Aman Pandey Sanjeev Kumar

PUID: 0036571904

IE535 Linear Programming

Final Computing Project

1. Model:17 – Farm Fertilizer Problem

I. Question:

A farmer has to purchase the following quantities of fertilizer from four different shops, subject to the following capacities and prices. How can he fulfill his requirements at minimal cost?

Fertilizer Type	*Minimum Required (tons)*
1	185
2	50
3	50
4	200
5	185

Shop Number	*Maximum (all types combined) They Can Supply*
1	350 tons
2	225
3	195
4	275

	Price in Money Units per Ton of Fertilizer Type				
At Shop	**1**	**2**	**3**	**4**	**5**
1	45.0	13.9	29.9	31.9	9.9
2	42.5	17.8	31.0	35.0	12.3
3	47.5	19.9	24.0	32.5	12.4
4	41.3	12.5	31.2	29.8	11.0

II. Formulation:

The LPP cannot be formulated directly, since it has to satisfy a combination of all the shops and fertilizers. Let i denote the i^{th} fertilizer and j denote the j^{th} shop

Min Z =
45.0x_11+42.5x_12+47.5x_13+41.3x_14+13.9x_21+17.8x_22+19.9x_23+12.5x_24+29.9x_31+31.0x_32+24.0x_33+31.2x_34+31.9x_41+35.0x_42+32.5x_43+29.8x_44+9.9x_51+12.3x_52+12.4x_53+11.0x_54

Constraints:

The amount of each type of fertilizer purchased must meet the minimum requirement:

$x11 + x12 + x13 +x\ 14 \geq 185$ # constraints for fertilizer type 1
$x21 + x22 +x\ 23 + x24 \geq 50$ # constraints for fertilizer type 2
$x31 + x32 + x33 + x34 \geq 50$ # constraints for fertilizer type 3
$x41 + x42 + x43 + x44 \geq 200$ # constraints for fertilizer type 4
$x51 + x52 +x53 + x54 \geq 185$ # constraints for fertilizer type 5

The total amount of fertilizer that each shop can supply is limited:

$x11 + x21 + x31 + x41 + x51 \leq 350$ # constraints for shop 1
$x12 + x22 + x32 + x42 + x52 \leq 225$ # constraints for shop 2
$x13 + x23 + x33 + x43 + x53 \leq 195$ # constraints for shop 3
$x14 + x24 + x34 + x44 + x54 \leq 275$ # constraints for shop 4

The decision variables must be non-negative:
$x_{ij} \geq 0\ \forall\ i \in \{1,2,3,4,5\},\ \forall\ j \in \{1,2,3,4\}$

1.2.1 Standard Form:

Since we have >= in the formulation we will have to use artificial variables in the standard form. Also, the objective function is minimization and that is why the artificial variables in the objective function are going to be positive.

Objective Function:

Min Z =
45.0x_11+42.5x_12+47.5x_13+41.3x_14+13.9x_21+17.8x_22+19.9x_23+12.5x_24+
29.9x_31+31.0x_32+24.0x_33+31.2x_34+31.9x_41+35.0x_42+32.5x_43+29.8x_44
+9.9x_51+12.3x_52+12.4x_53+11.0x_54 + 0s1+ 0s2 + 0s3 + 0s4 + 0s5 + 0s6 + 0s7 +
0s8 + 0s9 + a1 + a2 + a3+ a4 + a5

Constraints for the Fertilizers:
x11 + x12 + x13 + x14 - s1+ a1 = 185 # constraints for fertilizer type 1
x21 + x22 +x 23 + x24 - s2 + a2 = 50 # constraints for fertilizer type 2
x31 + x32 + x33 + x34 - s3 + a3 = 50 # constraints for fertilizer type 3
x41 + x42 + x43 + x44 - s4 + a4 = 200 # constraints for fertilizer type 4
x51 + x52 + x53 + x54 - s5 + a5 = 185 # constraints for fertilizer type 5

Constraints for the Shops:
x11 + x21 + x31 + x41 + x51 + s6 = 350 # constraints for shop 1
x12 + x22 + x32 + x42 + x52 + s7 = 225 # constraints for shop 2
x13 + x23 + x33 + x43 + x53 + s8 = 195 # constraints for shop 3
x14 + x24 + x34 + x44 + x54 + s9 = 275 # constraints for shop 4

2. Code for the Two Phase Simplex Method:

I. The code takes in the input manually by the user.
II. The code checks for feasibility at the end of the Phase 1 iterations. If the objective value at the end of the Phase is found to be 0 then the code declares the LPP is feasible.
III. The code also checks for unbounded conditions. The function 'perform_ratio_test' computes the ratio values and identifies the pivot row. But if all the ratio values are negative then the code flags the LPP as unbounded
IV. The code also checks for redundancy. Using example 4.5 from BJS it is known that in the final iteration of the Phase 1 if there are found to be any artificial variables in the basis then the LPP has redundant constraints. The code for the same is implemented
V. The code for phase 1 and phase 2 is implemented using the different functions defined.
VI. The code computes the optimality for the " Farm Fertilizer Problem" at the end of Phase 2.
VII. The code also formulates the LPP into two separate commercial solvers GUROBI and CPLEX (Check section 2.2)

