Math 303 Report #1

Members of team :

Abdalla saed 202201933
Omar Toulba 202202155
Abdalrahman Omer 202202254
Alhassan Ali 202200681

Revised Simplex Method for Solving LPP

1. Motivation:

The revised simplex method is a variant of the simplex method invented by Georg Dantzig for linear programming in the field of mathematical optimization. The revised simplex method describes linear programs as matrix entities and presents the simplex method as a series of linear algebra calculations. Instead of spending time updating dictionaries at the end of each iteration, the revised simplex method does its heavy computations at the beginning of each iteration, which results in a significant reduction in time in the end. As Chvatal notes, "Each iteration of the revised simplex method may or may not take less time than the corresponding iteration of the standard simplex method. The outcome of this comparison depends not only on the particular implementation of the revised simplex method but also on the nature of the data. ... For large, sparse [linear programming] problems to be solved in applications, the revised simplex method runs faster than the standard simplex method. This is why modern computer programs for solving [linear programming] problems almost always use some form of the revised simplex method." The revised simplex method is mathematically equivalent to the standard simplex method but differs in implementation. Instead of maintaining a table that explicitly represents the modified constraints for a set of underlying variables, it maintains a matrix basis representation that represents the constraints. The matrix-oriented approach allows for greater computational efficiency by enabling sparse matrix operations.The original simplex method can be very time-consuming and

memory-intensive when dealing with a problem involving a large number of variables and constraints. Here comes the modified approach, which aims to: Reduce memory consumption and speed up calculations.

2. Theory:

The revised simplex method is based on the same underlying theory as the standard simplex method..

where:

x is the vector of decision variables.

c is the coefficient vector of the objective function.

A is the matrix of coefficients in the constraints.

b is the vector of constants on the right side of the constraints.

First of all, a linear program needs to be expressed in terms of matrices. for instance, the linear program stated previously:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum a_{ij} x_j \le b_i \quad (i = 1,2,\ldots,m) \\ & x_j \ge 0 \quad (j = 1,2,\ldots,n) \end{aligned}$$

can instead be recast in the form:

$$\begin{aligned} \text{minimize} \quad & cx \\ \text{subject to} \quad & Ax \le b \\ & x \ge 0 \end{aligned}$$

where:

$$c = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{11} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{n2} & \cdots & a_{mn} \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Revised Simplex Method
$d = B^{-1}b$
$\bar{c} = c_V - c_B \cdot B^{-1} \cdot V$ $\left\{ j \middle\| \bar{c}_j = \min_t(\bar{c}_t) \right\}$
If $\bar{c}_j \geq 0$, d is optimal solution.
$w = B^{-1} \cdot A_j$ $\left\{ i \middle\| \frac{d_i}{w_i} = \min_t \left(\frac{d_t}{w_t} \right), w_t > 0 \right\}$
If $w_i \leq 0$ for all i, the solution is unbounded.
Update B^{-1}.

3. Algorithm:

Initialize Variables

- Define m as the number of rows in A (constraints) and n as the number of columns in A (variables).

- Identify indices of the basic variables $B_{indices}$ based on the initial feasible solution, and the indices of non-basic variables $V_{indices}$.

Begin Simplex Iteration Loop

- Repeat the following steps until an optimal solution is found or the problem is determined to be unbounded.

Calculate B^{-1}

- Compute the inverse of the basis matrix B formed by columns of A corresponding to B_indices.

Compute $d = B^{-1} . b$

- Calculate the current solution vector d, which gives the values of the basic variables in the current solution.

Compute the Reduced Cost Vector $c^{\sim}$

- Compute the reduced cost vector c for the non-basic variables as: $c = c_V - c_B . B^{-1} . A_V$

- Here, c_V and c_B are cost vectors for non-basic and basic variables, respectively, and A_V is the submatrix of A corresponding to non-basic variables.

Select the Entering Variable j

- Identify the index j of the non-basic variable with the most negative value in c, as this variable has the potential to improve the objective function.

- If the minimum $\tilde{}$c[j] >= 0, then the current solution is optimal, and the algorithm should terminate, returning the solution.

Check Optimality Condition

- If the condition in Step 6 is met (all c[j] >= 0), the algorithm exits, and the current solution vector is optimal.

Compute the Direction Vector $w = B^{-1} . a_j$

- Calculate w, the direction vector, for the entering variable j by multiplying B^{-1} with the column a_j of A that corresponds to j.

Determine the Leaving Variable (Ratio Test)

- Find the basic variable to leave the basis by performing the ratio test:

- For each i where w[i] > 0, compute d[i] / w[i].

- Choose the smallest positive ratio to maintain feasibility.

- Let i be the index of the row with the smallest ratio. If no such i exists, the problem is unbounded.

Update the Basis

- Swap the entering variable j into the basis and the leaving variable (row i) out of the basis.

- Update $B_{indices}$ and $V_{indices}$ to reflect the new basis.

Repeat

- Go back to Step 3 with the updated basis until an optimal solution is found or the problem is determined to be unbounded.

---

4. Implementation

https://colab.research.google.com/drive/1LMdrYq-xW1BzVBitBt70QZuS-8EADsJA?usp=sharing

5. Solving Examples

.https://colab.research.google.com/drive/1LMdrYq-xW1BzVBitBt70QZuS-8EADsJA?usp=sharing

6. Comparison with Python Built-in Function

https://colab.research.google.com/drive/1LMdrYq-xW1BzVBitBt70QZuS-8EADsJA?usp=sharing

Reference:

A comparison of simplex method algorithms. (n.d.). by

Steven S. Morgan

A thesis presented to the Graduate School
of the University of Florida in partial fulfillment
of the requirements for the degree of
Master of Science

University of Florida

1997

https://web.archive.org/web/20110807134509/http://www.cise.ufl.edu/research/sparse/Morgan/index.htm