# INDIAN INSTITUTE OF TECHNOLOGY KANPUR

SURGE 2023

# Comparison of Different Matching Algorithms in Two-Sided Matching

Prepared By

**Shabadpreet Singh**
**Dept. of Mathematics and Statistics**

Mentored by

**Dr. Soumyarup Sadhukhan**
**Dept. of Mathematics and Statistics**

12 JULY 2023

## Contents

# Acknowledgement

I would like to express my heartfelt gratitude to Professor Soumyarup Sadhukhan for their invaluable guidance and support throughout the course of this research project. Their expertise, knowledge, and unwavering commitment to academic excellence have been instrumental in shaping the outcome of this report.

I am extremely grateful to Professor Soumyarup for devoting so much time and energy to reviewing and providing constructive feedback on various phases of this research. Their meticulous attention to detail, insightful suggestions, and insightful comments have substantially improved the calibre and rigour of this work.

I would also like to recognise Professor Soumyarup's generosity in sharing their profound insights and expertise in the field. Their mentoring and intellectual discussions have expanded my comprehension of the topic and inspired me to pursue new lines of inquiry.

Lastly, I would like to thank Professor Soumyarup for their unwavering support and encouragement, which served as a constant source of motivation throughout the duration of this research project. Their confidence in my abilities and willingness to invest their time and knowledge were crucial to my academic achievement.

I am extremely privileged to have had the opportunity to work under Professor Soumyarup's supervision. In addition to shaping my academic trajectory, their mentoring has instilled in me a passion for research and a dedication to excellence. I am extremely appreciative of their contributions to this endeavor and their ongoing commitment to fostering an atmosphere of academic excellence. This report would not have been feasible without their incalculable contributions.

I am also indebted to Professor Debasis Mishra for the project report. His note on Mechanism Design (Mishra (2008)) helped me to understand the topic in great detail. Some of the definitions and examples presented in this report are taken from that note.

Date: SHABADPREET SINGH
12/07/2023 Dept. of Mathematics and Statistics

**Abstract**

We consider the both-sided matching framework where one side has to be matched with the other side, and both sides have preferences over the other side. In this setting, there are three widely recognized matching algorithms: Deferred Acceptance, Top-trading cycle, and Serial Dictatorship. In this study, we intend to compare the above matching algorithms based on their expected rank-utility under the uniform distribution on the preference profiles.

## 1. Introduction

The *matching theory* is a subfield in game theory where the goal is to match one side with the other. It can be broadly classified into two parts: (a) One-Sided Matching and (b) Two-Sided Matching. One-sided matching is popularly known as an *object assignment* problem where one side represents the agents, and the other represents the objects. In this model, agents have preferences over the objects, but the objects do not have any preferences. In two-sided matching, both sides represent agents, and they have preferences over the other side.

Matching is one of the most practically applied game-theoretic branches. It has many applications in real-world scenarios, such as school choice, kidney exchange programs, online advertising auctions, allocating housing units, and many more. It is also used in studying labor markets, where firms are matched with workers, or allocating public resources, where public goods are assigned to citizens. In all these contexts, matching is used to design mechanisms that allocate resources among agents efficiently and fairly.

There are several well-known matching algorithms in the literature. In two-sided matching, one such algorithm is **TTC**, which works given an initial endowment of the objects. Another important algorithm, in this context, is the **Serial Dictatorship** algorithm which assumes an ordering over the agents, and the agents get to pick their favorite object according to that ordering. In two-sided matching, the *Gale-Shapley algorithm* is a well-known solution *concept*, which provides a method for finding stable matching in two-sided matching problems. A matching is stable if no pair of agents (one from each side) have any incentive to break the matching. The Gale-Sahpley algorithm has been used in various practical applications,such as the National Resident Matching Program for medical residency placements in the United States.

The purpose of this paper is to undertake a thorough comparison of the three most prominent matching algorithms in two-sided matching scenarios, DA, TTC, and SD. By analyzing the advantages and disadvantages of both algorithms, we hope to obtain a better understanding of their performance, stability, and fairness. This analysis will provide researchers, practitioners, and policymakers with a greater comprehension of the trade-offs involved when selecting an algorithm for matching in a given context.