2.1. Code:

```
import numpy as np # For Matrix Operations and manipulation of numbers and indices
from prettytable import PrettyTable # To create a table for the simplex
from docplex.mp.model import Model # For using CPLEX as commercial solver
from gurobipy import * # For using GUROBI as commercial solver

#*****************PUID ending is 4, thus chose Model 17 LPP*****************
#*******************MODEL 17 - Farm Fertilizer Problem********************
# Inputs after converting the LP into standard form
# Below are the inputs after the  formulation of the LPP
#Coefficients of objective function
cost_coefficients_original = 
np.array([45,42.5,47.5,41.3,13.9,17.8,19.9,12.5,29.9,31.0,24.0,31.2,31.9,35.0,32.
5,29.8,9.9,12.3,12.4,11,0,0,0,0,0,0,0,0,0,1,1,1,1,1])

# Giving names to variables
variables = np.array(["X11", "X21", 
"X31","X41","X12","X22","X32","X42","X13","X23","X33","X43","X14","X24","X34","X4
4","X15","X25,","X35","X45","S1","S2","S3","S4","S5","S6","S7","S8","S9",
                 "A1","A2","A3","A4","A5"])

#Coefficients of the constraints
coefficients_matrix = np.array([
```

```
    [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,1,0,0,0,0], #
constraints for fertilizer type 1
    [0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,1,0,0,0], #
constraints for fertilizer type 2
    [0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,1,0,0], #
constraints for fertilizer type 3
    [0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,1,0], #
constraints for fertilizer type 4
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,-1,0,0,0,0,0,0,0,0,1], #
constraints for fertilizer type 5
    [1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],  #
constraints for shop 1
    [0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],  #
constraints for shop 2
    [0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0],  #
constraints for shop 3
    [0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0],  #
constraints for shop 4
])

# RHS/Sol of the constraints
constraints_rhs = np.transpose([np.array([185,50,50,200,185,350,225,195,275])])

#Index of the Slack and Artificial Variables that form the Indentity Matrix to
form the initial BFS for Phase-1
basic_vars_index = np.array([25,26,27,28,29,30,31,32,33])

# Cost Coefficients for the phase 1
cost_coefficients_phase_1 =
np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1])

#Input artificial variables indexes
artificial_vars_index = np.array([29,30,31,32,33])

phase_2 = True
```

Function 1:

```
def simplex_tableau(variables, basic_vars_index, cost_coefficients,
coefficients_matrix, constraints_rhs):
    # Extract basic variables and their corresponding cost coefficients using
NumPy indexing
    basic_vars = variables[basic_vars_index]
    cost_basic_vars = cost_coefficients[basic_vars_index]
```

```
    # Convert basic_vars_index, basic_vars, and cost_basic_vars to column vectors
    basic_vars_index = basic_vars_index[:, np.newaxis]
    basic_vars = basic_vars[:, np.newaxis]
    cost_basic_vars = cost_basic_vars[:, np.newaxis]

    # Stack coefficients_matrix and constraints_rhs to create the initial simplex
tableau
    simplex_tableau = np.column_stack((coefficients_matrix,
constraints_rhs)).astype(float)

    # Stack cost_basic_vars and basic_vars_index to create the auxiliary tableau
    simplex_tableau_aux = np.column_stack((cost_basic_vars,
basic_vars_index)).astype(float)

    # Return the initial simplex tableau, auxiliary tableau, and basic variables
    return simplex_tableau, simplex_tableau_aux, basic_vars
```

Function 2:

```
def compute_objective_function(simplex_tableau, simplex_tableau_aux,
cost_coefficients):
    """
    Compute the values of the objective function and check if the current
solution is optimal.

    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS
values.
    - simplex_tableau_aux: The auxiliary tableau containing cost coefficients and
basic variable indices.
    - cost_coefficients: The coefficients of the objective function.

    Returns:
    - objective_function_values: Values of the objective function (zj-cj) for
each variable.
    - is_feasible: Boolean indicating whether the current solution is optimal.
    """
    # Use NumPy vectorized operations to compute the objective function values
    objective_function_values = np.sum(simplex_tableau_aux[:, 0, np.newaxis] *
simplex_tableau[:, :-1], axis=0) - cost_coefficients[:simplex_tableau.shape[1] -
1]

    # Check if all values of the objective function are non-positive, indicating
optimality
    is_feasible = np.all(objective_function_values <= 0)
```

```python
    return objective_function_values.tolist(), is_feasible


def identify_pivot_column(simplex_tableau, objective_function_values):
    """
    Identify the pivot column for the next iteration.

    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS values.
    - objective_function_values: Values of the objective function (zj-cj) for each variable.

    Returns:
    - pivot_column: The identified pivot column in the simplex tableau.
    - pivot_column_index: The index of the identified pivot column.
    """
    # Find the index of the variable with the most positive objective function value
    pivot_column_index = np.argmax(objective_function_values)

    # Extract the identified pivot column from the simplex tableau
    pivot_column = simplex_tableau[:, pivot_column_index]

    return pivot_column, pivot_column_index
```

