CS648: Randomized Algorithms

Project Report

Gale Shapley Algorithm

Under the Guidance of: Prof. Surender Baswana
Department of Computer Science and Engineering
IIT Kanpur

Submitted By: Lakshika **210554**
Vanshika Gupta **211146**

MARCH 2024

Contents

1 Abstract

This project presents a probabilistic analysis of the Gale-Shapley algorithm for stable matching, exploring its time complexity in scenarios where the preference lists of individuals are chosen randomly and independently. The Gale-Shapley algorithm is a popular method for solving the stable marriage problem. While its worst-case time complexity is $O(n^2)$, this analysis demonstrates that under certain randomized conditions, the algorithm exhibits a reduced time complexity of $O(n \log n)$. The report provides a detailed investigation, implementation, and analysis of the algorithm's behaviour, offering insights into its efficiency under probabilistic scenarios.

2 Introduction

In the real world, there are so many cases when we face pairing problems: students choosing their future universities, men and women seeking each other to create families, internet services that need to be connected with users with the smallest amount of time. In all these cases, we need to create stable pairs that need to satisfy certain criteria. We can imagine a stable pair for (A, a) as a pair where neither A nor a have any better options.

The stable marriage problem involves matching participants from two equally sized groups, ensuring that no two individuals prefer each other over their current partners. The Gale-Shapley algorithm is well-known in cooperative game theory in tackling stable matching problems such as stable marriages, stable roommates, college to students, residency applicants to hospitals, human organs to patients, etc. It provides a solution to this problem by iteratively proposing and accepting/rejecting potential matches, resulting in a stable pairing. This project focuses on analyzing the algorithm's performance when individuals' preference lists are randomly generated, aiming to understand how randomness affects its time complexity.

3 Problem Statement

The specific objectives of this project are:

- To implement the Gale-Shapley algorithm for stable matching.
- To analyze the algorithm's time complexity under randomized preference lists.
- To compare the average time complexity with the worst-case time complexity.

4 Algorithm

Stable Matching Problem

Consider n women (w_1 , w_2 . . , w_n) and n men (m_1 , m_2 . . , m_n) .

- A matching is a 1-1 correspondence between the men and the women.
- Each person has a strict preference list of the members of the other sex.
- A matching is unstable iff there exist w_i and m_j such that: -
  - w_i and m_j are not matched together
  - w_i prefers m_j to her match
  - m_j prefers w_i to his match
- A matching that is not unstable is stable.

4.1 Gale Shapley Algorithm

The gale shapley algorithm is a deterministic algorithm, which solves the stable matching problem in a very neat and elegant way. The basic idea behind this algorithm is "man proposes, woman disposes": each currently unattached man proposes to the most desirable woman on his list who has not already rejected him, and this woman then decides whether to accept or reject a proposal.

Algorithm 1 Gale-Shapley Algorithm

```
1: Assign each person to be free;
2: while some man m is free do
3:     w ← first woman on m's list to whom m has not yet proposed;
4:     if w is free then
5:         Assign m and w to be engaged to each other;
6:     else
7:         if w prefers m to her fiance m_0 then
8:             Assign m and w to be engaged and m_0 to be free;
9:         else
10:            w rejects m;
11:        end if
12:    end if
13: end while
14: Output the matching.
```

Time Complexity

1. The time complexity of the deterministic algorithm is $O(n^2)$
2. The probabilistic analysis of this deterministic algorithm has a surprising time complexity of $O(n \log n)$

4.2 Average Case Analysis of the Gale Shapley Algorithm

In probabilistic analysis, we consider random input and investigate what happens when it's processed by a fixed algorithm.
In the Gale Shapley algorithm described above, randomness is introduced by choosing the preference list of each man and woman randomly uniformly and independent of others.

1. **Expected Number of Iterations for Convergence:** By analyzing the average behavior of the algorithm across various random preferences, we can estimate the expected number of iterations required for the algorithm to converge to a stable matching.
2. **Correctness of the Algorithm:** The algorithm is guaranteed to converge to a stable matching in a finite number of steps, regardless of the initial preferences. This convergence property ensures that the algorithm terminates with a stable solution within a bounded number of iterations.
3. **Randomness in inputs:** Each time a man has to make a proposal, he picks a random woman from the set of women not already propositioned by him, and proceeds to propose to her. Clearly, this is equivalent to choosing the random preference lists prior to the execution of the algorithm.
4. **Data Simulations:** Data Simulations are employed by choosing random uniform inputs to empirically validate the probabilistic guarantees derived from theoretical analysis.
5. **Probabilistic Error Bounds:** Probabilistic analysis can yield error bounds that quantify the deviation between the expected performance of the algorithm and its actual behavior under random preferences. These bounds provide probabilistic guarantees on the algorithm's accuracy and robustness in producing stable matchings.

5 Implementation

The implementation of the algorithm has been done using various tools including C++, Excel, and R

5.1 Using C++

The algorithmic solution to this problem was written in C++, and then different types of data were generated by running the algorithm for different inputs. This highlights the role of randomness in the data generation process. The algorithm was executed multiple times with different sets of random inputs, resulting in the production of distinct datasets.