## 2. Basic Framework

Let $\boldsymbol{M} = \{M_1, \ldots, M_n\}$ be a set of men and $\boldsymbol{W} = \{W_1, \ldots, W_n\}$ be a set of women where $n \in \mathbb{N}$. Every man $M \in \boldsymbol{M}$ has a strict preference $\succ_M$ over the set of women and every women $W \in \boldsymbol{W}$ has a strict preference $\rhd_W$ over the set of men.[1] For a preference $\succ$, by $r_k(\succ)$, we denote the $k^{th}$ ranked alternative in $\succ$, i.e., $r_k(\succ) = W$ iff $|\{X \in \boldsymbol{W} \mid XPW\}| = k$. For $W_x, W_y \in \boldsymbol{W}$ and $M \in \boldsymbol{M}$, $W_x \succ_M W_y$ means $M$ prefers $W_x$ over $W_y$. Similar notations can be defined for a preference $\rhd$ over $\boldsymbol{M}$. We denote the set of all strict preferences over the set $\boldsymbol{W}$ as $\mathbb{L}(\boldsymbol{W})$ and set of all strict preferences over the set $\boldsymbol{M}$ as $\mathbb{L}(\boldsymbol{M})$. By a preference profile $(\succ, \rhd)$, we mean a $2n$-tuple of preferences $(\succ_{M_1}, \ldots, \succ_{M_n}, \rhd_{W_1}, \ldots, \rhd_{W_n})$ where $\succ_{M_i}$ is the preference of man $M_i$ and $\rhd_{W_j}$ is the preference of woman $W_j$ for $i, j \in \{1, \ldots, n\}$.

A matching $\mu : \boldsymbol{M} \to \boldsymbol{W}$ is a bijective mapping, i.e., it matches a man with a woman. For a matching $\mu$ and $M \in \boldsymbol{M}$, $\mu(M)$ denotes the women, $M$ is matched with in $\mu$. Similarly, for a women $W \in \boldsymbol{W}$, $\mu^{-1}(W)$ denotes the man, $W$ is matched with in $\mu$. We denote the set of all matchings by $\mathscr{M}$.

Below, we define an important and well-known property of a matching. It's called stability. A matching is stable at a preference profile if there is no pair of a man and a woman who prefer themselves over their partners in the matching. Stability ensures that no pair of a man and a woman has any incentive to break the matching.

**Definition 2.1.** A matching $\mu$ is **unstable** at a preference profile $(\succ, \rhd)$ if there exists $M \in \boldsymbol{M}$ and $W \in \boldsymbol{W}$ such that $W \succ_M \mu(M)$ and $M \rhd_W \mu^{-1}(W)$. The pair $(M, W)$ is called a **blocking pair** of $\mu$ at $(\succ, \rhd)$. A matching $\mu$ is **stable** at $(\succ, \rhd)$ if there is no blocking pair of $\mu$ at $(\succ, \rhd)$.

For example, consider the following preference profile:

| $\succ_1$ | $\succ_2$ | $\succ_3$ | $\rhd_1$ | $\rhd_2$ | $\rhd_3$ |
|---|---|---|---|---|---|
| $W_3$ | $W_3$ | $W_2$ | $M_3$ | $M_2$ | $M_3$ |
| $W_2$ | $W_2$ | $W_3$ | $M_2$ | $M_3$ | $M_1$ |
| $W_1$ | $W_1$ | $W_1$ | $M_1$ | $M_1$ | $M_2$ |

[1]A strict preference over a set $S$ is a complete, reflexive, antisymmetric, and transitive binary relation on $S$.

The matching $(M_1 : W_1, M_2 : W_3, M_3 : W_2)$ will be *unstable* as both $M_1 \& W_3$ will prefer each other over their current match, therefore, forming a *blocking pair*.
A *stable matching* for the above preference would be $(M_1 : W_1, M_2 : W_2, M_3 : W_3)$. No *blocking pair* exists.

A *matching function* $f$ is a mapping from $\mathbb{L}(\boldsymbol{W})^n \times \mathbb{L}(\boldsymbol{M})^n \to \mathscr{M}$, i.e., for every preference profile $(\succ, \rhd)$, matching function assigns a matching. A matching function is *stable* if it gives a stable matching at every preference profile. In the following, we define another desirable property of a matching function, called *strategy-proofness*. Strategy-proofness guarantees that no one has any incentive to misreport their true preference. In game-theoretic language, it says that truth telling is a dominant strategy.