Function 3:

```python
def perform_ratio_test(simplex_tableau, pivot_column):
    """
    Perform the ratio test using Bland's rule to identify the pivot row and column for the next iteration of the simplex algorithm.

    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS values.
    - pivot_column: The identified pivot column in the simplex tableau.

    Returns:
    - ratio_values: The computed ratio values for each row.
    - pivot_row_index: The index of the identified pivot row.
    - pivot_column_index: The index of the identified pivot column.
    - simplex_tableau_display: The tableau with additional ratio values for display purposes.
    - is_unbounded: Boolean indicating whether the problem is unbounded.
```

```python
    """
    is_unbounded = False

    # Compute ratio values for each row
    ratio_values = simplex_tableau[:, -1] / pivot_column

    # Identify the indices of rows with positive ratios
    positive_ratio_indices = np.where(ratio_values > 0)[0]

    if len(positive_ratio_indices) > 0:
        # Choose the pivot row using Bland's rule (minimum index among positive
ratios)
        pivot_row_index = np.min(positive_ratio_indices)
    else:
        # All ratio values are non-positive, indicating unboundedness
        is_unbounded = True
        pivot_row_index = None

    # Identify the indices of columns with positive ratios
    positive_ratio_columns = np.where(ratio_values > 0)[0]

    if len(positive_ratio_columns) > 0:
        # Choose the pivot column using Bland's rule (minimum index among
positive ratios)
        pivot_column_index = np.min(positive_ratio_columns)
    else:
        # All ratio values are non-positive, indicating unboundedness
        is_unbounded = True
        pivot_column_index = None

    # Combine the ratio values with the simplex tableau for display purposes
    simplex_tableau_display = np.hstack((simplex_tableau, ratio_values[:,
np.newaxis]))

    return ratio_values, pivot_row_index, pivot_column_index,
simplex_tableau_display, is_unbounded
```

Function 4:

```python
def update_simplex_tableau(simplex_tableau, simplex_tableau_aux, basic_vars,
variables, pivot_column_index, pivot_row_index, cost_coefficients):
    """
    Perform row operations to update the simplex tableau, auxiliary tableau, and
basic variables after finding the pivot.
```

```
    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS
values.
    - simplex_tableau_aux: The auxiliary tableau containing cost coefficients and
basic variable indices.
    - basic_vars: The basic variables for the current iteration.
    - variables: The variable names.
    - pivot_column_index: The index of the identified pivot column.
    - pivot_row_index: The index of the identified pivot row.
    - cost_coefficients: The coefficients of the objective function.

    Returns:
    - updated_tableau: Updated simplex tableau after row operations.
    - updated_tableau_aux: Updated auxiliary tableau after row operations.
    - basic_vars: Updated basic variables after row operations.
    """
    pivot_row = simplex_tableau[pivot_row_index, :]
    pivot_value = pivot_row[pivot_column_index]

    updated_tableau = simplex_tableau - np.outer(simplex_tableau[:,
pivot_column_index], pivot_row) / pivot_value
    updated_tableau[pivot_row_index, :] = pivot_row / pivot_value

    updated_tableau_aux = simplex_tableau_aux.copy()
    updated_tableau_aux[pivot_row_index, 1] = pivot_column_index
    updated_tableau_aux[pivot_row_index, 0] =
cost_coefficients[pivot_column_index]

    basic_vars[pivot_row_index] = variables[pivot_column_index]

    return updated_tableau, updated_tableau_aux, basic_vars
```

Function 5:

```
def compute_objective_value(simplex_tableau, simplex_tableau_aux,
cost_coefficients):
    """
    Compute the value of the objective function for the current solution.

    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS
values.
    - simplex_tableau_aux: The auxiliary tableau containing cost coefficients and
basic variable indices.
    - cost_coefficients: The coefficients of the objective function.
```

```
    Returns:
    - objective_value: The computed value of the objective function for the
current solution.
    """
    rhs_values = simplex_tableau[:, -2]
    variable_indices = simplex_tableau_aux[:, 1].astype(int)

    objective_value = np.sum(rhs_values * cost_coefficients[variable_indices])

    return objective_value
```