5.2 Using Excel

The data generated was stored in Excel for further use and meticulous analysis.

5.3 Using R

R provides a vast array of tools for data manipulation, statistical analysis, visualization, and machine learning. It is a valuable tool to analyze data and derive meaningful conclusions.
R has been vastly utilized in our project for doing all the comparisons between theoretical and analytical analysis.

1. The data stored in Excel was imported into R
2. Different types of tables were made by manipulation of the stored data
3. We created different types of plots such as barplots, scatterplots, and boxplots. These plots offer valuable insights into the differences between our observed data and theoretical expectations. They also reveal how our data behaves when subjected to various input variations.

6 Analysis

6.1 Theoretical Analysis

The time complexity of the Gale Shapley Algorithm is proportional to the number of proposals made by men. Henceforth, in this whole project, the time complexity of the algorithm and the number of proposals have been used interchangeably.

Modification of Algorithm for theoretical analysis

A simplified modification of the Gale-Shapley algorithm: -

- At each iteration, a man proposes to a woman selected uniformly at random from all n women, regardless of previous rejections.
- The revised algorithm generates more proposals due to the inclusion of some "wasted" rejections. Consequently, the expected running time serves as an upper bound on the original algorithm's runtime.
- This modification however does not alter the algorithm's outcome, as a man rejected by a woman will face rejection again upon proposing to her in subsequent iterations.

Expected Number of Proposals

Theorem 6.1. *If the preference lists are chosen randomly uniformly, the expected number of proposals in the Gale-Shapley algorithm is* $O(n \log n)$.

Proof: The algorithm terminates once all women have received at least one proposal.
The randomized stable matching process is analogous to the coupon collector problem, so the total number of proposals such that every woman is proposed to atleast once is the same as the total number of coupons till each coupon has appeared atleast once.
Coupon Collector Problem : Expected duration of coupon collector experiment is $O(n \log n)$
The expected number of proposals in Gale-Shapley is at most nH_n or $O(n \log n)$

6.2 Analytical Analysis

The average number of proposals in the Gale-Shapley algorithm is obtained analytically by randomly generating 100 samples for each N and calculating their mean. These empirical values are then compared with the theoretical expected number of proposals, which is given by nH_n, where n represents the number of elements in the proposing set and H_n denotes the n-th harmonic number.

N	Analytical Avg Proposals	Numerical Avg Proposals(nH_n)
1000	8392	7485.471
2000	18832	16356.74
3000	28559	25751.25
4000	38416	35485.56
5000	51100	45472.54
6000	62653	55660.88
7000	77271	66016.67
8000	82836	76515.8
9000	94673	87140.26
10000	110621	97876.06

