# Ellipsoid Method

Anoushka Vyas 20171057

Presentation Link

## I. Introduction

The Ellipsoid Method is an iterative method for minimizing convex functions. The first algorithm proposed for linear programming was the Simplex Method, but no variant of it is known to be polynomial time. The Ellipsoid Method is the first polynomial time algorithm discovered for linear programming. The method generates a set of ellipsoids with decreasing volume after every iteration, thus closing on a minimizer of a convex function (Fig. 1.).

The term paper is organised in the following way, section II describes the history of the development of the method in brief, section III describes the problem statement mathematically, section IV explains the Basic Ellipsoid Algorithm and the theorems and proves related to the algorithm, section V explains combinatorial optimization in detail, section VI has the applications of Ellipsoid Method, section VII concludes the term paper and section A in appendix contains the Simplex Method as part of the background study.

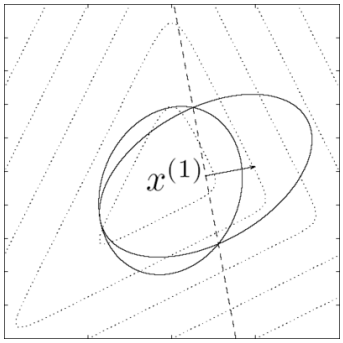


Fig. 1. Volume of ellipsoid decreases (Image Courtesy [1])

## II. Background

The Russian mathematician Leonid G. Khachiyan published his famous paper [2] where he showed that linear programs can be solved efficiently as part of the polynomially solvable problems. Khachiyan's approach was based on ideas similar to the Ellipsoid Method arising from convex optimization. These methods were developed by David Yudin and Arkadi Nemirovskii ([3], [4], [5]), and originally by Naum Shor ([6]). Khachiyan's effort was to modify existing methods enabling him to prove the polynomial running time of his proposed algorithm.

The Ellipsoid Method was the only algorithm for solving linear programs whose runtime had been proved to be polynomial until Karmarkar's algorithm ([7]) was introduced. Karmarkar's interior-point method and variants of the simplex algorithm are much faster than the Ellipsoid Method in practice even in the worst case.

## III. Problem Statement

The Ellipsoid Method is designed to solve decision problems rather than optimization problems.Therefore, we first consider the decision problem of finding a feasible point to a system of linear inequalities

$$C^T x \leq d \tag{1}$$

where $C$ is a $n \times m$ matrix and $d$ is an $n$-dimensional vector. We assume all data to be integral and $n$ to be greater or equal than 2. The goal is to find a vector $x \in R^n$ satisfying (1) or to prove that no such $x$ exists. This problem is equivalent to a linear programming optimization problem as shown in section VI-B of the form

$$\begin{aligned} \min_{x} \quad & q^T x \\ s.t. \quad & C^T x \leq d, \\ & x \geq 0, \end{aligned} \tag{2}$$

in a way that both problems can be solved interchangeably.

## IV. Basic Ellipsoid Algorithm

### A. Overview

The problem starts with a big ellipsoid $E$ that is guaranteed to contain $P$ where

$$P = \{x \in R^n : Cx \leq d\}. \tag{3}$$

If the center of the ellipsoid is in $P$, then the algorithm is stopped Otherwise, we find an inequality $c_i^T x \leq d_i$ ($c_i$ is the $i^{th}$ row of matrix $C$) which is satisfied by all points in $P$ but not satisfied by the center. One iteration of the ellipsoid algorithm is illustrated in Fig. 2.

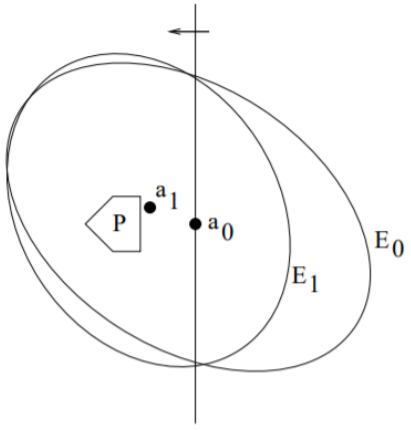


Fig. 2. One iteration of the ellipsoid algorithm (Image Courtesy [1])

The ellipsoid algorithm can be written as:

- $E_0$ is ellipsoid containing $P$
- while center $a_k$ of $E_k$ is not in $P$ do:
  1) Let $c_i^T x \leq c_i^T a_k$ such that $P \subseteq \{x : c_i^T x \leq c_i^T a_k\}$
  2) Let $E_{k+1}$ be the minimum volume ellipsoid containing $E \cap \{x : c_i^T x \leq c_i^T a_k\}$
  3) $k = k + 1$

The ellipsoid algorithm has the important property that the ellipsoids constructed shrink in volume as the algorithm proceeds. This means that if the set $P$ has positive volume, we will eventually find a point in $P$.