Function 6:

```
ef display_simplex_tableau(simplex_tableau, simplex_tableau_aux, basic_vars,
iteration_count, variables, ratios=False,
                             z=False, objective_function_values=()):
    """
    Display the simplex tableau for each iteration of the simplex algorithm.

    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS
values.
    - simplex_tableau_aux: The auxiliary tableau containing cost coefficients and
basic variable indices.
    - basic_vars: The basic variables for the current iteration.
    - iteration_count: The current iteration number.
    - variables: The variable names.
    - ratios: Flag indicating whether to display the Ratio Test column.
    - objective_function_values: Values of the objective function (zj-cj) for
each variable.

    Returns:
    None (prints the tableau to the console)
    """
    print("Simplex Tableau - Iteration " + str(iteration_count))

    # Combine auxiliary tableau, basic variables, and main tableau
    simplex_tableau_table = np.hstack((simplex_tableau_aux, basic_vars,
simplex_tableau))

    # Determine the structure of the first row based on whether ratios and z are
displayed
    header_row = (
```

```
        ["cost_basic_vars", "basic_vars_index", "basic_vars"] +
variables.tolist() +
        ["RHS"] if not ratios else ["cost_basic_vars", "basic_vars_index",
"basic_vars"] +
        variables.tolist() + ["RHS"] + ["Ratio Test"]
    )

    # Set objective function values based on whether z is displayed
    objective_function_values = (
        np.hstack((("--", "--", "red_cost"), objective_function_values, ("0")))
        if not ratios else np.hstack((("--", "--", "red_cost"),
objective_function_values, ("0", "0")))
    )

    # Check if the objective function row is included
    simplex_tableau_table = np.vstack((header_row, objective_function_values,
simplex_tableau_table[1:])) if z else np.vstack((header_row,
simplex_tableau_table))

    # Display the tableau using PrettyTable
    pretty_table = PrettyTable()
    pretty_table.field_names = simplex_tableau_table[0]
    for row in simplex_tableau_table[1:]:
        pretty_table.add_row(row)

    print(pretty_table)
```

Function 7:

```
def perform_Iterations(simplex_tableau, simplex_tableau_aux, basic_vars,
variables, cost_coefficients):
    """
    Iterate operations until an optimal solution is found or the problem is
determined to be unbounded.

    Parameters:
    - simplex_tableau: The main simplex tableau containing coefficients and RHS
values.
    - simplex_tableau_aux: The auxiliary tableau containing cost coefficients and
basic variable indices.
    - basic_vars: The basic variables for the current iteration.
    - variables: The variable names.
    - cost_coefficients: The coefficients of the objective function.

    Returns:
```

```
    - Result tuple containing:
      - simplex_tableau_with_ratio: The final simplex tableau with additional
ratio values for display.
      - simplex_tableau_aux: The final auxiliary tableau.
      - basic_vars: The final basic variables.
      - iteration_count: The total number of iterations performed.
      - optimal_value: The optimal value of the objective function for the found
solution.
    """
    iteration_count = 0
    is_unbounded = False
    optimal_value = 0

    while True:
        if iteration_count > 0:
            # Perform row operations to update the tableau after finding the
pivot
            simplex_tableau, simplex_tableau_aux, basic_vars =
update_simplex_tableau(
                simplex_tableau, simplex_tableau_aux, basic_vars, variables,
pivot_column_index, pivot_row_index, cost_coefficients
            )

        # Compute the values of the objective function and check for optimality
        objective_function_values, is_feasible =
compute_objective_function(simplex_tableau, simplex_tableau_aux,
cost_coefficients)

        # Identify the pivot column for the next iteration
        pivot_column, pivot_column_index = identify_pivot_column(simplex_tableau,
objective_function_values)

        # Perform the ratio test to identify the pivot row
        ratio_values, pivot_row_index, simplex_tableau_with_ratio, is_unbounded =
perform_ratio_test(simplex_tableau, pivot_column)

        # Check for optimality
        if is_feasible:
            break

        # Display the tableau with detailed information
        iteration = iteration_count
        display_simplex_tableau(simplex_tableau_with_ratio, simplex_tableau_aux,
basic_vars, iteration_count, variables,
```

```
                                ratios=True, z=True,
objective_function_values=objective_function_values)

        iteration_count += 1

    if is_feasible:
        optimal_value = compute_objective_value(simplex_tableau_with_ratio,
simplex_tableau_aux, cost_coefficients)
        print("Objective Value of the LP  ", optimal_value)

        # Use PrettyTable to display the basic variables with RHS values
        solutions_table = PrettyTable()
        solutions_table.field_names = ["basic_vars", "RHS"]
        for row in np.hstack((basic_vars,
np.atleast_2d(simplex_tableau_with_ratio[:, -2]).T)):
            solutions_table.add_row(row)

        print("Solutions:")
        print(solutions_table)

        # Check redundancy of constraints
        artificial_vars_in_basis = [var for var in basic_vars if
var.startswith("A")]
        if any(artificial_vars_in_basis):
            print("Constraints are not redundant.")
        else:
            print("Constraints are redundant.")

    # If the problem is unbounded, print a message indicating unboundedness
    if is_unbounded:
        print("The provided LPP is Unbounded")


# Check if Phase 2 of the two-phase simplex method is required
if phase_2 == True:
    # Use the cost coefficients from Phase 1 for initialization
    cost_coefficients = cost_coefficients_phase_1
else:
    # Use the original cost coefficients for Phase 2
    cost_coefficients = cost_coefficients_original

# Initialize the simplex tableau, auxiliary tableau, and basic variables
simplex_tableau, simplex_tableau_aux, basic_vars = simplex_tableau(variables,
basic_vars_index, cost_coefficients, coefficients_matrix, constraints_rhs)
```

```
# Unpack the results for Phase 1
simplex_tableau_phase_1, simplex_tableau_aux_phase_1, basic_vars_phase_1,
iteration_count, optimal_value_phase_1 = perform_Iterations(
    simplex_tableau, simplex_tableau_aux, basic_vars, variables,
cost_coefficients)

if phase_2:
    # Check if Phase 1 found a feasible solution
    if optimal_value_phase_1 == 0:
        # Use original objective function coefficients for Phase 2
        cost_coefficients = cost_coefficients_original

        # Remove the ratio column and artificial variables columns from the Phase
1 tableau
        artificial_vars_index_set = set(artificial_vars_index)
        simplex_tableau_phase_2 = simplex_tableau_phase_1[:, [
            col_idx for col_idx in range(simplex_tableau_phase_1.shape[1] - 1)
            if col_idx not in artificial_vars_index_set
        ]]

        # Copy auxiliary tableau from Phase 1
        simplex_tableau_aux_phase_2 = simplex_tableau_aux_phase_1

        # Remove artificial variables from the list of variables
        variables_phase_2 = variables[[
            var_idx for var_idx in range(variables.shape[0])
            if var_idx not in artificial_vars_index_set
        ]]

        # Copy basic variables from Phase 1
        basic_vars_phase_2 = basic_vars_phase_1

        # Update coefficients of the original objective value for Phase 2
        for index in range(basic_vars_phase_2.shape[0]):
            variable_index = int(simplex_tableau_aux_phase_2[index, 1])  #
Indexes of the basic variables
            simplex_tableau_aux_phase_2[index, 0] =
cost_coefficients[variable_index]  # Get original coefficients of basic variables

        # Perform the simplex method for Phase 2
        simplex_result_phase_2 = perform_Iterations(
            simplex_tableau_phase_2, simplex_tableau_aux_phase_2,
basic_vars_phase_2, variables_phase_2, cost_coefficients
        )
```

```
        # Unpack the results for Phase 2
        simplex_tableau_phase_2, simplex_tableau_aux_phase_2, basic_vars_phase_2,
iteration_count_phase_2, optimal_value_phase_2 = simplex_result_phase_2

    else:
        print("The provided LPP is infeasible")
```

