcal{N}.$$
LP Relaxation of TSP

\[
\text{(TSP)} \quad \begin{aligned}
\min & \quad \sum_{e \in \mathcal{E}} c_e x_e \\
\text{subject to} & \quad \sum_{e \in \delta(i)} x_e = 2, \quad \forall i \in \mathcal{N}, \\
& \quad \sum_{e \in \delta(S)} x_e \geq 2, \quad \forall S \subset \mathcal{N}, \quad S \neq \emptyset, \mathcal{N}, \\
& \quad 0 \leq x_e \leq 1 \quad \forall e \in \mathcal{E}.
\end{aligned}
\]

This problem has an exponential number of inequalities since there are \(2^n - 2\) of proper subsets of \(S\).
Oracle to Check the Separation

Given $x_e^*$, we would like to check if

$$
\sum_{e \in \delta(S)} x_e^* \geq 2, \ \forall S \subseteq \mathcal{N}, \ S \neq \emptyset, \mathcal{N}.
$$

Assign $x_e^*$ as the capacity for every edge $e \in \mathcal{E}$, then the problem is to check if the min-cut of the graph is greater than or equal to 2.

This problem can be formulated as Maximum Flow problems (how?) and can be solved as a small LP.
The ellipsoid method can be used to find an element of a convex set $Y$ given by a system of convex inequalities.

$$Y = \{ y \in \mathbb{R}^m : f_j(y) \leq 0, \quad j = 1, \ldots n \}$$

where each $f_j(y)$ is a continuous convex function.

Finding an element of $Y$ can be thought of as being equivalent to solving a convex optimization problem when its sub-gradient vector is computable.

How to generate a separation hyperplane?