### B. Theoretical Analysis

Let us first define an ellipsoid as

*Lemma 4.1:* Given a center $a$, and a positive definite matrix $A$, the ellipsoid $E(a, A)$ is defined as $\{x \in R^n : (x-a)^T A^{-1}(x-a) \leq 1\}$.

For the positive definite matrix $A$ is there exists $B$ such that $A = B^T B$, and hence $A^{-1} = B^{-1}(B^{-1})^T$. Thus, it can be said that ellipsoids are affine transformations of unit spheres.

*Lemma 4.2:* $\frac{Vol(E_{k+1})}{Vol(E_k)} < e^{-\frac{1}{2(n+1)}}$

*Proof:* The volume of an ellipsoid is proportional to the product of its side lengths. Hence the ratio between the unit ellipsoid $E_k$ and $E_{k+1}$ is

$$\begin{aligned} \frac{Vol(E_{k+1})}{Vol(E_k)} &= \frac{(\frac{n}{n+1})(\frac{n^2}{n^2-1})^{\frac{n-1}{2}}}{1} \\ &= (\frac{n}{n+1})(\frac{n^2}{n^2-1})^{\frac{n-1}{2}} \\ &< e^{-\frac{1}{n+1}} e^{\frac{n-1}{2(n^2-1)}} \\ &= e^{-\frac{1}{n+1}} e^{\frac{1}{2(n+1)}} \\ &= e^{-\frac{1}{2(n+1)}} \end{aligned} \tag{4}$$

where the property $1 + x \leq e^x$ for all $x$, with strict inequality of $x \neq 0$ is used.

Let $a_k$ be the center of $E_k$, and $c_i^T x \leq c_i^T a_k$ be the halfspace through $a_k$ that contains $P$. Therefore, the half-ellipsoid that we are trying to contain is $E(a_k, A) \cap \{x : c_i^T x \leq c_i^T a_k\}$. The following affine transformation $T$ defined by $y = T(x) = (B^{-1})^T(x - a)$ is applied to $E(a_k, A)$ as

$$\begin{aligned} \{x : c_i^T x \leq c_i^T a_k\} &\xrightarrow{T} \{y : c_i^T (a_k + B^T y) \leq c_i^T a_k\} \\ &= \{y : c_i^T B^T y \leq 0\} \\ &= \{y : l^T x \leq 0\} \end{aligned} \tag{5}$$

where $l$ is given by

$$l = \frac{Bc_i}{\sqrt{c_i^T B^T B c_i}} = \frac{Bc_i}{\sqrt{c_i^T A c_i}}. \tag{6}$$

Substituting $b = B^T l = \frac{Ac_i}{\sqrt{c_i^T A c_i}}$,

$$\begin{aligned} E_{k+1} &= E(a_k - \frac{1}{n+1}b, \frac{n^2}{n^2-1}B^T(I - \frac{2}{n+1}ll^T)B) \\ &= E(a_k - \frac{1}{n+1}b, \frac{n^2}{n^2-1}(A - \frac{2}{n+1}bb^T)). \end{aligned} \tag{7}$$

The final Ellipsoid Method can be written as:

- $k = 0$, $E_0 = E(a_0, A_0) \supseteq P$, $P = \{x : Cx \leq d\}$
- while $a_k \notin P$ do:
  1) Let $c_i^T x \leq d$ be an inequality that is valid for all $x \in P$ but $c_i^T a_k > d$
  2) $A_{k+1} = \frac{n^2}{n^2-1}(A_k - \frac{2}{n+1}bb^T)$
  3) $a_{k+1} = a_k - \frac{1}{n+1}b$
  4) $k = k + 1$

## V. Combinatorial Optimization

In combinatorial optimization a set $S \subseteq \{0,1\}^n$ is given (for example the incidence vectors of all matchings in a graph) and the task is to optimize over $P = conv(S)$ ($conv$ determines the minimal feasible convex set that contains the convex set $S$ and is referred to as *convex hull*). $P$ is assumed to be full-dimensional or several variables can be eliminated to obtain a smaller full-dimensional problem.