2.2. Code for the Solver – GUROBI and CPLEX

2.2.2. Code for GUROBI

```
# Using commerical solver GUROBI for the verification of the solution

from gurobipy import *

# Create a new model
m = Model("17. Fertilizer_problem")

# Define sets
FERTILIZERS = range(1, 6)
SHOPS = range(1, 5)

# Define parameters
minReq = {1: 185, 2: 50, 3: 50, 4: 200, 5: 185}
maxSupply = {1: 350, 2: 225, 3: 195, 4: 275}
cost = {
    (1, 1): 45.0, (1, 2): 42.5, (1, 3): 47.5, (1, 4): 41.3,
    (2, 1): 13.9, (2, 2): 17.8, (2, 3): 19.9, (2, 4): 12.5,
    (3, 1): 29.9, (3, 2): 31.0, (3, 3): 24.0, (3, 4): 31.2,
    (4, 1): 31.9, (4, 2): 35.0, (4, 3): 32.5, (4, 4): 29.8,
    (5, 1): 9.9,  (5, 2): 12.3, (5, 3): 12.4, (5, 4): 11.0
}

# Define decision variables
x = m.addVars(FERTILIZERS, SHOPS, name="x")

# Define objective function
m.setObjective(sum(cost[i,j]*x[i,j] for i in FERTILIZERS for j in SHOPS),
GRB.MINIMIZE)

# Define constraints
for i in FERTILIZERS:
    m.addConstr(sum(x[i,j] for j in SHOPS) >= minReq[i])
for j in SHOPS:
```

```
    m.addConstr(sum(x[i,j] for i in FERTILIZERS) <= maxSupply[j])

# Optimize model
m.optimize()

# Print the optimal solution
for v in m.getVars():
    print('%s: %g' % (v.varName, v.x))
print('Objective: %g' % m.objVal)
```

