Optimization Methods PS s7 Lab 4

Joris Plaščinskas

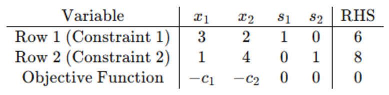
December 8, 2024

Linear Simplex Algorithm

Linear simplex algorithm is an iterative optimization method used to find the optimal values of a linear problem (all constraint and objective function are linear). The algorithm generally follows these steps:

- **Formalize the problem into standard form**. The problem is given in inequalities form. The standard form uses equations, so the inequalities will have to be rewritten. We do this by adding slack variables, each constraint will get it's own s_i variable that represents how far away from being strict it is. If the slack variable is 0, that means the constraint is tight.

- **Initialize tableau**. In the middle are the constraint coefficients, on the right are RHS values. Bottom row is objective function coefficients. And the bottom right cell is objective function value. An example table can be seen below:



Variable	x_1	x_2	s_1	s_2	RHS
Row 1 (Constraint 1)	3	2	1	0	6
Row 2 (Constraint 2)	1	4	0	1	8
Objective Function	$-c_1$	$-c_2$	0	0	0

Figure 1: Tableau Example

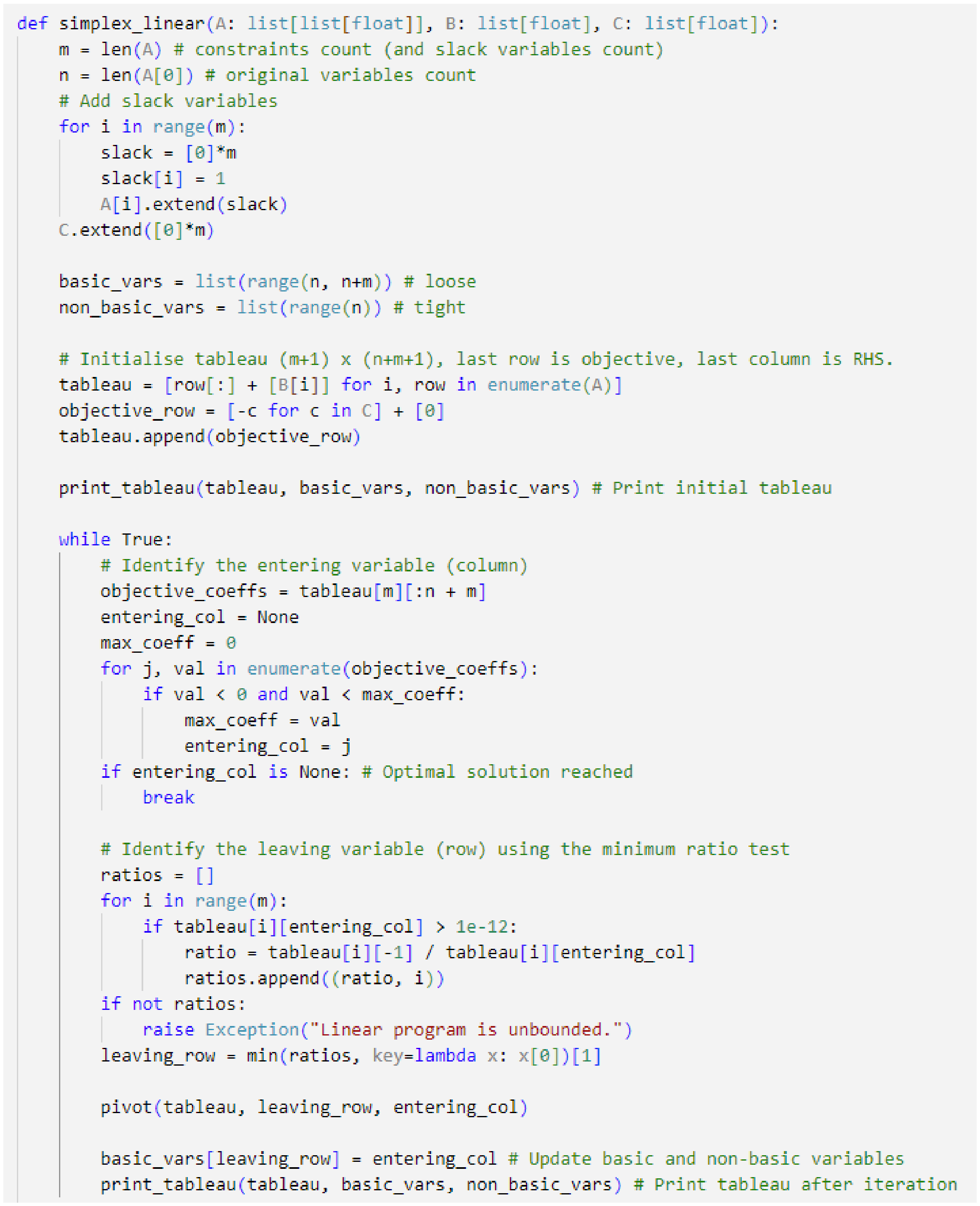
- **While optimal value is not reached, iterate on the below steps**. Optimal value is reached when all of the values in the objective function row in tableau are negative, that means there is no way to increase it more (for maximization problems).