### A. Optimization

Given a polytope (a geometric object with "flat" sides), the aim is to find a feasible point in it using some optimization problem. Let $c_i^T x$ with $c \in R^n$ be the objective function over $P$. Assuming that $c \in Z^n$, we can check the non-emptiness of

$$P' = P \cap \{x : c_i^T x \leq d + \frac{1}{2}\} \tag{8}$$

where $d \in Z$ and $d$ has to be minimum.

As $S \subseteq \{0,1\}^n$, $d$ takes values in the range $[-nc_{max}, nc_{max}]$ where $c_{max} = max_i c_i$. $d$ can be found using binary search and the emptiness of $P'$ can be checked through the ellipsoid method. The final complexity of the algorithm will be polynomial number of steps which is, $O(log(nc_{max})) = O(logn + logc_{max})$.

### B. Initilization

To check the emptiness of $P'$ we need to use ellipsoid method and to perform it we need to initialise an ellipsoid for the zeroth iteration. As starting ellipsoid, we can use the ball centered at the vector $(\frac{1}{2}, \frac{1}{2}, ..., \frac{1}{2})$ and of radius $\frac{1}{2}\sqrt{n}$, which goes through all the incident vectors. The ball has volume of

$$Vol(E_0) = \frac{1}{2^n}(\sqrt{n})^n Vol(B_n), \tag{9}$$

where $B_n$ is the unit ball which has a value

$$Vol(B_n) = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}. \tag{10}$$

On taking logarithm on both sides in equation (9), the following complexity is obtained $log(Vol(E_0)) = O(nlogn)$.

### C. Termination Criteria

Let $P'$ be non-empty and say $v_0 \in P' \cap \{0,1\}^n$, and $v_1, v_2, ..., v_n \in P \cap \{0,1\}^n = S$ be full dimensional. The $v_i$'s are not necessarily in $P'$, hence define $w_i$ such that

$$w_i = \begin{cases} v_i & if \quad c_i^T x \leq d + \frac{1}{2} \\ v_0 + \alpha(v_i - v_0) & otherwise \end{cases}$$

where $\alpha = \frac{1}{2nc_{max}}$. Now checking whether $w_i \in P'$ as

$$\begin{aligned} c^T w_i &= c^T v_0 + \alpha c^T (v_i - v_0) \\ &\leq d + \frac{1}{2nc_{max}} nc_{max} \\ &= d + \frac{1}{2} \end{aligned} \tag{11}$$

This implies that $w_i \in P'$.

Now, $P'$ can be a written as the convex set of $(v_0, w_1, w_2, ..., w_n)$. When a parallelepiped is formed by subtracting $v_0$ from $w_i$ as

$$w_i - v_0 = \beta_i (v_i - v_0) \tag{12}$$

where $\beta_i \in \{\alpha, 1\}$ for $i = 1, 2, ..., n$, then its volume is $n!$ times the volume of the above convex set. The parallellepiped has volume equal to the product of the $\beta_i$ (which is at least $\alpha^n$) times the volume of a parallelepiped with integer vertices, which is at least 1. Thus,

$$Vol(P') \geq Vol(Convexset) = \frac{1}{n!} (\frac{1}{2nc_{max}})^n \tag{13}$$

The number of iterations of the Ellipsoid Method before either discovering that $P'$ is empty or reaching a feasible point is at most

$$log(Vol(E_0)) - log(Vol(P')) = O(nlogn + nlogc_{max}). \tag{14}$$

### D. Separation Oracle

To decide when $x \in R^n$ is in $P'$, we need to find a Separation Oracle for $P$.

*Lemma 5.1:* A polynomial time Separation Oracle for a convex set P is a procedure which given $x$, either tells that $x \in P$ or returns a hyperplane separating $x$ from $P$. The procedure should run in polynomial time.