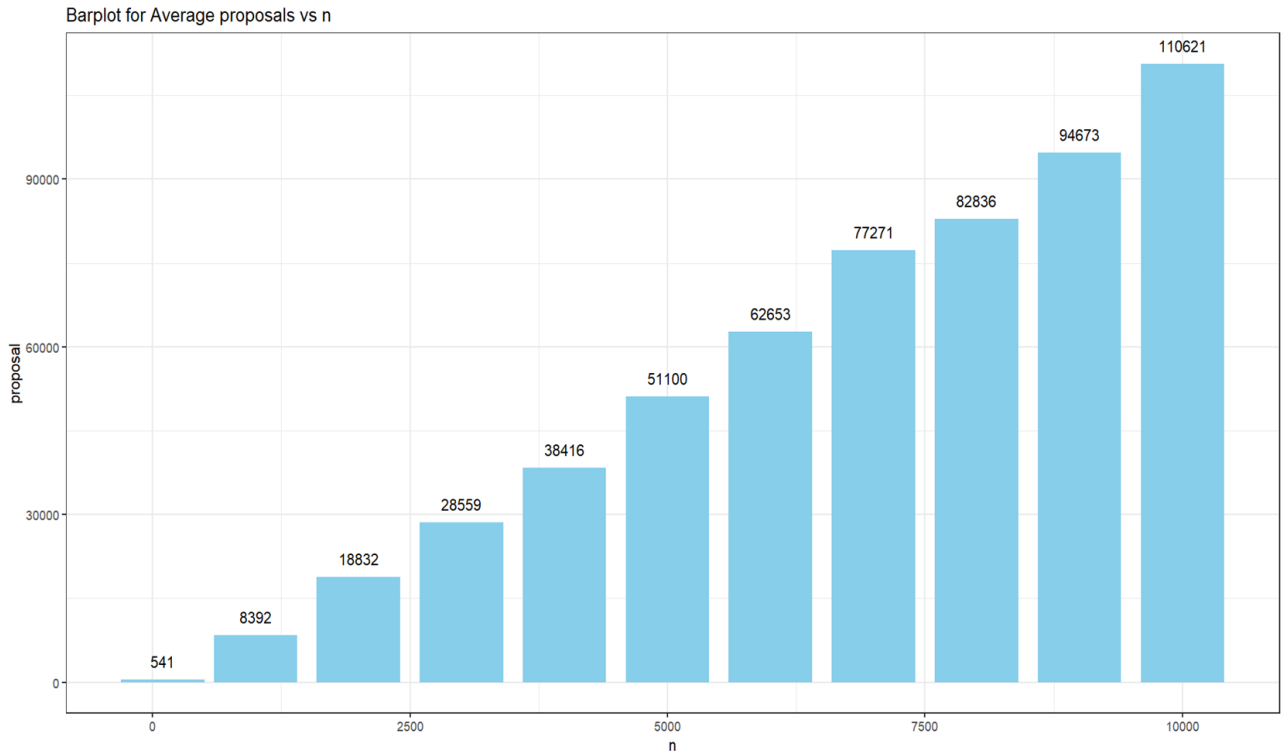


Figure 6.1: Increasing Proposals with Increasing n

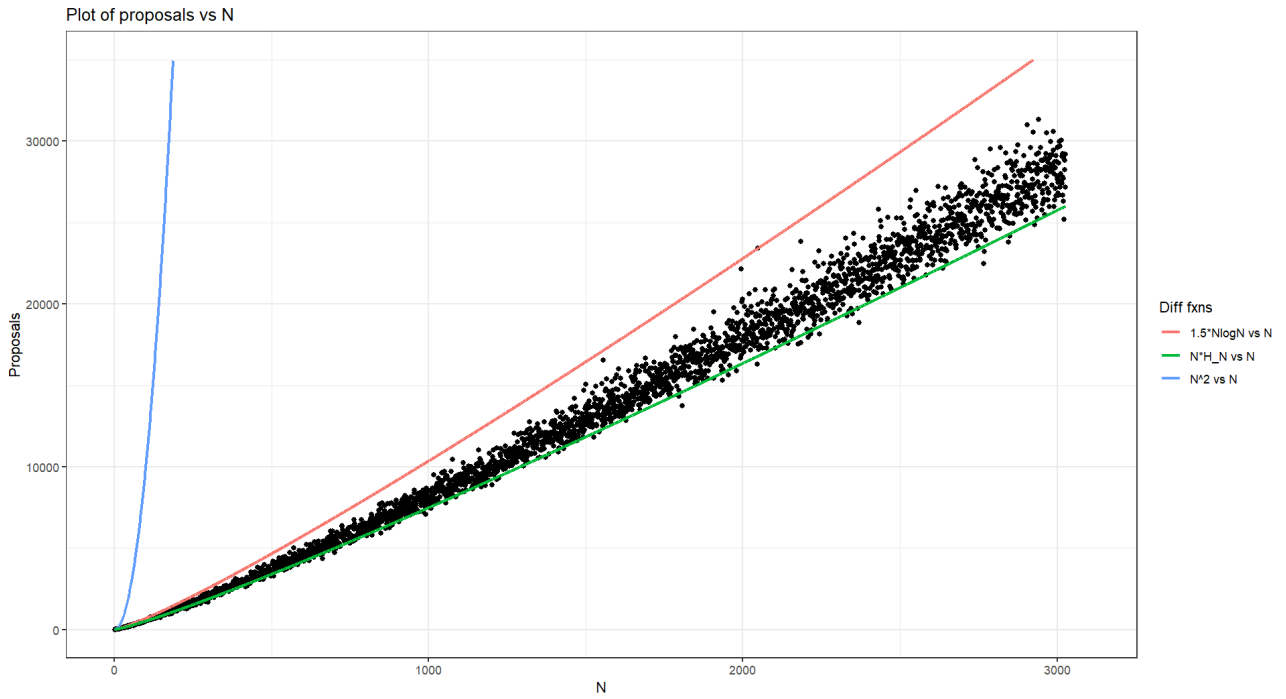


Figure 6.2: This scatterplot tells us that almost all the proposals are far lesser than N^2 and very close to NH_N