  - **Identify the entering variable**. The entering variable is selected by looking at the bottom (objective) row and selecting the highest negative value.

  - **Identify the leaving variable**. For each row calculate the ratio of RHS of the constraint to the coefficient of the entering variable in that row. Only consider rows where the coeeficient of the entering variable is positive

- **Perform a pivot around the pivot variable**. First we need to divide the pivot row by the pivot element to make the pivot element equal to 1. Then we adjust the other rows to make all other elements in the pivot column equal to 0.

The code can be seen in the image below:



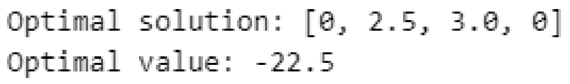
```python
def simplex_linear(A: list[list[float]], B: list[float], C: list[float]):
    m = len(A) # constraints count (and slack variables count)
    n = len(A[0]) # original variables count
    # Add slack variables
    for i in range(m):
        slack = [0]*m
        slack[i] = 1
        A[i].extend(slack)
    C.extend([0]*m)

    basic_vars = list(range(n, n+m)) # loose
    non_basic_vars = list(range(n)) # tight

    # Initialise tableau (m+1) x (n+m+1), last row is objective, last column is RHS.
    tableau = [row[:] + [B[i]] for i, row in enumerate(A)]
    objective_row = [-c for c in C] + [0]
    tableau.append(objective_row)

    print_tableau(tableau, basic_vars, non_basic_vars) # Print initial tableau

    while True:
        # Identify the entering variable (column)
        objective_coeffs = tableau[m][:n + m]
        entering_col = None
        max_coeff = 0
        for j, val in enumerate(objective_coeffs):
            if val < 0 and val < max_coeff:
                max_coeff = val
                entering_col = j
        if entering_col is None: # Optimal solution reached
            break

        # Identify the leaving variable (row) using the minimum ratio test
        ratios = []
        for i in range(m):
            if tableau[i][entering_col] > 1e-12:
                ratio = tableau[i][-1] / tableau[i][entering_col]
                ratios.append((ratio, i))
        if not ratios:
            raise Exception("Linear program is unbounded.")
        leaving_row = min(ratios, key=lambda x: x[0])[1]

        pivot(tableau, leaving_row, entering_col)

        basic_vars[leaving_row] = entering_col # Update basic and non-basic variables
        print_tableau(tableau, basic_vars, non_basic_vars) # Print tableau after iteration
```

Figure 2: Simplex Implementation

Exercise in Matrix Form

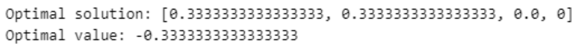
$$
\begin{aligned}
\text{Maximize} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{where} \quad & \mathbf{c} = \begin{bmatrix} -2 \\ 3 \\ 0 \\ 5 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \\
\text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\
& \mathbf{A} = \begin{bmatrix} -1 & 1 & -1 & -1 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 8 \\ 10 \\ 3 \end{bmatrix}, \\
& \mathbf{x} \geq \mathbf{0}
\end{aligned}
$$

Results and Conclusion



```
Optimal solution: [0, 2.5, 3.0, 0]
Optimal value: -22.5
```

Figure 3: Solution Standard



```
Optimal solution: [0.3333333333333333, 0.3333333333333333, 0.0, 0]
Optimal value: -0.3333333333333333
```

Figure 4: Solution My

Both in the standard exercise solution and in my version, the second and third constraints are active, which means they have the same base.