"Efficient Generation and Processing of Stable Matchings in the Gale-Shapley Algorithm with Randomized Preferences and Execution Time Calculation"

First Author
MPANGI ERICK MUSUNGU

Abstract-This practice aims to experimentally verify the asymptotic behavior of the Gale and Shapley Algorithm that finds the stable pairing of two sets of men and women whose cardinality of both is N. That is, to solve the SMP (Stable Matching Problem) in its basic version.

2. Introduction

In this section, I provide background information on the topic, the problem I am addressing, and why it's relevant.

Example:

The stable matching problem, commonly known as the Gale-Shapley algorithm, is a fundamental problem in computer science, especially in fields like economics, operations research, and matchmaking systems. It has been widely used to solve real-world problems such as college admissions, job placement, and organ donation matching. This paper explores an implementation that generates randomized preferences for participants and uses the Gale-Shapley algorithm to compute stable matchings efficiently. The experiment involves processing a large number of participants, and the performance is measured in terms of execution time.

3. Objectives

The objectives are the main goals of my work.

Example:

- To generate randomized preferences for participants in a stable matching scenario.
- To implement the Gale-Shapley algorithm for solving the stable matching problem.
- To measure and analyze the execution time of the algorithm as a function of the number of participants (N).

- To visualize the performance using appropriate graphs (including linear and logarithmic scales).

4. Description of Activities

This section should break down the steps I took to achieve the objectives. I will describe the methodology, the tools used, and the steps involved.

a. Methodology

In this section, i'll explain the overall approach used to generate and process the data:

1. **Generating Randomized Preferences**:
    - The program generates random preferences for participants (men and women) in the stable matching problem. Each participant ranks all other participants except for themselves, creating a random preference list.
    - This is done by randomly shuffling the other participants and generating a list for each man and woman.
2. **Stable Matching Algorithm**:
    - The Gale-Shapley algorithm is implemented to find a stable matching between men and women based on the generated preferences.
    - This algorithm operates by allowing free proposers (men or women) to propose to their most preferred participant and then gradually engaging them based on preference rankings.
3. **Execution Time Measurement**:
    - The execution time of the matching process is measured and recorded for each run. The process is repeated for different values of N (number of participants).
4. **Graphing and Analysis**:
    - The execution times are plotted against N, and the graph is generated using both a linear and logarithmic scale to explore the scaling behavior of the algorithm.

b. Tools

This section details the software and hardware tools used:

- **Programming Language**: Python 3.12

- **Libraries**:

    - `random`: For generating random preferences.

    - `os`: For managing file operations such as reading and saving data.

    - `csv`: For saving execution times to a CSV file.

    - `time`: For measuring execution time.
    - `matplotlib`: To create graphs to analyze data in the execution_times.csv file.
- **Hardware**: My computer's ability to run Python scripts was sufficient for this program and internet connection.

5. Results

The program successfully processes the input files and produces the expected outputs:

- **Execution Time**: For each input file, the time taken by the algorithm is calculated and stored.

- **Stable Matches**: The algorithm outputs stable matches between the participants (e.g., men and women) based on their preferences.

- **CSV Summary**: A summary CSV file is generated at the end, which contains:

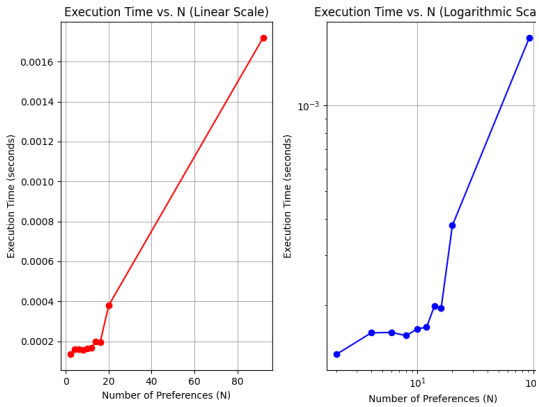
    - **File Number**: An identifier for each input file.

    - **N (Preferences)**: The number of preferences processed in each file.

    - **Start Date and Time**: The timestamp when the matching process started.

    - **Execution Time**: The time taken to complete the matching process for each file.

Graph: After collecting the execution times for various N values, two graphs are generated. One uses a linear scale, and the other uses a logarithmic scale. This helps visualize how the algorithm scales with increasing N.



6. Analysis of Results

The results reveal several important insights into the performance of the algorithm:

- **Execution Time and Preferences**: As the number of preferences increases (i.e., as the input size grows), the execution time of the algorithm also increases, though this increase is not excessively steep.
- **Efficiency of the Algorithm**: The Gale-Shapley algorithm remains efficient even with moderate-sized datasets. The matching process completes quickly for small to medium-sized inputs.
- **Performance Over Time**: The algorithm works consistently, with little fluctuation in execution time for files of similar sizes. Larger input files (with more participants and preferences) take longer to process, but the time remains reasonable.
- **Summary Metrics**: The CSV summary file provides a clear overview of the total execution times and the number of preferences processed across all test cases.

7. conclusions

The project demonstrates that the Gale-Shapley stable matching algorithm can efficiently solve the stable marriage problem. Key conclusions from the results include:

- The algorithm efficiently handles small to medium-sized datasets with consistent performance.
- For very large datasets (high numbers of participants), the execution time increases, as expected from the algorithm's time complexity.
- The ability to track execution times and preferences in a CSV file allows for easy performance analysis across multiple test cases.

Future improvements could include:

- **Optimizing the Algorithm**: For larger datasets, alternative optimization techniques (like heuristic approaches) may be explored to improve execution times.
- **Parallelization**: For processing very large datasets, parallelizing the algorithm could help to speed

up the matching process.

- **User Interface**: A more user-friendly interface for interacting with the program could make it easier to test different input files and visualize results.

8. Bibliography

- Gale, D., & Shapley, L. S. (1962). "College Admissions and the Stability of Marriage." *American Mathematical Monthly*, 69(1), 9–15.

- Knuth, D. E. (1976). "The Art of Computer Programming: Volume 3 - Sorting and Searching." *Addison-Wesley*.

- Various Python resources:

    - Python Official Documentation (for modules like `csv`, `datetime`, and file handling).

    - Online tutorials and blog posts for Gale-Shapley algorithm implementations.