**Definition 2.2.** A matching function $f : \mathbb{L}(\boldsymbol{W})^n \times \mathbb{L}(\boldsymbol{M})^n \to \mathscr{M}$ is **strategy-proof for men** if for all $M_i \in \boldsymbol{M}$, and all $(\succ, \rhd) \in \mathbb{L}(\boldsymbol{W})^n \times \mathbb{L}(\boldsymbol{M})^n$, and all $\succ'_{M_i} \in \mathbb{L}(W)$,

$$f(\succ, \rhd)(M_i) \succ_{M_i} f((\succ'_{M_i}, \succ_{-M_i}), \rhd)(M_i).$$

Similarly, we can define **strategy-proofness for women**. A matching function is **strategy-proof** if it is strategy-proof for both men and women.

Consider a scenario where there are an equal number of men and women who need to be paired up for marriage. Each individual ranks the members of the opposite gender based on their preferences. The goal is to create stable marriages where no two individuals prefer each other over their current partners.

The *Deferred Acceptance* algorithm is a strategy-proof mechanism that guarantees a stable matching. It works as follows:

- Initially, each man proposes to his most preferred woman.
- Each woman collects all her proposals and rejects all but her favorite proposal.
- Any rejected man proposes to his next preferred woman who has not rejected him yet.
- The process continues until every woman is paired with a man.

Let's see why this algorithm is strategy-proof. Suppose a man tries to manipulate the algorithm by proposing to a woman he does not truly prefer. This woman, according to his ranking, may not be his top choice, but he thinks she is more likely to accept his proposal. However, since the woman collects all proposals and chooses her favorite, she will still reject him if she prefers someone else.

By proposing truthfully, each participant maximizes their chances of getting matched with their most preferred partner. Deviating from truthful preferences will not lead to a better outcome. Hence, the Deferred Acceptance algorithm is strategy-proof in ensuring a stable and fair matching.

A matching at a preference profile is *Pareto optimal* if, for every other matching, there exists atleast one man or woman who gets a strictly better match in the current matching. Mathematically speaking, a matching $\mu$ is Pareto optimal at $(\succ, \rhd)$ if for every $\mu'(\neq \mu)$, either there is $M \in \boldsymbol{M}$ with $\mu(M) \neq \mu'(M)$ and $\mu(M) \succ_M \mu'(M)$ or there is $W \in \boldsymbol{W}$ with $\mu^{-1}(W) \neq \mu'^{-1}(W)$ and $\mu^{-1}(W) \rhd_W \mu'^{-1}(W)$. A matching function is *Pareto optimal* if it produces a Pareto optimal matching at every preference profile.

**Example :** Considering the preference profile 2, in this case, one possible Pareto efficient matching could be $(M_1 : W_1, M_2 : W_2, M_3 : W_3)$.

In this matching, no man can be assigned to a woman they prefer more than their current assignment. Similarly, no woman can be assigned a man she prefers more than her current assignment. Thus, there is no alternative matching that would make everyone involved happier, making this matching *Pareto efficient*.

# 3. Three Important Matching Algorithms

## 3.1 Deferred Acceptance (DA)

The **Deferred Acceptance (DA)** mechanism, also known as the Gale-Shapley algorithm, is a widely used algorithm in matching theory for solving the stable matching problem. It works as follows:

1. Each agent in one set proposes a tentative match to their most preferred agent in the other set.

2. Each agent receiving proposals considers all the proposals they have received and tentatively accepts the proposal from their most preferred proposer. They reject all other proposals.

3. Agents who have been rejected in the previous round repeat the process by proposing to their next preferred agent who has not yet rejected them.

4. The process continues in multiple rounds, with agents proposing and considering proposals until no more proposals are made or rejected.

5. Once the process reaches a stable state, where no further proposals or rejections occur, the matchings are finalized. Each agent is matched with the partner they tentatively accepted during the process.

The deferred acceptance mechanism guarantees the existence of a stable matching, where there are no pairs of agents who would both prefer to be matched with each other rather than their current partners. It ensures that no agent has an incentive to break their assigned match and form a new pair.

