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](#_Toc183806925)

[1.1 Topic 3](#_Toc183806926)

[1.2 Introduction 3](#_Toc183806927)

[1.3 Plan & Work Logging 4](#_Toc183806928)

[2. Simplex Algorithm and Linear Programming 4](#_Toc183806929)

[2.1 Linear Programming Concepts 4](#_Toc183806930)

[3.The Simplex Algorithm 5](#_Toc183806931)

[3.1 Introduction to Simplex 5](#_Toc183806932)

[3.2 Step-by-Step Process with Solved Example 5](#_Toc183806933)

[4.Python Implementation of Simplex Algorithm 7](#_Toc183806934)

[4.1 Solving Using SciPy 7](#_Toc183806935)

[4.2 Custom Simplex Implementation 8](#_Toc183806936)

[5.Real World Examples 8](#_Toc183806937)

[6.Visualization of Linear Programming 10](#_Toc183806938)

[Explanation of Python Functions and Main Execution 14](#_Toc183806939)

[Main Execution Flow: 15](#_Toc183806940)

[Practical Application and User Guidance: 15](#_Toc183806941)

[Additional Documentation Considerations: 16](#_Toc183806942)

[Conclusion 16](#_Toc183806943)

[Memory Usage and Efficiency 16](#_Toc183806944)

[References 17](#_Toc183806945)

# 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

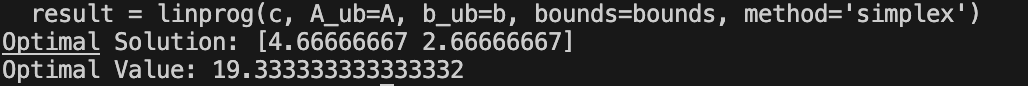
Description automatically generated

# 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 of Python Functions and Main Execution

**Function:** simplex

* **Purpose**: Implements the Simplex Algorithm to solve linear programming problems.
* **How It Works**:
  + **Objective**: Maximize an objective function represented by a coefficient vector c.
  + **Constraints**: Subject to matrix A multiplied by variable vector x being less than or equal to vector b, with x being non-negative.
* **Details**:
  + **Inputs**:
    - c: Coefficients of the objective function.
    - A: Coefficients of the inequality constraints.
    - b: Right-hand side values of the constraints.
  + **Outputs**:
    - solution: Optimal values of the decision variables.
    - optimal\_value: The highest value of the objective function achieved.
  + **Process**:
    - Constructs an initial tableau by integrating matrix A with slack variables and the objective function.
    - Iteratively performs pivot operations to navigate towards the optimal solution.
    - Updates entering and leaving variables to refine the basis.
    - Determines the optimal solution and its corresponding value once the solution meets convergence criteria.

**Function:** plot\_feasible\_region

* **Purpose**: Visualizes the feasible region defined by the constraints.
* **Details**:
  + **Inputs**:
    - A: Matrix of the inequality constraints.
    - b: Right-hand side values.
  + **Process**:
    - Draws each constraint as a line on the x1-x2 plane.
    - Shades the area where all constraints overlap, marking the feasible region.
    - Adds labels for clarity and sets appropriate axis limits for better visualization.

## Main Execution Flow:

* **Setup**:
  + Defines a sample linear programming problem to maximize an objective function 3x1 + 2x2 subject to given constraints.
  + Establishes problem variables using numpy arrays for coefficients and constraints.
* **Execution**:
  + Attempts to find the optimal solution using the simplex function.
  + Handles potential errors like unbounded solutions by catching exceptions and providing feedback.
* **Visualization**:
  + Depicts the feasible region using the plot\_feasible\_region function, aiding in understanding the constraints and solution space.

## Practical Application and User Guidance:

* Users can adjust the c, A, and b arrays to adapt the code for different linear programming challenges.
* Visualization assists in comprehending how the Simplex Algorithm explores the solution space to locate the optimal point.
* The code serves educational purposes well, enabling both students and professionals to interactively learn about the mechanics of the Simplex Algorithm.

## Additional Documentation Considerations:

* **Parameter Explanation**:
  + Clarifies the role of each parameter within the function and the significance of the returned values.
* **Error Handling**:
  + Discusses common computational issues and how the code addresses them, such as unbounded solutions.
* **Optimization Tips**:
  + Provides insights on improving the efficiency of the algorithm, potentially through more advanced data structures or optimization techniques.
* **Suggestions for Enhancements**:
  + Offers ideas for extending the algorithm’s capabilities or integrating it with additional Python libraries for more complex scenarios.

# Conclusion

The integration of linear programming with the Simplex Algorithm through Python offers a systematic approach to solving real-world optimization problems efficiently. This documentation not only guides the user through using the provided Python code but also enhances understanding of underlying principles and potential applications.

# Memory Usage and Efficiency

* **Data Structures**:
  + **Tableau**: A matrix used to track the state of the Simplex algorithm, where m is the number of constraints and n is the number of decision variables.
  + **Basis Variables**: A list used to track which variables are currently the basis of the solution.
* **Space Complexity**:
  + Dominated by the tableau, the space complexity is calculated based on the size of the tableau and the need to store intermediate results during computation.

# 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. 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>
7. Gurobi Optimization. (n.d.). Chapter 5: Linear programming with the Simplex method. Retrieved from <https://www.gurobi.com/resources/ch5-linear-programming-simplex-method/>
8. GeeksforGeeks. (n.d.). Simplex algorithm - Tabular method. Retrieved from <https://www.geeksforgeeks.org/simplex-algorithm-tabular-method/>