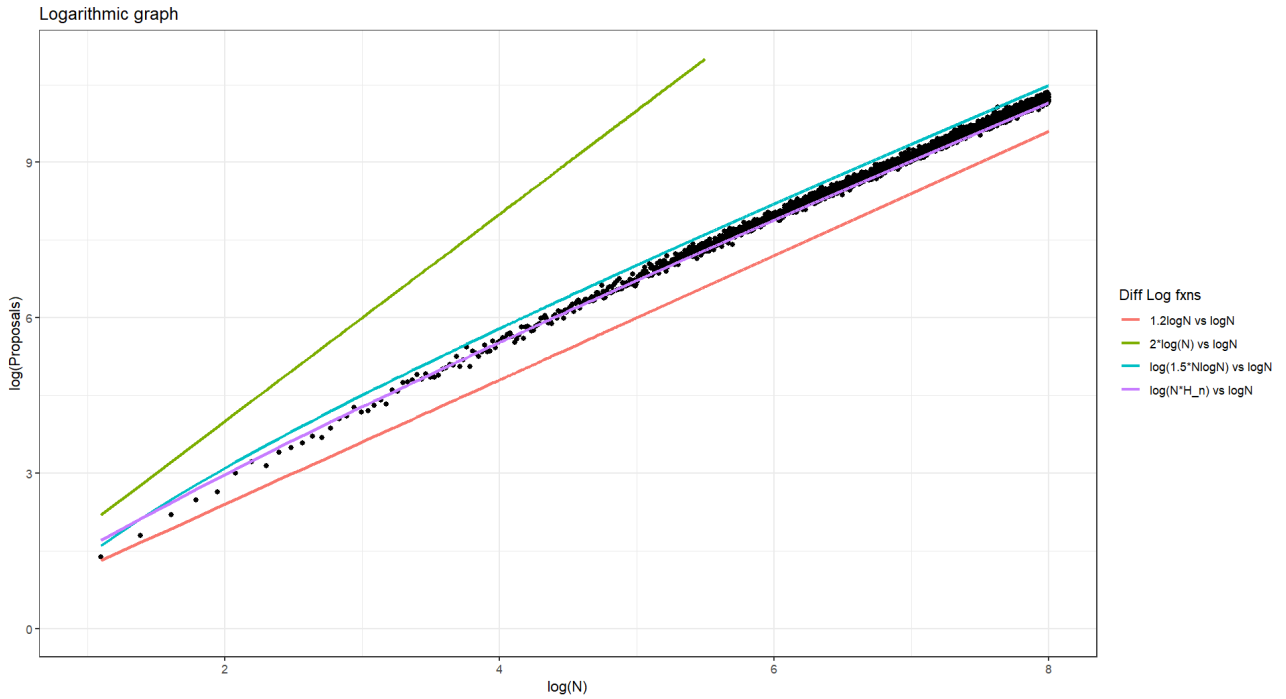
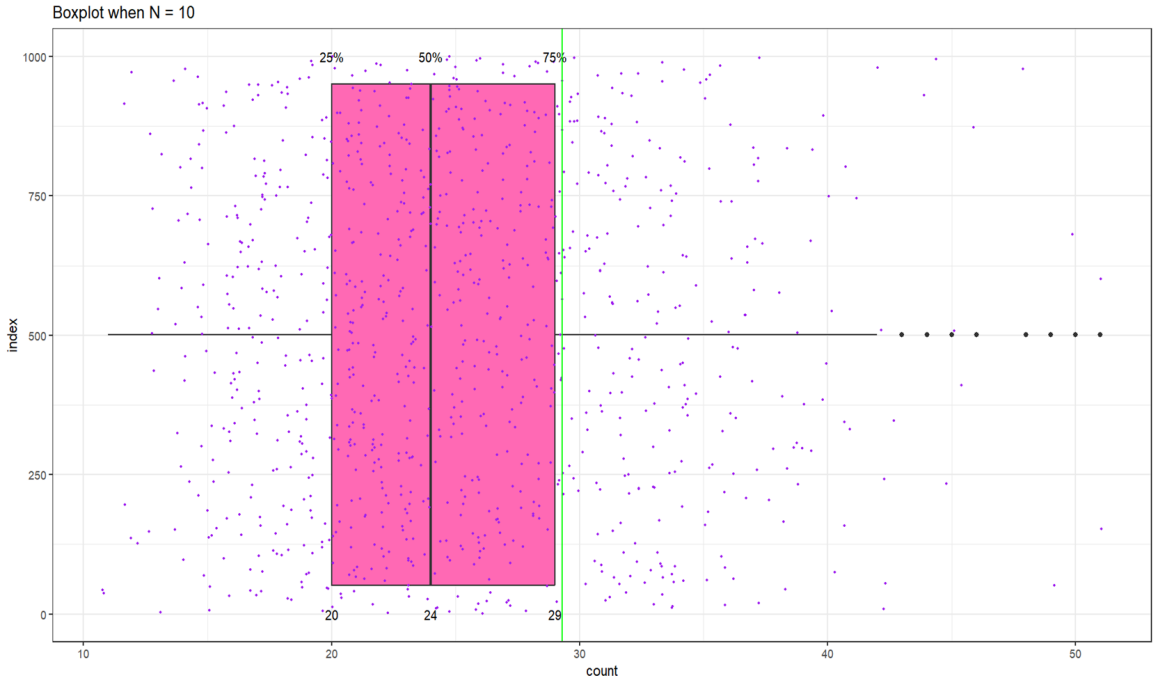