2.2.2. Code for CPLEX

```
from docplex.mp.model import Model

# Create a model
mdl = Model("Warehousing")

# Define decision variables
x = {(i,j): mdl.continuous_var(name="x_{0}_{1}".format(i,j)) for i in range(1,6)
for j in range(1,6)}

# Define objective function
mdl.minimize(650*x[(1,1)] + 1000*(x[(2,1)]+x[(1,2)]) +
1350*(x[(3,1)]+x[(2,2)]+x[(1,3)]) + 1600*(x[(4,1)]+x[(3,2)]+x[(2,3)]+x[(1,4)]) +
1900*(x[(5,1)]+x[(4,2)]+x[(3,3)]+x[(2,4)]+x[(1,5)]))

# Define constraints
mdl.add_constraint(x[(1,1)] + x[(1,2)] + x[(1,3)] + x[(1,4)] + x[(1,5)] >= 30)
mdl.add_constraint(x[(1,1)] + x[(2,1)] + x[(1,2)] + x[(2,2)] + x[(1,3)] +
x[(2,3)] + x[(1,4)] + x[(2,4)] + x[(1,5)] + x[(2,5)] >= 20)
mdl.add_constraint(x[(2,1)] + x[(3,1)] + x[(2,2)] + x[(3,2)] + x[(1,3)] +
x[(2,3)] + x[(3,3)] + x[(1,4)] + x[(2,4)] + x[(3,4)] + x[(1,5)] + x[(2,5)] +
x[(3,5)] >= 40)
mdl.add_constraint(x[(3,1)] + x[(4,1)] + x[(3,2)] + x[(4,2)] + x[(2,3)] +
x[(3,3)] + x[(4,3)] + x[(1,4)] + x[(2,4)] + x[(3,4)] + x[(4,4)] + x[(1,5)] +
x[(2,5)] + x[(3,5)] + x[(4,5)] >= 10)
mdl.add_constraint(x[(4,1)] + x[(5,1)] + x[(4,2)] + x[(3,3)] + x[(2,4)] +
x[(1,5)] + x[(5,2)] + x[(4,3)] + x[(3,4)] + x[(2,5)] + x[(5,3)] + x[(4,4)] +
x[(3,5)] + x[(5,4)] + x[(4,5)] + x[(5,5)] >= 50)

# Solve the model
solution = mdl.solve()

# Print the solution
```

```python
if solution:
    print(solution)
else:
    print("No solution found")
```

3. Results:

These are the results for Model 17. The code was written using Jupyter Notebook in Vs Code. I was unable to extract the tables for the iterations as images or convert the output into PDF from the output cells in Jupyter Notebook. Copying the tables also did not work. The Tables have 36 columns and 10 rows and thus they wouldn't fit in a single page. Showing the screenshots of the results is a requirement in the Project description and the only option that was left was for me to take partial screen clippings of the results for both Phase 1 and Phase 2.

3.1. Results for Phase 1:

Only the last half of every table is attached here due to space constraints. Apologizes for any inconvenience this may cause.

Iteration 1:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	2.000	2.000	2.000	2.000	0.000	0.000	0.000	0.000	-1.000	1.000	1.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	200.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	185.000
0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	350.000	350.000
0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	225.000	inf
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 2:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000	-2.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	200.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	165.000	165.000
0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	225.000	inf
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 3:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	200.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	165.000	inf
0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	225.000	225.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 4:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	-2.000	-1.000	-1.000	-1.000	-2.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	50.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	200.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	165.000	165.000
0.000	0.000	0.000	-1.000	-1.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	-1.000	0.000	0.000	0.000	-1.000	205.000	205.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 5:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	-2.000	-2.000	-1.000	-1.000	-2.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	50.000
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	200.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	1.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	-1.000	115.000	115.000
0.000	0.000	0.000	-1.000	-1.000	1.000	1.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	-1.000	-1.000	0.000	0.000	-1.000	155.000	155.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 6:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	-2.000	-2.000	-2.000	-1.000	-2.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	200.000	200.000
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	0.000	-1.000	65.000	65.000
0.000	0.000	0.000	-1.000	-1.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	-1.000	105.000	105.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 7:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	-2.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	50.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	1.000	1.000	1.000	0.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	135.000	135.000
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	0.000	-1.000	65.000	-65.000
0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	40.000	40.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 8:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000
0.000	0.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	10.000	10.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	95.000	95.000
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	0.000	-1.000	-1.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	-1.000	105.000	-105.000
0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	40.000	-40.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	195.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 9:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	-2.000	-2.000	-1.000	-1.000	-1.000	0.000	0.000
0.000	0.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	10.000	-10.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	50.000
1.000	0.000	1.000	1.000	1.000	0.000	0.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	85.000	85.000
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	1.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	-1.000	115.000	-115.000
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	1.000	1.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	-1.000	-1.000	0.000	0.000	0.000	185.000	185.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 10:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	-2.000	-2.000	-2.000	-1.000	-1.000	0.000	0.000
0.000	0.000	-1.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	-1.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	60.000	-60.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	35.000	35.000
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	135.000	135.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 11:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	-2.000	-2.000	-2.000	-2.000	-2.000	0.000	0.000
1.000	0.000	0.000	1.000	1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	95.000	-95.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	35.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
-1.000	0.000	0.000	0.000	-1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	-1.000	-1.000	-1.000	-1.000	-1.000	100.000	100.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	inf

Iteration 12:

X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	A1	A2	A3	A4	A5	RHS	Ratio Test
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	195.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	50.000	inf
1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	35.000	inf
0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	185.000	inf
0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	50.000	inf
-1.000	0.000	0.000	0.000	-1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	-1.000	-1.000	-1.000	-1.000	-1.000	100.000	-100.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	275.000	275.000