**Example :** Consider the preference profile at 2, applying DA;

$$\textbf{Step 1} \begin{cases} M_1, M_2 \text{ proposes} W_3 \\ M_3 \text{ proposes} W_2 \end{cases}$$

As $M_1 \rhd_3 M_2$, $W_3$ rejects $M_2$, and $M_1, M_3$ gets temporarily matched with $W_3, W_2$.

$$\textbf{Step 2 :} M_2 \text{ proposes } W_2(M_3)$$

As $M_2 \rhd_2 M_3$, $W_2$ rejects $M_3$(her initial match) and accept the proposal of $M_2$.

$$\textbf{Step 3 :} M_3 \text{ proposes } W_3(M_1)$$

As $M_3 \rhd_3 M_1$, $W_3$ rejects $M_1$(her initial match) and accept the proposal of $M_3$.

$$\textbf{Step 4 :} M_1 \text{ proposes } W_2(M_2)$$

As $M_2 \rhd_2 M_1$, $W_2$ rejects proposal of $M_1$.

$$\textbf{Step 4 :} M_1 \text{ proposes } W_1$$

Having received no other proposal $W_1$ accepts his proposal.

$\therefore$ The final matchings are $(M_1 : W_1, M_2 : W_2, M_3 : W_3)$.

DA is *strategy-proof* and produces *stable* and *Pareto-efficient* matching over all preferences of men and women, though it may not produce Pareto-optimal matching from one side.

### 3.2 Top-trading cycle (TTC)

The **Top Trading Cycle (TTC)** algorithm is a mechanism used to solve the problem of two-sided matching, where agents have preferences over potential matches. It works as follows:

1. The algorithm starts by identifying cycles within the preferences. A cycle occurs when a set of agents can be matched in a way that every agent in the cycle prefers their assigned match to any available unmatched agent outside the cycle.

2. Within each cycle, agents are allowed to trade their current matches with other agents in the cycle to improve their outcomes. The trades occur simultaneously, meaning all exchanges within the cycle happen at once.

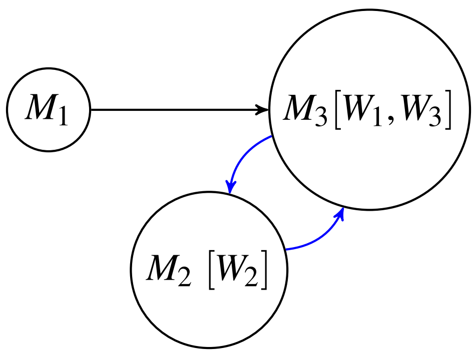
3. The algorithm continues identifying and resolving cycles until no more cycles can be found.

4. Once all cycles have been resolved, the matches are finalized, and each agent is assigned to their respective match.

A TTC with fixed endowment mechanism is *strategy proof and always produces Pareto-optimal matching from men's side.*

**Example :** Consider the preference profile at 2, applying TTC by constructing a directed graph with 3 nodes, one for each man, and putting a directed edge from node of $M_i$ to node of $M_j$ if the top-ranked object of $M_i$ is endowed to $M_j$.

In each graph, men who are part of the *cycle with blue edges* can trade their endowments.



A cycle exists between $M_2$ and $M_3$, therefore exchange of endowments occur, $M_2$ and $M_3$ are matched with $W_3$ and $W_2$ respectively.
Having no other choice, $M_1$ gets matched with $W_1$.

$\therefore$ The final matchings are $M_1 : W_1, M_2 : W_3, M_3 : W_2$.

We notice that even though the matching is *unstable*, it is still Pareto optimal from the men's side.

### 3.3 Serial Dictatorship (SD)

The **Serial Dictatorship (SD)** mechanism is a method for allocating individuals or agents to each other based on their preferences in a one-by-one sequential manner. It involves a sequence of "dictators" who have the authority to make decisions and determine the pairings or matchings. The serial dictatorship with a fixed priority is strategy-proof and efficient.

**Example :** Consider the preference profile at 2, applying SD with priority sequence as $M_1 \rightarrow M_2 \rightarrow M_3$.

**Step 1 :** $M_1$ gets matched to $W_3$.
**Step 2 :** $M_2$ gets matched to $W_2$.
**Step 3 :** $M_3$ gets matched to $W_1$.

Serial Dictatorship is *strategy proof* from men's side but may produce *unstable* matching.