### E. Optimum Solution

The algorithm gives a point $x^* \in P$ of value at most $d + \frac{1}{2}$. However, we are interested in finding a point $x \in P \cap \{0,1\}^n = S$ of value exactly $d$. To do this start from $x^*$ which we got and find an extreme point $x \in P$ such that $c^T x \leq c^T x^*$.

*Theorem 5.2:* Let $S = \{0,1\}^n$ and $P = conv(S)$. Assume that $P$ is full-dimensional and we are given a separation oracle for $P$. Then, given $c \in Z^n$, one can find $min\{c^T x : x \in S\}$ by the Ellipsoid Method by using a polynomial number of operations and calls to the separation oracle.
*Proof:* The proof can be found in [8].

The number of iterations of the Ellipsoid Method for combinatorial optimization is $O(n^2 log^2 n + n^2 log^2 c_{max})$, each iteration requiring a call to the Separation Oracle and a polynomial number of operations.

## VI. Applications

### A. Minimum Cost Arborescence Problem

In a typical Network Design or Network Optimization setting, the task is to define a network infrastructure and a corresponding operating regime that meets a certain set of topological and operative requirements.

Let $G = (V, A)$ be a network where $V$ is the set of nodes and $A$ is the set of arcs. An arborescence (rooted at $r$) is a tree $T$ such that $T$ is a spanning tree of $G$ if we ignore the direction of edges. There is a directed unique path in $T$ from $r$ to each other node $v \in V$.

The minimum cost arborescence problem relies on solving a primal-dual method

$$\begin{aligned} min \quad & \sum_{a \in A} c_a x_a \\ s.t. \quad & \\ & \sum_{a \in \delta^-(S)} x_a \geq 1 \quad \forall S \subseteq V \backslash \{r\} \\ & x_a \geq 0 \quad a \in A \end{aligned} \tag{15}$$

where for a given arbitrary subset of nodes $S \subseteq V$, $\delta^-(S) = \{(i,j) \in A | i \in V \backslash S, j \in S\}$ and $c : A \rightarrow R_{\geq 0}$ is the sharing cost function, such that $c_a$ is the cost of taking $a \in A$ as part of the shared network (e.g., the installation cost).

To solve this algorithm using Ellipsoid Method in polynomial time, a Separation Oracle for it can be easily defined as if $x_a^* < 0$ for some $a \in A$, just return the inequality $x_a \geq 0$. Otherwise, for every $t \in V \backslash \{r\}$, consider the minimum $r - t$ cut problem in which the capacity on arc $a$ is given by $x_a^*$. This can be solved by maximum flow problem.

If for some $t \in V \backslash \{r\}$, the minimum $r - t$ cut has value less than 1 then we have found a violated inequality. Otherwise, the feasible point $x_a^* \in P$ is found. The Separation Oracle can be implemented by doing $|V| - 1$ maximum flow computations, and hence is polynomial.

### B. Linear Programming

The Ellipsoid Method solves the problem of finding a feasible point of a system of linear inequalities. This problem is closely related to the problem of solving (2) (data is integers).

There are two methods to solve the optimization problem (2) via the Ellipsoid Method. This will show that LP is in the class of polynomially solvable problems.

*1) Using Duality Theorem:* From duality theory, the linear optimization problem (2) is equivalent to finding a feasible point of the following system of linear inequalities

$$\begin{aligned} C^T x &\leq d \\ -x &\leq 0 \\ -C^T y &\leq q \\ -y &\leq 0 \\ -q^T x + d &\leq 0. \end{aligned} \tag{16}$$

The third and fourth inequality come from the dual problem of (2), and the last inequality results from the Strong Duality Theorem, which means zero slackness.

From the equivalence of the two problems (2) and (16), it can be concluded that the linear programming problem can be solved in polynomial time as the input data of (16) is polynomially bounded in the length of the input data of (2).

Using the above inequalities, both the primal ad dual problem is solved together using the Ellipsoid Method. However, the dimension of the problem increases from $n$ to $n+m$.

*2) Using Bisection Method:* This is also known as binary search or sliding objective hyperplane method. The algorithm generally involves starting with an upper and lower bound of an optimal solution and making the difference between the bounds smaller until they are zero or small enough.

To find the bounds we can use the solutions from the primal and dual problem by solving (2) with the Ellipsoid Method to get the lower bound unless it is infeasible and solving (16) or weak duality theorem to get an upper bound. If the Ellipsoid Method proves that the polytope of the dual problem is empty, we can use the duality theory to conclude that the optimization problem (2) is unbounded.