Basic Variables in last Iteration of Phase 1 :

```
basic_vars
------------
X2
X3
X2
X2
X1
X1
X2
S1
X4
```

Optimal Solution has reached indicating feasibility:

```
Objective Value of the LP 0.0
Solutions:
basic_vars          RHS
------------    -------
X21              90.000
X31             195.000
X23              50.000
X24              35.000
X15             185.000
X14             165.000
X22              50.000
S1              375.000
X41             275.000
```


```
Objective Value of the LP 0.0
Solutions:
basic_vars          RHS
------------    -------
X21              90.000
X31             195.000
X23              50.000
X24              35.000
X15             185.000
X14             165.000
X22              50.000
S1              375.000
X41             275.000
```


3.2. Results for Phase 2:

Iteration 1:

X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
-1.400	0.000	0.000	7.500	4.000	0.000	0.700	5.600	0.800	0.000	24.700	11.500	7.500	29.500	39.400	42.500	47.500	41.300	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	195.000	195.000
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	35.000	inf
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
0.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	375.000	375.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	275.000	inf

Iteration 2:

X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
-1.400	0.000	0.000	-40.000	4.000	0.000	0.700	-41.900	0.800	0.000	24.700	11.500	7.500	29.500	39.400	42.500	0.000	41.300	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	35.000	inf
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
0.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	-1.000	0.000	0.000	0.000	-1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	1.000	180.000	180.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	275.000	inf

Iteration 3:

X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
41.100	0.000	0.000	2.500	46.500	0.000	0.700	0.600	43.300	0.000	-17.800	-31.000	-35.000	-13.000	-3.100	0.000	0.000	41.300	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	195.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	35.000	inf
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
0.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	50.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	90.000	90.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	275.000	275.000

Iteration 4:

X33	X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
7.000	41.100	0.000	0.000	2.500	46.500	0.000	0.700	0.600	43.300	0.000	28.800	-31.000	-35.000	-13.000	-3.100	0.000	0.000	41.300	0.000	0.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	195.000	inf
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	35.000	35.000
0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
0.000	0.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	165.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	40.000	40.000
0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	225.000	225.000

Iteration 5:

X33	X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
7.000	41.100	0.000	-46.500	-44.000	0.000	0.000	-45.800	-45.900	-3.200	0.000	28.800	-31.000	11.500	33.500	43.400	0.000	0.000	41.300	0.000	0.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	195.000	inf
1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	50.000
0.000	0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	35.000	-35.000
0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
0.000	0.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	165.000	165.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
0.000	1.000	0.000	-1.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	1.000	5.000	5.000
0.000	1.000	0.000	-1.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	0.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	1.000	190.000	190.000

Iteration 6:

X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
-3.400	0.000	-2.000	0.500	0.000	0.000	-1.300	-1.400	-3.200	-44.500	-15.700	-31.000	-33.000	-11.000	-1.100	0.000	0.000	-3.200	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	195.000	195.000
0.000	0.000	1.000	1.000	0.000	0.000	1.000	1.000	0.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	-1.000	45.000	45.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	40.000	inf
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
-1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	-1.000	0.000	0.000	0.000	0.000	-1.000	160.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
1.000	0.000	-1.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	1.000	5.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	185.000	inf

Iteration 7:

X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
-3.400	0.000	-9.000	-6.500	0.000	0.000	-8.300	-8.400	-3.200	-37.500	-8.700	-24.000	-26.000	-4.000	5.900	0.000	0.000	3.800	0.000	0.000
0.000	0.000	-1.000	0.000	0.000	0.000	-1.000	0.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	1.000	1.000	150.000	150.000
0.000	0.000	1.000	1.000	0.000	0.000	1.000	1.000	0.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	0.000	0.000	-1.000	45.000	-45.000
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	40.000	inf
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
-1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	-1.000	0.000	0.000	0.000	0.000	-1.000	160.000	inf
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
1.000	0.000	-1.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	1.000	5.000	5.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	185.000	inf

Iteration 8:

X43	X14	X24	X34	X44	X15	X25,	X35	X45	S1	S2	S3	S4	S5	S6	S7	S8	S9	RHS	Ratio Test
-9.300	0.000	-3.100	-0.600	0.000	0.000	-2.400	-2.500	-3.200	-43.400	-14.600	-24.000	-31.900	-9.900	0.000	0.000	0.000	-2.100	0.000	0.000
-1.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	1.000	0.000	145.000	inf
1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	40.000	-40.000
0.000	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	185.000	inf
-1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000	-1.000	-1.000	-1.000	0.000	-1.000	0.000	0.000	0.000	0.000	-1.000	160.000	160.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	50.000	inf
1.000	0.000	-1.000	-1.000	0.000	0.000	-1.000	-1.000	0.000	1.000	1.000	0.000	1.000	1.000	1.000	0.000	0.000	1.000	5.000	-5.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	185.000	185.000