## 4. R code for the three algorithms

**Deferred Acceptance**

```r
###Deferred Acceptance####

DA <- function(pref_M, pref_W) {

  n <- dim(pref_M)[2] # number of men = women

  # Initialize everyone to be free i.e. unmatched
  unmarried_men <- seq(1, n)
  unmarried_women <- seq(1, n)
  engagements <- c()

```

```r
# While there are still unmarried men
while (length(unmarried_men) > 0) {

  # Choose an unmarried man
  man <- unmarried_men[1]

  # Get the man's preference list
  prefs <- pref_M[, man]

  # Go through each of the women on the man's preference list
  for (i in 1:n) {
    woman <- prefs[i]

    # Get the woman's preference list
    woman_prefs <- pref_W[, woman]

    # If the woman is unmarried, they become engaged
    if (woman %in% unmarried_women) {

      engagements <- rbind(engagements, c(man, woman))
      unmarried_men <- unmarried_men[-1]
      unmarried_women <- unmarried_women[unmarried_women != woman]
      break
    }

    # Otherwise, check if the woman prefers this man over her current
 partner
    else {
      current_man <- engagements[which(engagements[, 2] == woman), 1]

      if (which(woman_prefs == man) < which(woman_prefs == current_man)) {
        engagements <- engagements[-which(engagements[, 2] == woman), ]
        engagements <- rbind(engagements, c(man, woman))
        unmarried_men <- c(unmarried_men[-1], current_man)
        break
      }
    }
  }
}

```

```
  return(engagements[order(engagements[, 1]), ])
}
```

**Top Trading Cycle**

```
###dfs(Depth First Search)####

dfs <- function(pointing, parent.node, current.node, visited)
{
  if(current.node == parent.node){
    return(visited)
  }
  if(current.node %in% visited){
    return(NULL)
  }else{

    visited <- c(visited, current.node)
    current.node <- pointing[current.node, 2]

    return(Recall(pointing, parent.node, current.node, visited))
  }
}

###Temp_Matching function####
temp_match <- function(endowments, new.pref.m, unmatched_W)
{
  len <- length(new.pref.m)
  pointing <- matrix(c(1:len, rep(0, len)), ncol = 2)

  for(i in 1:len)
  {
    bool <- c()
    for(j in 1:len){
      bool <- c(bool, ( new.pref.m[i]  %in% endowments[[j]]) )
    }
    pointing[i, 2] <- which(bool == TRUE)
  }

  chain <- rep(NA, len)
  global <- c()
  for(i in 1:len)
```

```
  {
    parent.node <- i
    visited <- i
    current.node <- pointing[i, 2]

    if(i %in% global){next}

    chain[i] <- list(dfs(pointing, parent.node, current.node, visited))
    global <- c(global, chain[[i]])
  }

  global <- global[order(global)]

  #find list of men successfully matched to their first pref using igraph and store it in 'men'
  final.match <- matrix(c(global, new.pref.m[global]), ncol = 2)

  return(final.match)
}


###TTC####
TTC <- function(pref_M, pref_W)
{
  n <- dim(pref_M)[2] # number of men

  unmatched_M <- seq(1:n)
  unmatched_W <- seq(1:n)
  matched <- NULL

  while(length(unmatched_M) > 1)
  {
    new.pref.m <- c() # finding new preferences of unmatched men by removing matched women
    new.pref.w <- c() # new preferences for women


    for(i in unmatched_M){
      new.pref.m <- cbind(new.pref.m, pref_M[!(pref_M[, i] %in% matched[, 2]), i])
```

```r
    }
    for(i in unmatched_W){
      new.pref.w <- cbind(new.pref.w, pref_W[!(pref_W[, i] %in% matched[, 1])
    , i])
    }

    endowments <-rep(NA, n)

    for(i in 1:n){
      endowments[i] <- list(unmatched_W[ which(new.pref.w[1, ] == i) ])
    }

    endowments <- endowments[unmatched_M]  # as endowments for already
    matched doesn't exists

    temp <- temp_match(endowments, new.pref.m[1, ], unmatched_W)
    matched <- rbind(matched, matrix(c(unmatched_M[temp[, 1]], temp[, 2]),
    ncol = 2))

    unmatched_M <- unmatched_M[!(unmatched_M %in% matched[, 1])]
    unmatched_W <- unmatched_W[!(unmatched_W %in% matched[, 2])]
  }

  # if only 1 unmatched men and women remains they will be matched together
    no need to run loop again which saves time
  if(length(unmatched_M == 1)){
    matched <- rbind(matched, c(unmatched_M, unmatched_W))
  }

  return(matched[order(matched[, 1]), ])
}
```

