Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 46

Hello everyone. This is the first lecture of week 10. In this lecture, we will complete our discussion on the simplex method.

Recall that in the ninth week, we discussed the simplex method in detail. We started with the reasoning why vertices are the only potential solutions in a Linear Programming (LP) problem. We then discussed the simplex method step by step, starting with a basic feasible solution, verifying whether a given vertex is the minimizer, and then exchanging variables between the sets of basic and non-basic variables. Finally, we implemented the algorithm in code.

In this lecture, we will first review the code once more.

If you look at the algorithm we discussed, we first define these variables: B, N, c_B, c_N, B_set, N_set, and b.

We then compute and print x, where $x = [B^{-1}b, 0]$.

After that, we verify the optimality condition. If the condition is not satisfied (i.e., the sum is not equal to zero), then the current vertex is not the minimizer, and we proceed to exchange a variable between the basic and non-basic sets.

This exchange involves welcoming an incoming basic variable from the non-basic set and removing a particular basic variable to the non-basic set. This step performs the pivot operation, and we then print the updated solution.

I would like to suggest one change to avoid potential errors. For exchanging values between two variables in Python, the syntax `a, b = b, a` works well for scalars.

However, for exchanging vectors (like columns of matrices), this might cause issues. In languages like C or C++, such an exchange requires a temporary variable because you cannot assign both variables simultaneously without overwriting. Therefore, for vector exchanges, it is safer to use a temporary variable.

For example:

Python

```
t = B[:, j].copy()
```

B[:, j] = N[:, q]

N[:, q] = t

This ensures a correct exchange without unintended side effects. With this change, the code will run correctly.

We have worked through one example: maximizing x_2 subject to constraints that define a feasible set as a triangle with vertices (0, 0), (1, 0), and (½, ½).

Now, let us consider another example to illustrate the simplex method further and discuss some of its drawbacks.

Example Problem

Minimize: $3x_1 - 2x_2 + x_3$

Subject to:

* $x_1 - 2x_3 \geq 4$
* $2x_1 - x_2 + x_3 \geq 2$
* $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$

This is an LP in three dimensions.

While drawing and finding vertices is possible in three dimensions, it becomes impractical for higher dimensions. Therefore, we will follow the algorithmic approach without relying on geometry.

The first step is to convert the LP into standard form ($Ax = b, x \geq 0$).

We introduce slack variables x_4 and x_5:

* $x_1 - 2x_3 \geq 4$ becomes $-x_1 + 2x_3 + x_4 = -4$, with $x_4 \geq 0$
* $2x_1 - x_2 + x_3 \geq 2$ becomes $-2x_1 + x_2 - x_3 + x_5 = -2$, with $x_5 \geq 0$

The problem in standard form is:

Minimize: $3x_1 - 2x_2 + x_3 + 0x_4 + 0x_5$

Subject to:

$$\begin{bmatrix} -1 & 0 & 2 & 1 & 0 \\ -2 & 1 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -4 \\ -2 \end{bmatrix}$$

With: $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0$

Where:

* $c^T = [3, -2, 1, 0, 0]$
* $A = [\ [-1, 0, 2, 1, 0], [-2, 1, -1, 0, 1]\ ]$
* $b = [-4, -2]$
* $x = [x_1, x_2, x_3, x_4, x_5]^T$

#Finding an Initial Basic Feasible Solution

We need to find two linearly independent columns for the basis matrix B such that $B^{-1}b \geq 0$. Trying columns for x_4 and x_5 (which form the identity matrix) gives:

* B = I, so $x\_B = B^{-1}b = [-4, -2]$, which is not ≥ 0. Not feasible.

Trying columns for x_1 and x_4:

* $B = [\ [-1, 1], [-2, 0]\ ]$
* $B^{-1}b$ yields negative components. Not feasible.

Trying columns for x_1 and x_5:

* $B = [\ [-1, 0], [-2, 1]\ ]$
* $x\_B = B^{-1}b = [4, 6]$, which is ≥ 0. Feasible.