Optimal Solution:

```
Objective Value of the LP
17449.0
Solutions:
basic_vars         RHS
------------    ------
S7               65.00
S8              145.00
X3               50.00
X4              200.00
X1              185.00
X2              160.00
X4               50.00
S6              165.00
X4               25.00
```


```
Objective Value of the LP
17449.0
Solutions:
basic_vars         RHS
------------    ------
S7               65.00
S8              145.00
X3               50.00
X4              200.00
X1              185.00
X2              160.00
X4               50.00
S6              165.00
X4               25.00
```


C:\Users\amanp\AppData\Local

3.3 Commercial Solver Results:

The Primary commercial solver that is used to solve the LPP is GUROBI using python. The method used by the solver is unclear but the results indicates that the solver either uses Dual Simplex Method or the Dual Simplex Path following Interior Point Method.

3.3.1 Results using GUROBI:

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (win64 - Windows 11.0 (22621.2))

CPU model: 13th Gen Intel(R) Core(TM) i9-13900H, instruction set [SSE2|AVX|AVX2]
Thread count: 14 physical cores, 20 logical processors, using up to 20 threads

Optimize a model with 9 rows, 20 columns and 40 nonzeros
Model fingerprint: 0x27987ed0
```

```
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+01, 5e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e+01, 4e+02]
Presolve time: 0.00s
Presolved: 9 rows, 20 columns, 40 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   6.700000e+02   0.000000e+00      0s
       6    1.7449000e+04   0.000000e+00   0.000000e+00      0s

Solved in 6 iterations and 0.01 seconds (0.00 work units)
Optimal objective 1.744900000e+04
x[1,1]: 0
x[1,2]: 160
x[1,3]: 0
x[1,4]: 25
x[2,1]: 0
x[2,2]: 0
x[2,3]: 0
x[2,4]: 50
x[3,1]: 0
x[3,2]: 0
x[3,3]: 50
x[3,4]: 0
x[4,1]: 0
x[4,2]: 0
x[4,3]: 0
x[4,4]: 200
x[5,1]: 185
x[5,2]: 0
x[5,3]: 0
x[5,4]: 0
Objective: 17449
```

Screenshot of the Result obtained using GUROBI:


```
Solved in 6 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.744900000e+04
x[1,1]: 0
x[1,2]: 160
x[1,3]: 0
x[1,4]: 25
x[2,1]: 0
x[2,2]: 0
x[2,3]: 0
x[2,4]: 50
x[3,1]: 0
x[3,2]: 0
x[3,3]: 50
x[3,4]: 0
x[4,1]: 0
x[4,2]: 0
x[4,3]: 0
x[4,4]: 200
x[5,1]: 185
x[5,2]: 0
x[5,3]: 0
x[5,4]: 0
Objective: 17449
```

3.3.2 Results using CPLEX:

A secondary solver CPLEX has also been used for further validation of the results.

```
solution for: Model
objective: 17449.000
status: OPTIMAL_SOLUTION(2)
x_1_2 = 160.000
x_1_4 = 25.000
x_2_4 = 50.000
x_3_3 = 50.000
x_4_4 = 200.000
x_5_1 = 185.000
```

Screenshot of the results obtained using CPLEX:


```
solution for: Model
objective: 17449.000
status: OPTIMAL_SOLUTION(2)
x_1_2 = 160.000
x_1_4 = 25.000
x_2_4 = 50.000
x_3_3 = 50.000
x_4_4 = 200.000
x_5_1 = 185.000
```


3.4. Interpretation of the Results:

The objective value of $17449.00 indicates that this is the minimum cost the farmer will have to spend to buy the different fertilizers of different quantities from different shops following all the constraints.

The values of the basic variables of the solution indicate the following:

- $x12$=160.000 means that 160 tons of fertilizer type 1 is purchased from shop 2.
- $x14$=25.000 means that 25 tons of fertilizer type 1 is purchased from shop 4.
- $x24$=50.000 means that 50 tons of fertilizer type 2 is purchased from shop 4.
- $x33$=50.000 means that 50 tons of fertilizer type 3 is purchased from shop 3.
- $x44$=200.000 means that 200 tons of fertilizer type 4 is purchased from shop 4.
- $x51$=185.000 means that 185 tons of fertilizer type 5 is purchased from shop 1.

These results indicate that, to minimize the cost while meeting the constraints, the farmer should purchase the specified amounts of each fertilizer type from different shops as outlined by the decision variables. The overall cost for this optimal solution is 17449.000.

4. Conclusion:

The Two-Phase Simplex Algorithm is implemented to solve the Farm Fertilizer Problem. The first phase of the simplex helped us verify if the problem was feasible and was not unbounded. It also helps identify redundancy in the constraints by checking the presence of artificial variables in the constraints at the end of Phase 1. If the constraints in the basis that do not have an artificial variable at the end of Phase 1 are removed.

Phase 2 of the algorithm finds the optimal objective value and helps the farmer make the right choice. Also, the results of the two-phase simplex solver is validated using the results from the commercial solver GUROBI and CPLEX.