**Serial Dictatorship**

```r
###Serial Dictatorship####

# returns matrix of nx2 with n being number of men and second column being
    the women matched to each men according to pref_m(preferences of man)
SD <- function(pref_M)
{
  n <- dim(pref_M)[2] # number of men
```

```
  match <- numeric(n) # to store matching

  #we are taking the priority order to be {1, 2..., n} with 1 being highest
    and n being lowest priority men
  for(i in 1:n)
  {
    match[i] <-pref_M[!(pref_M[,i] %in% match) , i][1]   #assigns first
    unmatched women to i'th man acc to his pref
  }

  return(matrix(c(1:n, match), ncol = 2))
}
```

## 5. Our Contribution

In this work, we intend to compare the three algorithms based on some criteria that, in some sense, take into account the utilities the agents (men and women) get from the algorithm across profiles. One such natural measure is expected rank utility. Although this notion is less explored in the literature, it is intuitive and simple to understand and interpret. Below we discuss it in detail with the help of a few definitions.

For a matching $\mu$ at a preference profile $(\succ,\rhd)$, the rank utility of a person (a man or a woman) is $n+1$ minus the rank of their partner in the matching. Mathematically, for a man $M$, the rank utility is $n+1-r(\mu(M),\succ_M)$ and for a woman $W$, it is $n+1-r(\mu^{-1}(W),\rhd_W)$. Note that the rank utility of a person is proportional to the rank of their partner, the higher ranked partner someone gets, the more utility they get. The total rank utility at a preference profile is the sum of the rank utilities of all men and women, i.e., $\sum\limits_{M\in \boldsymbol{M}}[n+1-r(\mu(M),\succ_M)]+\sum\limits_{W\in \boldsymbol{W}}[n+1-r(\mu^{-1}(W),\rhd_W)]$. Finally, for a matching function, to combine the utilities across different profiles, we assume a probability distribution over the preference profiles and compute the expectation of the rank utility under this probability distribution. Let $p:\mathbb{L}(\boldsymbol{M})^n\cup\mathbb{L}(\boldsymbol{W})^n\to[0,1]$ be a pmf on the set of preference profiles, then for a matching function $f$, the expected rank utility w.r.t. $p$ is given by

$$\sum_{[(\succ,\rhd)\in\mathbb{L}(\boldsymbol{M})^n\cup\mathbb{L}(\boldsymbol{W})^n]}\left[\sum_{M\in\boldsymbol{M}}[n+1-r(\mu(M),\succ_M)]+\sum_{W\in\boldsymbol{W}}[n+1-r(\mu^{-1}(W),\rhd_W)]\right]p((\succ,\rhd)).$$

## 5.1 A PARTICULAR CASE WHEN $n = 3$ AND $p$ IS UNIFORM

To start with, we assume $n = 3$ and $p$ is uniform and compare the three algorithms in terms of their expected rank utilities. Note that under the uniform distribution, it's equivalent to taking the sum of the rank utilities across different profiles. To get some idea about the comparison, We first compute the total rank utilities of three algorithms using R. You may find the code below.

### 5.1.1 FINDINGS USING R CODE

The below function was used to calculate the score of men and women given their preferences and current match as input.

```r
##Score Function####
calc_score <- function(pref_M, pref_W, matching)
{
  M_Match <- matching[, 2]
  n <- dim(pref_M)[2]
  # Pref_M will be a n x n matrix, with column representing preferences of a man
  # and n representing the number of men
  # same for pref_W
  # matching is n x 2 matrix of pairs of men and women calculated by various algorithms(like ttc, da or sd)
  # M_match is a column vector having women who is matched with men of that index
  # similarly W_Match has men matched to women

  W_Match <- order(M_Match)
  score <- 0

  for(i in 1:n)
  {
    score <- score + (n+1 - which(pref_M[ ,i] == M_Match[i]) ) #Adding Men score
    score <- score + (n+1 - which(pref_W[, i] == W_Match[i]) ) #Adding Women score
  }

  return(score)
}
```

## Main Function

The below code was used to call and run all the functions(DA, TTC and SD) and calculate and store their score using the *Score Function*.