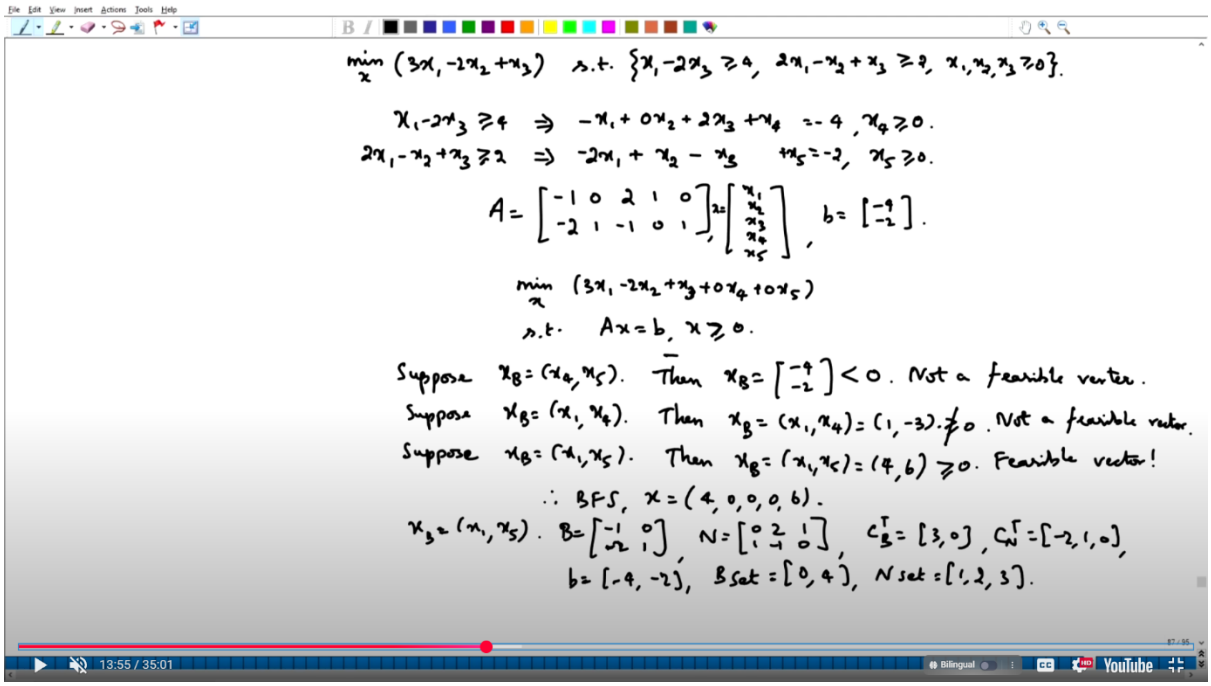
Thus, the initial basic feasible solution is:

* $x\_B = [x_1, x_5] = [4, 6]$
* $x\_N = [x_2, x_3, x_4] = [0, 0, 0]$
* Full solution: $x = [4, 0, 0, 0, 6]$

We define:

* $B = [\ [-1, 0], [-2, 1]\ ]$
* $N = [\ [0, 2, 1], [1, -1, 0]\ ]$
* $c\_B = [3, 0]$
* $c\_N = [-2, 1, 0]$
* $b = [-4, -2]$
* $B\_set = [0, 4]$ (indices of x_1 and x_5)
* $N\_set = [1, 2, 3]$ (indices of x_2, x_3, x_4)

(Refer Slide Time 13:55)



Plugging these into the code, we find that the problem is unbounded. The output shows:

* Initial solution: [4, 0, 0, 0, 6]
* After pivoting: [4, 6, 0, 0, 0]
* Then unbounded.

#Why is the Problem Unbounded?

The constraints are:

1. $x_1 - 2x_3 \geq 4$
2. $2x_1 - x_2 + x_3 \geq 2$

Consider the point:

* $x_3 = x_1/2 - 2$
* $x_2 = 2x_1 - x_3 - 2 = (5x_1)/2 - 4$

For $x_1 \geq 4$, this point satisfies both constraints with equality and all variables are non-negative. Thus, it is feasible.

The objective function is:

$3x_1 - 2x_2 + x_3 = 3x_1 - 2((5x_1)/2 - 4) + (x_1/2 - 2) = -3x_1/2 + 6$

As $x_1 \to \infty$, the objective function $\to -\infty$. Therefore, the problem is unbounded.

Issues with the Simplex Method

1. Finding an Initial Basic Feasible Solution (BFS):

The first step of the simplex method requires a BFS. This involves choosing m linearly independent columns such that $B^{-1}b \geq 0$. For large problems (e.g., 100 variables), this trial-and-error process can be computationally expensive and inefficient.

2. Pathological Example of Simplex Method Complexity

One alternative to the simplex method is to find all vertices of the feasible region, compute the objective function at each vertex, and select the vertex that yields the minimum value. However, this approach was rejected because, in higher-dimensional problems, the number of vertices can be exponentially large, making an exhaustive search impractical. This is why we adopted the simplex method, which uses a heuristic of moving between adjacent vertices to find the optimum.