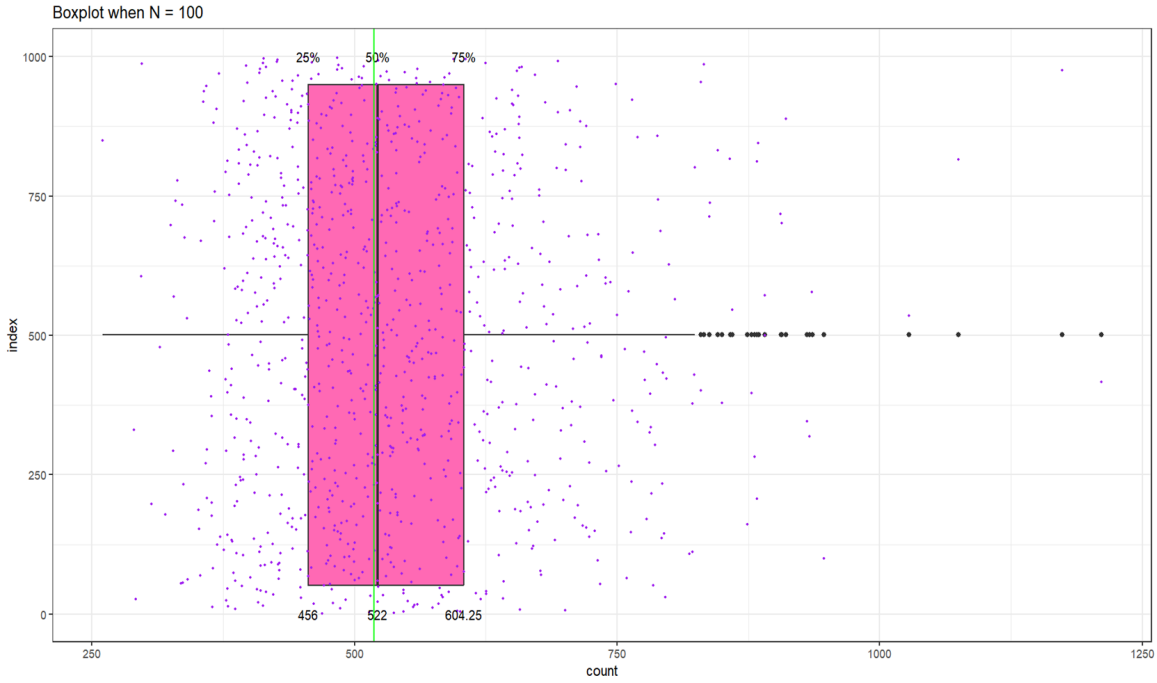
Figure 6.3: The log plot ensures us that the expected number of iterations is NH_N and is $O(n \log n)$

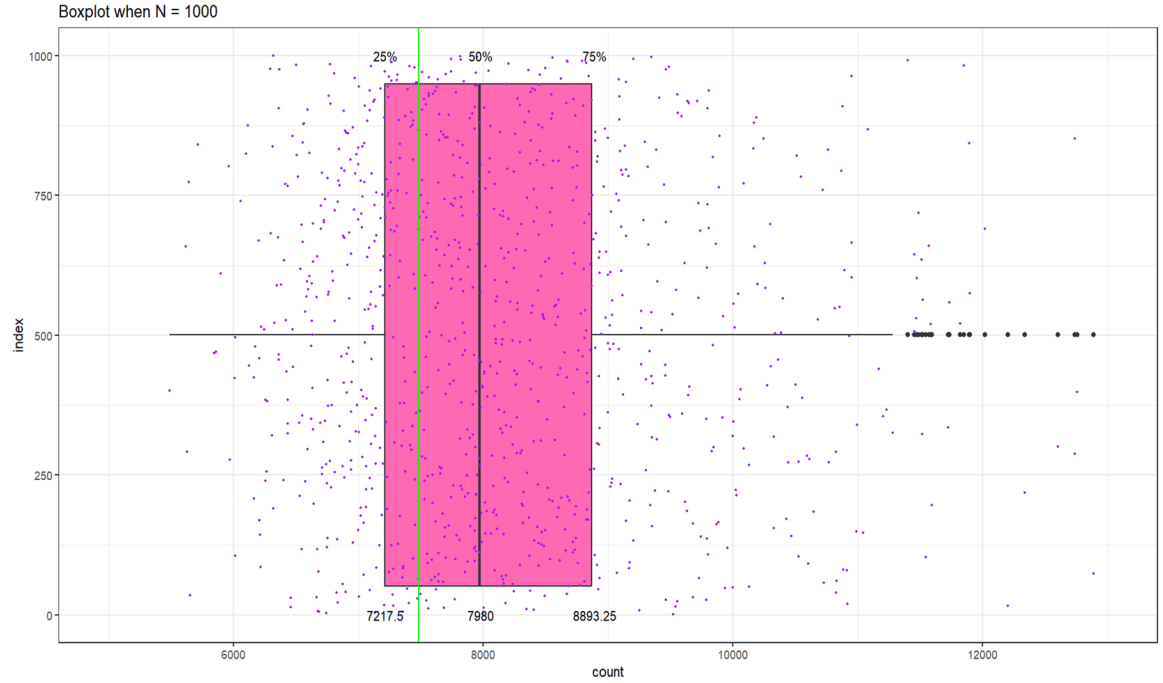
From the two scatterplots given above, we can say that the average number of proposals is approximately $\leq 1.5NlogN$