Here, we see that the total possible preferences in the case of 4 men and women case will be $4!^8$, which is approximately equal to $10^{11}$, and as we increase the number of men and women, total cases will increase drastically, making it computationally inefficient to consider each case.

$\therefore$ For $n \geq 4$, we did random sampling to get preference profiles of men and women and ran the loop for $10^6$ times and took average of it to get the mean score.

```r
source("Score_Function.R")  # loads 'calc_score' function
source("Serial_Dictatorship.R")  # loads 'SD' function
source("Deffered_Acceptance.R") # loads 'DA' function
# Example
# men <- matrix(c(1, 2, 3,
#                 2, 3, 1,
#                 1, 3, 2), nrow = 3)
# women <- matrix(c(2, 1, 3,
#                   3, 2, 1,
#                   3, 1, 2), nrow = 3)
# DA(men, women) # returns n pairs with first element as men and the other
    the women matched to him
#
source("TTC.R")  # loads 'TTC' function
# Example
# men <- matrix(c(1, 2, 3,
#                 2, 1, 3,
#                 3, 2, 1), ncol = 3)
#
# women <- matrix(c(1, 2, 3,
#                   3, 1, 2,
#                   2, 3, 1), ncol = 3)
# TTC(men, women)

#CALCULATING SCORES FOR 'n' MEN AND WOMEN CASE

#Random Sampling cases for n >= 4 (where 'n' is number of people)#####
n <- 10

n <- as.integer(readline(prompt = "Enter total number of men and women : "))
```

```r

library(pracma)
library(gtools)
way <- perms(seq(1 : (n/2) ))

score_ttc <- 0
score_da <- 0
score_sd <- 0


for(i in 1:1e6){

  if(i %% 1e4 == 0){print(i / 1e4)}

  prefs <-  sample(1:fact(n/2), n, replace = TRUE)

  pref_men   <- t(way[prefs[1:(n/2)], ])
  pref_women <- t(way[prefs[1:(n/2)], ])

  score_sd  <- score_sd  + calc_score(pref_men, pref_women, SD(pref_men))
  score_da  <- score_da  + calc_score(pref_men, pref_women, DA(pref_men, pref_women))
  score_ttc <- score_ttc + calc_score(pref_men, pref_women, TTC(pref_men, pref_women))
}

score_da  <- log(score_da)  - log(1e6) #Taking log as to increase Numerical stability
score_sd  <- log(score_sd)  - log(1e6)
score_ttc <- log(score_ttc) - log(1e6)
```

After running simulations and observing them we hypothesized the following statements:

1. According to total rank-utility, Deferred Acceptance is the best algorithm among Deferred acceptance, TTC, and serial dictatorship.
2. If we only consider the Men's Side while calculating rank-utility, TTC with women as objects is the best algorithm.

### 5.1.2 Results

In this section, we mathematically prove the findings, we get using R code in the previous section. We start with a Lemma that states an interesting fact about TTC and DA. At any profile, if a man is matched to his last-ranked women in DA, he will be matched with the same woman in TTC as well.

**Lemma 1.** *If a man is assigned to his last-ranked women in DA, then he will have the same assignment in TTC.*

**Proof:** Let $(\succ, \rhd)$ be a preference profile where for all $i \in \{1,2,3\}$, $M_i$ has preference $\succ_i$ and $W_i$ has preference $\rhd_i$, and $M_1$ is matched to her third preferred women in DA. Without loss of generality, let's assume $W_1 \succ_1 W_2 \succ_1 W_3$, and $M_2$ and $M_3$ are matched with $W_1$ and $W_2$, respectively in DA. Since the outcome of DA is stable, $W_1$ must prefer $M_2$ over $M_1$ and $W_2$ must prefer $M_3$ over $M_1$. Therefore, we have

$$\begin{array}{cc} \rhd_1 & \rhd_2 \\ \hline \vdots & \vdots \\ M_2 & M_3 \\ \vdots & \vdots \\ M_1 & M_1 \\ \vdots & \vdots \end{array}$$

