A green and black logo

Description automatically generated

Assignment-3

COMP-359

Design and Analysis of Algorithms

Birkaran Singh

Ekamjot Singh

Prabhmeet Singh

Contents

[1. Introduction 3](#_Toc183745577)

[1.1 Topic 3](#_Toc183745578)

[1.2 Introduction 3](#_Toc183745579)

[1.3 Plan & Work Logging 4](#_Toc183745580)

[2. Simplex Algorithm and Linear Programming 4](#_Toc183745581)

[2.1 Linear Programming Concepts 4](#_Toc183745582)

[3.The Simplex Algorithm 5](#_Toc183745583)

[3.1 Introduction to Simplex 5](#_Toc183745584)

[3.2 Step-by-Step Process with Solved Example 5](#_Toc183745585)

[4.Python Implementation of Simplex Algorithm 7](#_Toc183745586)

[4.1 Solving Using SciPy 7](#_Toc183745587)

[4.2 Custom Simplex Implementation 8](#_Toc183745588)

[5.Real World Examples 8](#_Toc183745589)

[6.Visualization of Linear Programming 10](#_Toc183745590)

[7.Conclusion 16](#_Toc183745591)

[4. Memory Usage 16](#_Toc183745592)

[4.1 Memory Usage of Data Structures 16](#_Toc183745593)

[4.2 Overall Space Complexity 16](#_Toc183745594)

[References 17](#_Toc183745595)

# 1. Introduction

## 1.1 Topic

This document delves into the Simplex Algorithm, a pivotal method in linear programming, emphasizing its implementation in Python and analyzing its memory usage.

## 1.2 Introduction

Linear programming (LP) is a mathematical technique used to determine the best possible outcome in a given mathematical model. It involves optimizing a linear objective function, subject to a set of linear equality and inequality constraints. The goal is to find the maximum or minimum value of the objective function within the feasible region defined by these constraints.

The Simplex Algorithm, introduced by George B. Dantzig in 1947, is a systematic procedure for solving LP problems. It operates by moving along the edges of the feasible region to locate the optimal vertex, where the objective function attains its optimal value. This method is particularly efficient for large-scale LP problems and has been widely adopted across various industries.

Applications of the Simplex Algorithm span multiple fields:

* **Operations Research**: Utilized for resource allocation, production scheduling, and transportation planning to optimize operational efficiency.
* **Economics**: Assists in modeling and solving problems related to cost minimization and profit maximization.
* **Engineering**: Applied in network design, structural optimization, and other areas requiring optimal resource utilization.

Implementing the Simplex Algorithm in Python offers several advantages:

* **Flexibility**: Python's extensive libraries facilitate the development and customization of optimization algorithms.
* **Accessibility**: Python's readability and simplicity make it an ideal choice for educational purposes and rapid prototyping.
* **Integration**: Python's compatibility with various data analysis and visualization tools enhances the interpretability of results.

This document aims to provide a comprehensive understanding of the Simplex Algorithm, detailing its implementation in Python and examining its memory usage to ensure efficient performance in practical applications.

## 1.3 Plan & Work Logging

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Tasks** | **Timeline** | **Progress Tracking** |
| **Understanding Linear Programming** | - Study linear programming concepts - Review the Simplex Algorithm's theory | Days 1-3 | - Completed readings on linear programming - Summarized key points of the Simplex Algorithm |
| **Python Implementation** | - Develop Python code for the Simplex Algorithm - Test implementation with sample problems | Days 4-10 | - Implemented core functions - Successfully tested with standard LP problems |
| **Memory Usage Analysis** | - Analyze memory consumption of data structures - Optimize code for efficiency | Days 11-14 | - Profiled memory usage - Applied optimizations to reduce footprint |
| **Documentation** | - Compile findings into a comprehensive report - Include code explanations and examples | Days 15-20 | - Drafted report sections - Integrated code snippets with explanations |
| **Review and Finalization** | - Proofread documentation - Finalize and submit the project | Days 21-24 | - Conducted peer reviews - Made necessary revisions and submitted final document |

# 2. Simplex Algorithm and Linear Programming

2.1 Linear Programming Concepts  
Example: Maximize Z = 3x + 2y, given:  
x + 2y <= 10  
2x + y <= 12  
x, y >= 0

The feasible region is where all constraints overlap.