Once bounds are obtained, the problem can be solved with Ellipsoid Method iteratively using the additional constraint

$$-q^T x \geq -\frac{u+l}{2}, \tag{17}$$

where $l$ and $u$ are lower and upper bounds respectively.

If the new problem is infeasible, the upper bound is updated to $\frac{u+l}{2}$, and this happens iteratively until the solution is found.

### C. Separation and Optimization

A property of the Ellipsoid Method is that it does not require an explicit list of all inequalities. In fact, it is enough to have a routine which solves the separation problem for $P$ (as shown in *Lemma 5.1*). From the Basic Ellipsoid Algorithm in section IV indicates that if one can solve the separation problem for polytope $P$, then the corresponding optimization problem (2) can also be solved in polynomial time.

*Lemma 6.1:* The separation problem and the optimization problem over the same family of polytopes are polynomially equivalent.

An example is the maximum stable set problem. Given a graph $G = (V, E)$ with node set $V$ and edges $e \in E$. A stable set $S$ of graph $G$ is defined as a subset of $V$ with the property that no two nodes of $S$ are adjacent meaning no edge between them exists in $E$. This is an $NP-hard$ optimization problem and it can be modeled as integer programming problem

$$\begin{aligned} max \quad & c^T x \\ s.t. \quad & x_i + x_j \leq 1 \quad \forall (i,j) \in E \\ & x \in \{0,1\} \end{aligned} \tag{18}$$

where $x_i = 1$ iff node i is in a maximum stable set, otherwise it is zero.

Relaxing the binary constraints for $x$ to create a LP problem

$$0 \leq x \leq 1, \tag{19}$$

and considering the odd-cycle inequalities

$$\sum_{i \in C} x_i \leq \frac{|C|-1}{2} \tag{20}$$

for each odd cycle $C$ in $G$, we can write the polytope which satisfies all these odd cycle inequalities as

$$P := \{x \in R^{|V|} | x \; satisfies (18), (19) \; and \; .(20)\} \tag{21}$$

This is called the cycle-constraint stable set polytope.

To show that (21) can be solved in polynomial time, an auxiliary graph $H$ is created with a double number of nodes. Solving a sequence of $n$ shortest path problems on $H$ solves then the separation problem with a total running time of order$|V||E|log(|V|)$. Now through the *Lemma 6.1* the stable set problem over the cycle-constraint stable set polytope can be solved in polynomial time.

This does not imply that the solution will be integral and hence, we cannot conclude that the stable set problem can be solved in polynomial time. But we can conclude that the stable set problem for graphs whose stable set polytope is equal to the cycle-constraint stable set polytope can be solved in polynomial time.

## VII. Summary

In the term paper, we have explored the Ellipsoid Method and the theoretical analysis behind the method. It is the first polynomial time algorithm discovered for linear programming. However, it has a few of disadvantages, the rate of convergence of the Ellipsoid Method is rather slow, especially when compared to practical experience with the simplex method. The Ellipsoid Method does not provide optimal dual variables. However, it is a powerful theoretical tool in the analysis of the computational complexity of combinatorial optimization problems.

## Appendix A
## Simplex Method

In this section, we cover the basics of solving a linear optimization problem by Simplex Tableau Method. For simplicity, we will learn how to perform this by an example given as:

$$\begin{aligned} \textsf{Maximize} \quad & Z = 40x_1 + 30x_2 \\ \textsf{s. t.} \quad & x_1 + x_2 \leq 12 \\ & 2x_1 + x_2 \leq 16 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{22}$$

It is important to ensure that the optimization problem is in standard form.