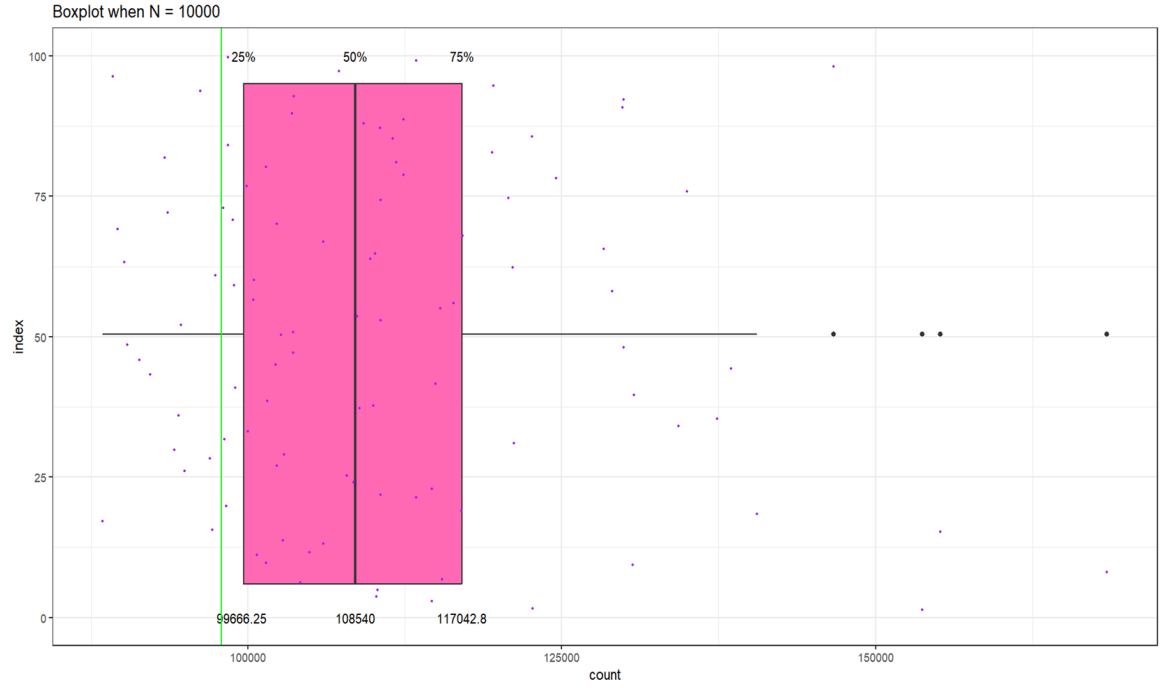
N	Avg Proposals	Worst Case Proposal	Standard Deviation
10	25.08	51	6.880513
100	540.411	1211	125.0827
1000	8207	17819	1386.656
10000	110621.1	168432	15507.23