Further, $M_2$ will prefer $W_1$ over $W_3$ and $M_3$ will prefer $W_2$ over $W_3$. To see this, assume for contradiction $M_2$ prefers $W_3$ over $W_1$. Since finally $M_2$ is matched with $W_1$, it means at some stage of the algorithm (say $t^{th}$ stage), $M_2$'s proposal to $W_3$ got rejected. Suppose $M_1$'s proposal to $W_3$ caused this rejection. Since $W_3$ is $M_1$'s least preferred woman, it must be that he had already proposed to $W_1$ and $W_2$. Therefore, by DA algorithm, $W_1$ and $W_2$ had at least one proposal at $t^{th}$ stage. But this is a contradiction as $W_3$ had two proposals at $t^{th}$ stage. Now assume $M_3$'s proposal to $W_3$ caused the rejection of $M_2$'s proposal to $W_3$ at the $t^{th}$ stage. Since finally $M_3$ is matched with $W_2$, it means, at some later stage, his proposal to $W_3$ also got rejected. But this rejection can only cause due to $M_1$'s proposal to $W_3$, which again means, at that stage, $W_3$ had two proposals, a contradiction. Thus, we have

$$\begin{array}{cc}
\succ_2 & \succ_3 \\
\hline
\vdots & \vdots \\
W_1 & W_2 \\
\vdots & \vdots \\
W_3 & W_3 \\
\vdots & \vdots
\end{array}$$

We claim that, in TTC, $M_1$ will be matched with $W_3$. Note that $M_2$ and $M_3$ have either $W_1$ or $W_2$ as their top preferred women. Moreover, $W_1$ and $W_2$ have either $M_2$ or $M_3$ as their top-preferred men. Therefore, in the first step of TTC, either they will point towards each other, or they will point towards themselves, or they both will point to either $M_2$ or $M_3$. If they both point to each other or themselves, they will get matched to $W_1$ and $W_2$, and as a result, $M_1$ will go with $W_3$. So, assume they both point to one of them. Since, in DA, $M_2$ got matched with $W_1$ and $M_3$ is matched with $W_2$, this means either both point to $M_2$ and $W_1$ is with $M_2$ initially, or both point to $M_3$ and $W_2$ is with $M_3$ initially. Suppose both point to $M_2$ and $W_1$ is with $M_2$ initially. Hence, $M_2$ will be matched with $W_1$. Further, as $M_3$ prefers $W_2$ over $W_3$ and $W_2$ prefers $M_3$ over $M_1$, $M_3$ will be matched to $W_2$ in the second round implying $M_1$ will be matched with $W_3$. The argument is similar when both point to $M_3$ and $W_2$ is with $M_3$ initially. This completes the proof of the lemma. ■

We have not been able to prove completely the findings mentioned in the previous section. In the following, we briefly describe our work plan for the future. Note that Lemma 1 immediately implies that if, at a preference profile, a man has more rank utility in TTC than in DA, it must be that he is getting his top-preferred woman in TTC and his second preferred woman in DA. We will use this fact to characterize the profiles where the total rank-utility of the men is strictly more in TTC than DA. Further, we will show that if, at a profile, TTC dominates DA with respect to the total rank-utility, it must be the case that the total rank-utility of the men is strictly more in TTC than DA. Thus, it's enough to characterize the profiles where TTC dominates DA in terms of their total rank-utility for the men.

## 6. METHODOLOGY

We implemented code for several algorithms, including the Top Trading Cycle (TTC), Deferred Acceptance (DA), and Serial Dictatorship (SD). Then, through conducting simulations using different preference profiles and analyzing the results, we derived meaningful insights that allowed us to establish two hypotheses and a lemma. To solidify our findings, we proceeded to mathematically formalize the proof for a lemma. This involved constructing a rigorous mathematical argument that logically demonstrates the validity of the lemma in question. The lemma serves as a definitive statement supported by mathematical evidence, providing a deeper understanding of the properties of DA and TTTC.

Overall, our research involved coding and simulating the TTC, DA, and SD algorithms, analyzing the results to generate hypotheses, and subsequently developing a mathematical proof to support our findings. This comprehensive approach allowed us to gain valuable insights into the matching processes and contribute to the understanding of their properties.

## REFERENCES

[1] MISHRA, D. (2008): "An introduction to mechanism design theory," *The Indian Economic Journal*, 56, 137–165.