The inequalities are now converted to equations by adding slack variables. Let $s_1$ and $s_2$ be two slack variables which represent how much short the LHS is from RHS in the inequalities. The new optimization problem now is

$$\begin{aligned} \text{Objective} \quad & Z - 40x_1 - 30x_2 = 0 \\ \text{s. t.} \quad & x_1 + x_2 + s_1 = 12 \\ & 2x_1 + x_2 + s_2 = 16 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{23}$$

Now the initial simplex tableau is constructed where each inequality is in one row (column $C$ is the RHS of the equations).

| $x_1$ | $x_2$ | $s_1$ | $s_2$ | $Z$ | $C$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 12 |
| 2 | 1 | 0 | 1 | 0 | 16 |
| $-40$ | $-30$ | 0 | 0 | 1 | 0 |

The last four columns of this table look like the final matrix for the solution of a system of equations if we arbitrarily choose $x_1 = 0$ and $x_2 = 0$, we get

$$\left[\begin{array}{cccc|c} s_1 & s_2 & Z & & C \\ 1 & 0 & 0 & | & 12 \\ 0 & 1 & 0 & | & 16 \\ 0 & 0 & 1 & | & 0 \end{array}\right] \tag{24}$$

The solution obtained by arbitrarily assigning values to some variables and then solving for the remaining variables is called the basic solution associated with the tableau.

The most negative entry in the bottom row identifies the pivot column. The most negative entry in the bottom row is $-40$; therefore the column 1 is identified. The most negative entry in the bottom row represents the largest coefficient in the objective function - the coefficient whose entry will increase the value of the objective function the quickest.

The next step is to find the pivot element. We need to do so because we are trying to maximize the objective function and we choosing the $x_1$ column to do so. But we cannot choose any value for $x_1$, we need to choose such a value which does not violate constraint. To perform the same, we divide the entries in the far right column by the entries in column 1, excluding the entry in the bottom row. The values we get are 12 and 8. Therefore row 2 is identified. The intersection of column 1 and row 2 is the entry 2, which is the pivot element.

The simplex method begins with a corner point and then moves to the next corner point always improving the value of the objective function. The value of the objective function is improved by changing the number of units of the variables. We may add the number of units of one variable, while throwing away the units of another. Pivoting allows us to do just that.

The variable whose units are being added is called the entering variable, and the variable whose units are being replaced is called the departing variable. The entering variable in the above table is $x_1$ , and it was identified by the most negative entry in the bottom row. The departing variable $s_2$ was identified by the lowest of all quotients (8).

Pivoting is used to obtain 1 in the location of the pivot element, and then making all other entries zeros in that column. The entire second row is divided by 2.

| $x_1$ | $x_2$ | $s_1$ | $s_2$ | $Z$ | $C$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 12 |
| 1 | 1/2 | 0 | 1/2 | 0 | 8 |
| $-40$ | $-30$ | 0 | 0 | 1 | 0 |

To obtain a zero in the entry first above the pivot element, we multiply the second row by $-1$ and add it to row 1.

| $x_1$ | $x_2$ | $s_1$ | $s_2$ | $Z$ | $C$ |
|---|---|---|---|---|---|
| 0 | 1/2 | 1 | $-1/2$ | 0 | 4 |
| 1 | 1/2 | 0 | 1/2 | 0 | 8 |
| $-40$ | $-30$ | 0 | 0 | 1 | 0 |

To obtain a zero in the element below the pivot, we multiply the second row by 40 and add it to the last row.

| $x_1$ | $x_2$ | $s_1$ | $s_2$ | $Z$ | $C$ |
|---|---|---|---|---|---|
| 0 | 1/2 | 1 | $-1/2$ | 0 | 4 |
| 1 | 1/2 | 0 | 1/2 | 0 | 8 |
| 0 | $-10$ | 0 | 20 | 1 | 320 |

There is still a negative entry in the bottom row, $-10$. Thus, the column $x_2$ is the pivot column and the first row is the pivot row as its coefficient is 8 while the coefficient of second row is 16. Thus, the pivot element is $\frac{1}{2}$, the intersection of column 2 and row 1.

To make the pivot element 1 multiply row 1 by 2.

| $x_1$ | $x_2$ | $s_1$ | $s_2$ | $Z$ | $C$ |
|---|---|---|---|---|---|
| 0 | 1 | 2 | $-1$ | 0 | 8 |
| 1 | 1/2 | 0 | 1/2 | 0 | 8 |
| 0 | $-10$ | 0 | 20 | 1 | 320 |

To make all other entries as zeros in this column, we first multiply row 1 by $-\frac{1}{2}$ and add it to row 2, and then multiply row 1 by 10 and add it to the bottom row.

| $x_1$ | $x_2$ | $s_1$ | $s_2$ | $Z$ | $C$ |
|---|---|---|---|---|---|
| 0 | 1 | 2 | $-1$ | 0 | 8 |
| 1 | 0 | $-1$ | 1 | 0 | 4 |
| 0 | 0 | 20 | 10 | 1 | 400 |

Now, there are no negative values in the bottom row. To determine the basic solution from the final table, consider only the rows which has a 1 and all other zeros. From the table, we can see that columns $s_1$ and $s_2$ are not those columns thus their values can be taken 0. Thus, $x_1 = 4$ and $x_2 = 8$ is the final answer and the value of the objective function is $Z = 400$.