Table 6.1: Statistics of the Number of Proposals







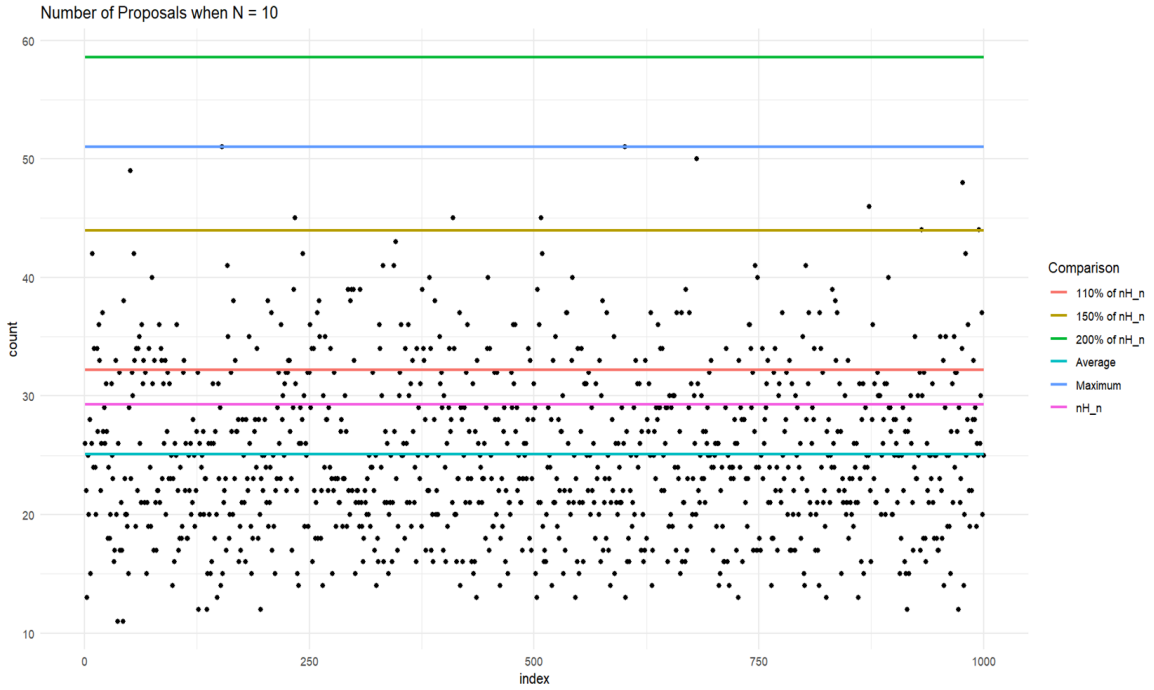


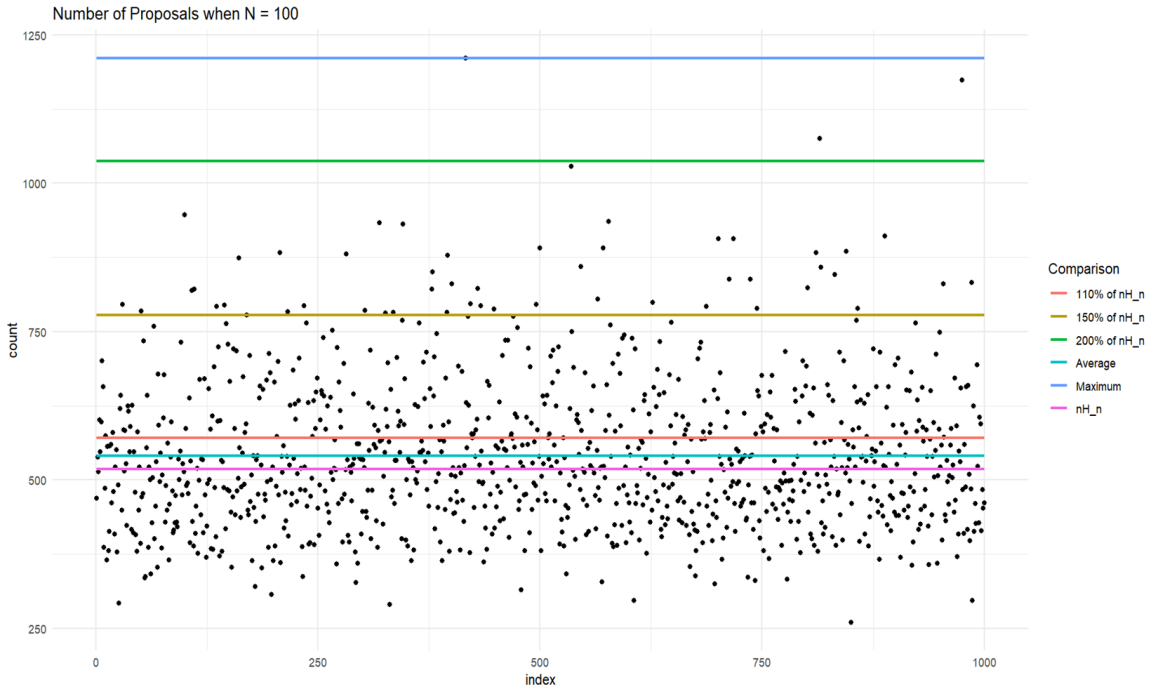
The bright green line in the boxplots represents nH_n for the corresponding value of n. From these plots, it is evident that as n increases, a higher proportion of observations exceed nH_n.

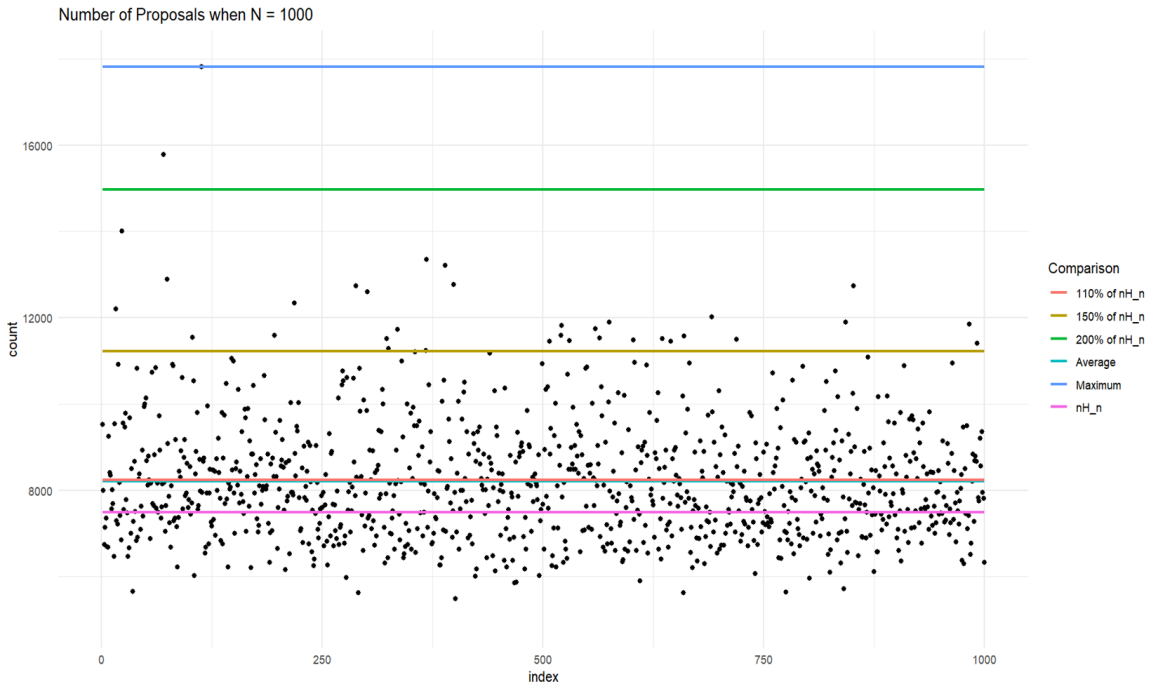
The above statistics are obtained by randomly generating 1000 samples for each N = 10, 100, 1000, and 100 samples for N = 10000