A natural question arises: what guarantees do we have that the simplex method will not also traverse an exponential number of vertices? Unfortunately, there is no such guarantee. There exist pathological examples where the simplex method visits all 2^n vertices before terminating, where n is the number of variables. This is surprising given that the simplex method is often efficient in practice, but its worst-case performance is poor.

Pathological Example

Consider the following linear programming problem:

Maximize:

$2^{n-1}x_1 + 2^{n-2}x_2 + \ldots + 2x_{n-1} + x_n$

Subject to n constraints:

1. $x_1 \leq 5$
2. $4x_1 + x_2 \leq 25$
3. $8x_1 + 4x_2 + x_3 \leq 125$
4. ...
5. $2^n x_1 + 2^{n-1}x_2 + \ldots + 4x_{n-1} + x_n \leq 5^n$

And $x \geq 0$.

This is a valid linear program. To convert it to standard form, we introduce n slack variables, resulting in 2n variables total. The constraint matrix A is structured as follows:

- The first constraint: $x_1 + x_{n+1} = 5$
- The second constraint: $4x_1 + x_2 + x_{n+2} = 25$
- The third constraint: $8x_1 + 4x_2 + x_3 + x_{n+3} = 125$
- ...

- The n-th constraint: $2^n x_1 + 2^{n-1} x_2 + ... + 4x_{n-1} + x_n + x_{2n} = 5^n$

Thus, the matrix A has the form:

A = [

[1, 0, 0, ..., 0, 1, 0, ..., 0],

[4, 1, 0, ..., 0, 0, 1, ..., 0],

[8, 4, 1, ..., 0, 0, 0, ..., 0],

...

[2^n, 2^{n-1}, ..., 4, 1, 0, 0, ..., 1]

]

The right-hand side vector is:

b = [5, 25, 125, ..., 5^n]

We choose the slack variables as the initial basic variables:

- B_set = [n, n+1, ..., 2n-1] (indices of slack variables)

- N_set = [0, 1, ..., n-1] (indices of original variables)

The basis matrix B is the identity matrix of size n, so $B^{-1}b = b = [5, 25, ..., 5^n] \geq 0$. The initial solution is:

- x_B = [5, 25, ..., 5^n]

- x_N = [0, 0, ..., 0] (original variables)

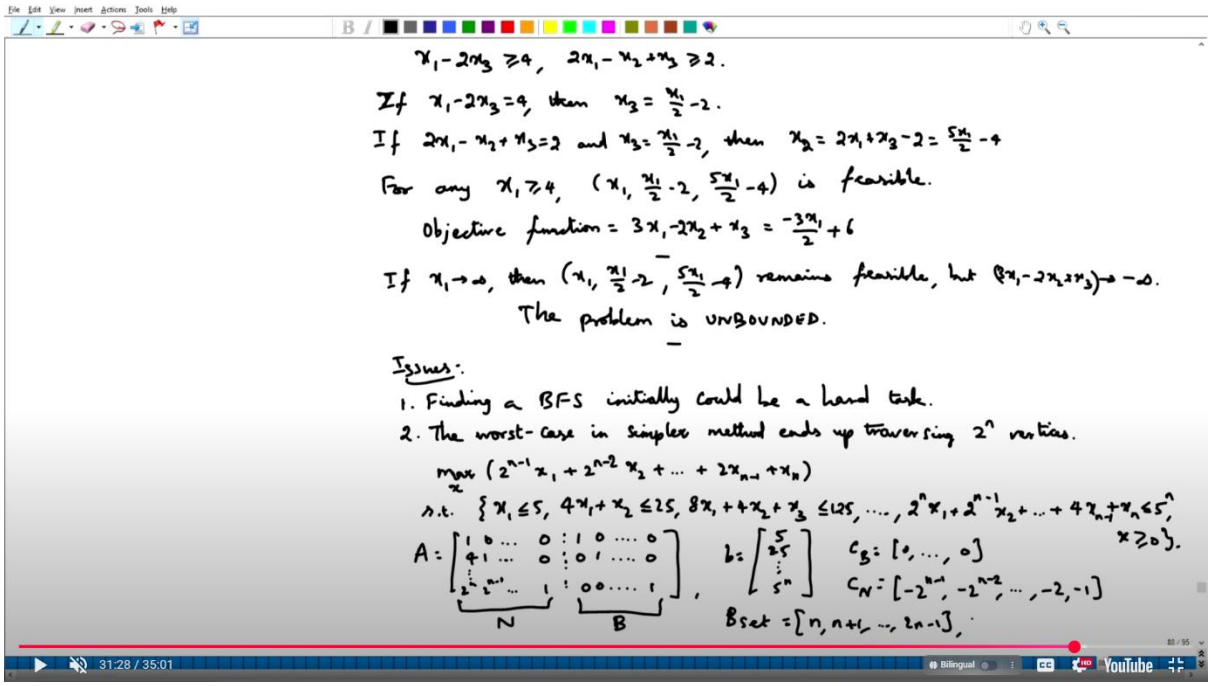
- Full solution: x = [0, 0, ..., 0, 5, 25, ..., 5^n]