Fig. 3. and Fig. 4. shows the graphical way of solving the problem (solved using PHP Simplex).

## REFERENCES


[1] "Ellipsoid method," [Online]. Available: https : / / en . wikipedia.org/wiki/Ellipsoid_method.

[2] L. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980, ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(80)90061-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0041555380900610.

[3] D. B. Yudin and A. S. Nemirovskii, "Evaluation of the informational complexity of mathematical programming problems," *Central Economic Mathematical Institute of the Academy of Sciences of the USSR*, vol. 13, no. 2, pp. 3–25, 1976-7.

[4] Yudin and Nemirovskii, "Informational complexity and efficient methods for the solution of convex extremal problems," *Central Economic Mathematical Institute of the Academy of Sciences of the USSR*, vol. 13, no. 3, pp. 25–45, 1977.

[5] D. Yudin and A. Nemirovskii, "Optimization methods adapting to the "significant" dimension of the problem," *Automation and Remote Control*, vol. 38, no. 4, pp. 75–87, 1977.

[6] N. Z. Shor, "Cut-off method with space extension in convex programming problems," *Cybernetics*, vol. 13, pp. 94–96, 1977.

[7] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '84, New York, NY, USA: Association for Computing Machinery, 1984, pp. 302–311, ISBN: 0897911334. DOI: 10.1145/800057.808695. [Online]. Available: https://doi.org/10.1145/800057.808695.

[8] S. Grotschel and Lovasz, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, 1981. [Online]. Available: https://doi.org/10.1007/BF02579273.


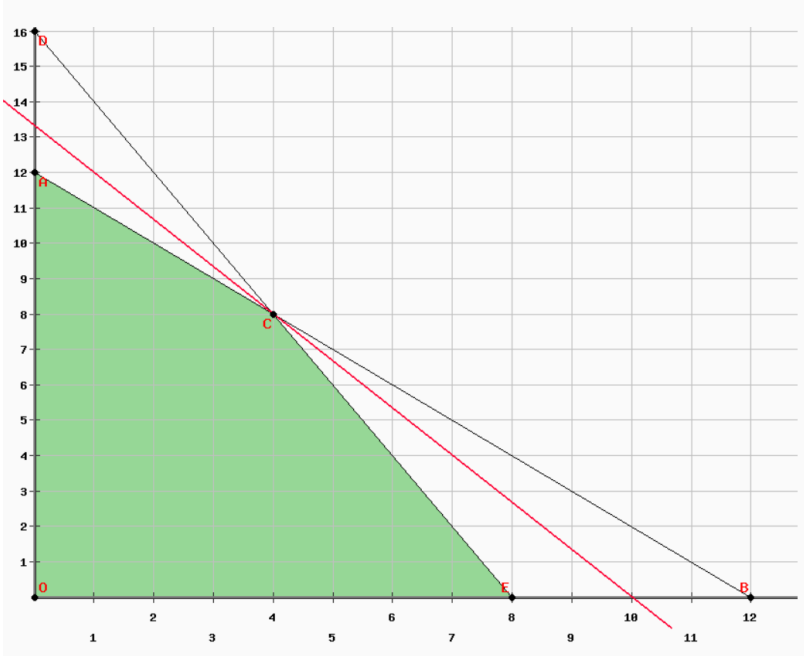


Fig. 3. Graphical representation of the feasible region

| Point | X coordinate ($X_1$) | Y coordinate ($X_2$) | Objective function value (Z) |
|---|---|---|---|
| OR | 0 | 0 | 0 |
| TO | 0 | 12 | 360 |
| B | 12 | 0 | 480 |
| C | 4 | 8 | 400 |
| D | 0 | 16 | 480 |
| AND | 8 | 0 | 320 |

Fig. 4. Coordinates of the feasible region and points of intersection with the axes