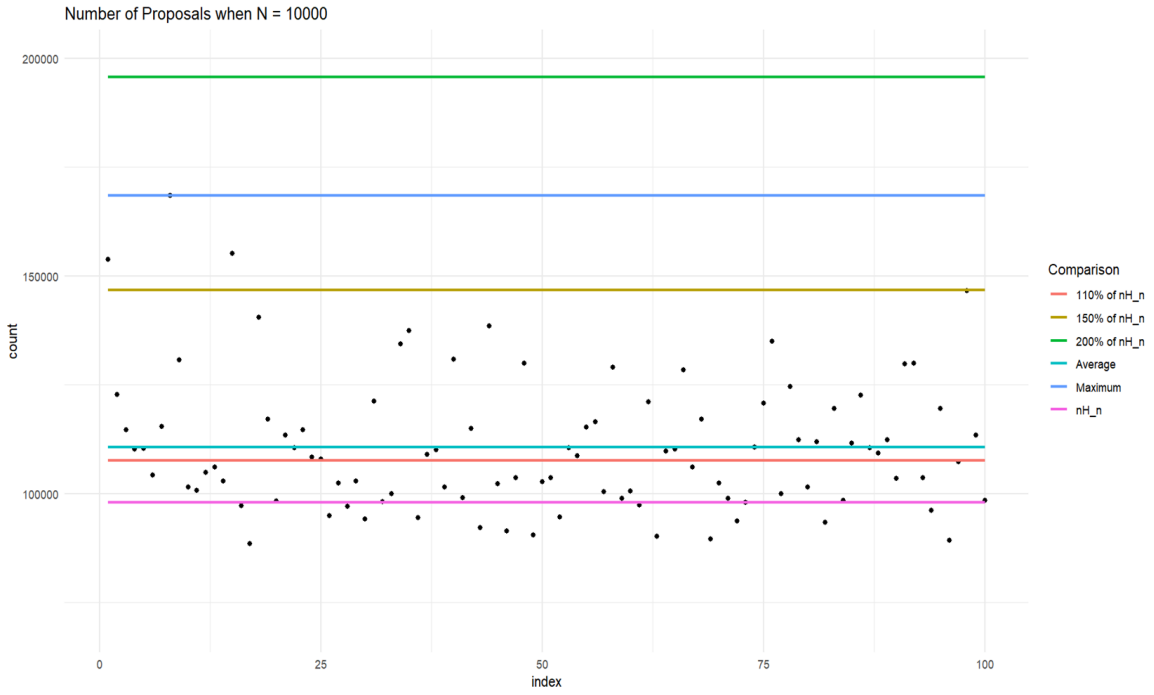
Avg Proposals exceeded by(%)	N = 10	N = 100	N = 1000	N = 10000
10	15.2	33.6	42.5	52
20	7.9	22.4	23	24
50	1.1	5.2	3.4	3
100	0	0.3	0.2	0

Table 6.2: Percentage exceeding expected proposals









The above plots provide some useful insights

1. There are almost negligible points which have value $\geq 2nH_n$ for all n
2. As n increases, the average number of proposals are getting closer to each other and getting more concentrated in the region $(1.1nH_n, 2nH_n)$

7 Conclusion

1. We drew the scatterplot for different values of n in range (1, 3000) for 10 randomly generated samples for each value of n. The first observation was that it is much less than n^2. Comparing it with different lines, it is close to nH_n.

2. Drawing the log scatterplot for them ensures that the line is some constant times $n \log n$.

3. The statistics of the average number of proposals reveal that increasing n leads to increasing standard deviation.

4. With increasing n, more and more number of proposals are exceeding nH_n and they are getting concentrated in a smaller range.

In conclusion, the Gale-Shapley algorithm offers an efficient solution to the stable matching problem, especially when preference lists are chosen randomly and uniformly. The probabilistic analysis presented in this project sheds light on the algorithm's behaviour under various conditions and highlights its potential for real-world applications.

Future research could focus on further optimizing the Gale-Shapley algorithm and exploring its applications in different domains. Additionally, more extensive experiments could be conducted to validate the findings of this study and provide deeper insights into the algorithm's performance characteristics.