The cost vectors are:

- c_B = [0, 0, ..., 0] (slack variables have zero cost)

- c_N = [-2^{n-1}, -2^{n-2}, ..., -2, -1] (negative because we maximize the objective)

(Refer Slide Time 31:28)



Computational Behavior

For n=4:

- B is the 4×4 identity matrix.

- N = [[1, 0, 0, 0], [4, 1, 0, 0], [8, 4, 1, 0], [16, 8, 4, 1]]

- c_N = [-8, -4, -2, -1]

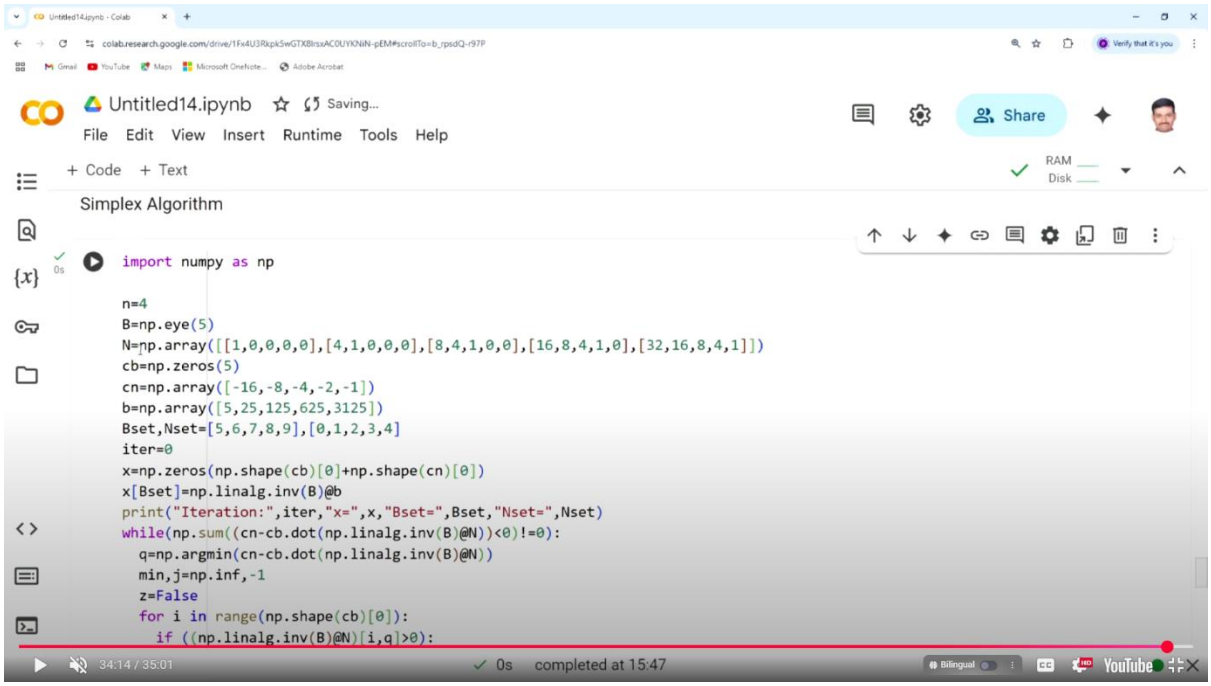
- b = [5, 25, 125, 625]

- B_set = [4, 5, 6, 7]

- N_set = [0, 1, 2, 3]

The simplex method takes $16 = 2^4$ iterations to reach the optimal solution [0, 0, 0, 0, 5, 25, 125, 625].

(Refer Slide Time 31:28)



For n=5:

- The method takes $32 = 2^5$ iterations to reach the optimal solution.

In general, for dimension n, the simplex method requires 2^n iterations. This exponential growth demonstrates the worst-case complexity of the algorithm.

#Conclusion

Due to these two issues—the difficulty of finding an initial basic feasible solution and the exponential worst-case performance—we will explore other algorithms in the next lecture. Thank you.