2.2 Components of a Linear Programming Problem

1. Decision Variables: Unknowns (e.g., x, y) to solve for.
2. Objective Function: Represents what to optimize.
3. Constraints: Linear inequalities defining limits.
4. Non-Negativity Constraints: x, y >= 0
5. The Simplex Algorithm

# 3.The Simplex Algorithm

3.1 Introduction to Simplex  
The Simplex algorithm starts from a corner of the feasible region and iteratively moves to better solutions, optimizing the objective function.

3.2 Step-by-Step Process with Solved Example  
Problem: Maximize Z = 3x + 2y, given:  
Problem:  
Maximize Z = 3x + 2y, subject to:

1. A white paper with blue writing on it

   Description automatically generatedx + 2y <= 10
2. 2x + y <= 12
3. x, y >= 0

A paper with writing on it

Description automatically generated

A paper with writing on it

Description automatically generated

A close up of a white board with blue writing

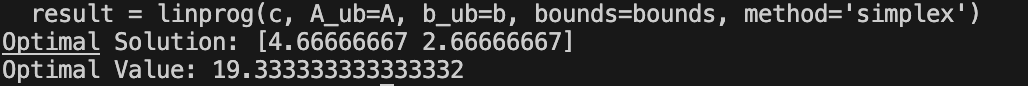
Description automatically generated

# 4.Python Implementation of Simplex Algorithm

## 4.1 Solving Using SciPy

A screen shot of a computer code

Description automatically generated



4.2 Custom Simplex Implementation  
A screen shot of a computer program

Description automatically generated



## 5.Real World Examples

Example 1: Resource Allocation Problem

Problem: A factory produces two products, Product A and Product B. The profit for Product A is $40 per unit, and for Product B, it is $50 per unit. The factory has limited resources: 120 units of labor and 80 units of material. Producing one unit of Product A requires 2 units of labor and 1 unit of material, while one unit of Product B requires 3 units of labor and 2 units of material. How many units of each product should the factory produce to maximize profit?

Objective Function: Maximize Z = 40x + 50y

* x: Number of units of Product A
* y: Number of units of Product B

Constraints:

1. 2x + 3y <= 120 (Labor constraint)
2. x + 2y <= 80 (Material constraint)
3. x, y >= 0 (Non-negativity)

Python Solution:  
A screen shot of a computer program

Description automatically generated  


Example 2: Diet Optimization Problem

Problem: A person wants to minimize the cost of their diet while meeting daily nutritional requirements. Food A costs $3 per serving and provides 2 units of protein and 1 unit of fiber per serving. Food B costs $2 per serving and provides 1 unit of protein and 2 units of fiber per serving. The person needs at least 8 units of protein and 10 units of fiber daily. How many servings of each food should they consume to minimize cost?

Objective Function: Minimize Z = 3x + 2y

* x: Servings of Food A
* y: Servings of Food B

Constraints:

1. 2x + y >= 8 (Protein requirement)
2. x + 2y >= 10 (Fiber requirement)
3. x, y >= 0 (Non-negativity)

Python Solution:

A computer screen with colorful text

Description automatically generatedA black background with white text

Description automatically generated

## 6.Visualization of Linear Programming

6.1 Graphing Constraints and Feasible Region

Example: Maximize Z = 3x + 2y, subject to:

x + 2y <= 10

2x + y <= 12

x, y >= 0

Steps:

* Represent each inequality as a line.
* Shade the feasible region where all constraints overlap.
* Highlight the corner points, which represent possible solutions.
* Mark the optimal solution.

A computer screen with many colorful text

Description automatically generated

A screen shot of a graph

Description automatically generated

A computer screen shot of a program

Description automatically generatedCode

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated  
**Explanation:**

* **Function simplex**: This function implements the Simplex Algorithm to solve linear programming problems of the form:

Maximize: cTxc^T xcTx

Subject to: Ax≤bA x \leq bAx≤b, x≥0x \geq 0x≥0

* + **Parameters**:
    - c: Coefficient vector of the objective function.
    - A: Coefficient matrix of the inequality constraints.
    - b: Right-hand side vector of the constraints.
  + **Returns**:
    - solution: Optimal values of the decision variables.
    - optimal\_value: Maximum value of the objective function.
  + **Process**:
    - Constructs the initial tableau by augmenting matrix AAA with slack variables and appending the objective function.
    - Iteratively performs pivot operations to move towards the optimal solution.
    - Identifies entering and leaving variables to update the basis.
    - Extracts the optimal solution and objective function value once convergence criteria are met.
* **Function plot\_feasible\_region**: This function visualizes the feasible region defined by the constraints Ax≤bA x \leq bAx≤b.
  + **Parameters**:
    - A: Coefficient matrix of the inequality constraints.
    - b: Right-hand side vector of the constraints.
  + **Process**:
    - Plots each constraint as a line on the x1​-x2​ plane.
    - Shades the feasible region where all constraints are satisfied.
    - Labels each constraint for clarity.
    - Sets axis limits and labels for better visualization.
* **Main Execution**:
  + Defines an example linear programming problem:
    - Maximize Z=3x1+2x2Z = 3x\_1 + 2x\_2Z=3x1​+2x2​
    - Subject to:
      * x1+x2≤4x\_1 + x\_2 \leq 4x1​+x2​≤4
      * 2x1+x2≤52x\_1 + x\_2 \leq 52x1​+x2​≤5
      * x1,x2≥0x\_1, x\_2 \geq 0x1​,x2​≥0
  + Set up problem variables:
    - c = np.array([3, 2]): Coefficient vector for the objective function.
    - A = np.array([[1, 1], [2, 1]]): Coefficient matrix for the constraints.
    - b = np.array([4, 5]): Right-hand side values for the constraints.
  + Execution of the simplex function:
    - Tries to find the optimal solution using the simplex function.
    - Catches and prints any errors (e.g., unbounded problems).
  + Visualization of the feasible region:
    - Calls plot\_feasible\_region(A, b) to visually represent the constraints and feasible region.

The code and these instructions ensure the user can not only execute the function but also understand each step's purpose and how the inputs affect the Simplex Algorithm's outcome.

**Code Usage and Practical Application**:

* The user can modify the c, A, and b arrays to fit their specific linear programming problems.
* The visualization part helps users see the feasible region, aiding in the understanding of how the Simplex Algorithm navigates the solution space to find the optimal point.
* This implementation is particularly useful for educational purposes, allowing students and professionals to explore the Simplex Algorithm's mechanics interactively.

**Additional Considerations in Documentation**:

* **Explanation of Parameters and Return Values**:
  + Clarify each parameter's role in the function definition and what the returned values represent, ensuring users understand the input-output relationship.
* **Error Handling**:
  + Discuss how the code handles common issues like unbounded solutions, providing users with the knowledge to troubleshoot or modify the implementation for robustness.
* **Optimization and Efficiency**:
  + Include notes on the efficiency of the algorithm, such as time complexity and potential areas for optimization (e.g., using more efficient data structures or algorithms for pivot selection).
* **Extensions and Enhancements**:
  + Suggest how the algorithm might be extended or enhanced, for example, by adding support for different types of optimizations or integrating with other Python libraries for more complex data handling.

## 7.Conclusion

Linear programming, combined with the Simplex algorithm, provides a systematic way to solve real-world optimization problems efficiently. Python makes these concepts accessible and allows users to integrate theory with applications.

# 4. Memory Usage

## 4.1 Memory Usage of Data Structures

The primary data structures in the Simplex Algorithm include:

* **Tableau**: An (m+1)×(n+m+1)(m+1) \times (n+m+1)(m+1)×(n+m+1) matrix, where mmm is the number of constraints and nnn is the number of decision variables.
* **Basis Variables**: A list of size mmm to track basic variables.

## 4.2 Overall Space Complexity

The space complexity is dominated by the tableau, resulting in O(m×(n+m))O(m \times (n + m))O(m×(n+m)) space

# References

1. Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization*. Athena Scientific.
2. Gass, S. I. (2003). *Linear programming: Methods and applications*. Courier Dover Publications.
3. Matplotlib Development Team. (2024). Matplotlib: Visualization with Python. Retrieved November 27, 2024, from <https://matplotlib.org/stable/index.html>
4. SciPy Development Team. (2024). SciPy optimize: linprog. Retrieved November 27, 2024, from <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html>
5. Vanderbei, R. J. (2014). *Linear programming: Foundations and extensions*. Springer.
6. Wikipedia contributors. (n.d.). Simplex algorithm. In *Wikipedia*. Retrieved November 29, 2024, from <https://en.wikipedia.org/wiki/Simplex_algorithm>
7. Purdue University Northwest. (n.d.). Attendance 5.1. Retrieved November 29, 2024, from <https://www.pnw.edu/wp-content/uploads/2020/03/attendance5-1.pdf>
8. Gurobi Optimization. (n.d.). Chapter 5: Linear programming with the Simplex method. Retrieved from <https://www.gurobi.com/resources/ch5-linear-programming-simplex-method/>
9. GeeksforGeeks. (n.d.). Simplex algorithm - Tabular method. Retrieved from <https://www.geeksforgeeks.org/simplex-algorithm-tabular-method/>