Eindhoven University of Technology

BACHELOR

Organ allocation

Comparing online matching algorithms to maximize total welfare in an organ allocation system

Bliek, Noa

Award date:
2023

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Organ allocation

Comparing online matching algorithms to maximize total welfare in an organ allocation system

Bachelor Final Project

N. Bliek

1371525

Supervisor
dr. B. Smeulders

Eindhoven, July 2023

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Applied Mathematics
Combinatorial Optimization

Contents

1 Introduction

In the medical world, there are a lot of patients in need of organ transplants to survive or increase their quality of life (Burra & De Bona, 2007). In 2022, there were 1247 patients on a waiting list to receive an organ in the Netherlands (NTS, 2023). Of these patients, 938 were in need of kidney transplantation, which is also the longest waiting group of patients. Although there are numerous patients on waiting lists to secure a new organ that is compatible, it is not always in the best interest of the total welfare to match the available organ to the first patient on the waiting list. It is important for organs and patients to be compatible since the organs can otherwise be rejected or decrease the quality of life for a patient. This concept can be explained as a matching problem where patients are matched to organs based on several criteria.

There exist matching algorithms that are based on an offline environment where all information is known. For the donor allocation system, these algorithms would need to know the amount and quality of all the incoming organs. This information makes it possible to change a match to optimize the total welfare. However, this optimization can have a high computational cost and take a lot of time. When a certain match is made, one can come back later and change this match if this would increase the total welfare of the system. How these algorithms work, can be determined by the definition of the welfare of the system. One algorithm for this setting is a matching that minimizes the difference between the organs and patients. For this, a definition of difference would also need to be determined.

Organ transplantation does not uphold the assumptions of this offline setting. It cannot be determined upfront how many organs will be donated or what their exact qualities are. When there is an incoming organ, a fast decision has to be made without the knowledge of future incoming organs. A patient could be matched to an organ today, while a better-matching organ comes in tomorrow which would increase the total welfare of the system. By not matching an incoming organ, there is a risk that a patient cannot find another organ that is compatible and therefore this patient has no contribution to the total welfare of the system. It could be better to make any match rather than leave organs unmatched. This study is therefore looking for a solution to an online matching problem.

For this specific problem, there have been multiple different types of solutions that look at different constraints and try to optimize the solution in different ways. For instance, one can look at two integers, Kidney Donor Transplant Index and Expected Post-Transplant Survival, and try to minimize their distance for each patient (Mattei, Saffidine, & Walsh, 2017). This distance should be minimized to prevent organs from being rejected by surgeons when they don't accept the quality of the organ for their patient. This is done to prevent the patient's body to reject the organ after transplantation or there might be a chance that a better organ will be donated that increases the patient's quality of life further. In (Mattei et al., 2017), the quality of an organ is determined by the Kidney Transplant Index which gives an integer between 0 to 100 for each organ. The quality of a patient is determined by the Expected Post-Transplant Survival score which gives an integer between 0 to 100 for each patient. This matching is already applied to the organ donation setting, but there are also a lot of matching algorithms that have different settings. The point of view can be changed to make it applicable to the organ donation setting.

One of these algorithms to solve a matching problem is a ranking algorithm that would first assign a random ranking to the patients and then matches each incoming organ to the eligible patient of the highest rank if this is applicable (Karp, Vazirani, & Vazirani, 1990). In this case, a set of eligible patients is determined for each organ so that the organ will only be matched to one of these, or it will not be matched at all. Among the set of eligible patients, the organ will be matched to the one with the highest rank, or in this context that might be translated to the patient that has the highest quality.

One can also look at this problem as a function that represents the gain of a certain match between organ and patient based on the preference of both patient and organ (Li, 2008). In this case, both the organs and patients have a preference list and the total preference should be optimized. In this case, the total welfare would also be determined by the independent preference of the patients and organs.

There is also the classical secretary problem. First, a certain ranking is determined for each organ and their value is revealed one by one. For each organ, it can be decided to accept the organ or reject it, but it is not possible to go back to a previous organ. In the classical secretary problem, first, a fixed amount of incoming donors is determined and the highest value of the organs so far is noted. After this stopping criterion, the first incoming organ that has a higher value than all the last incoming organs is accepted. This matching algorithm looks specifically at the individual patient's interest to get a good match, which might not be in favor of the total welfare (Freeman, 1983).

For the donor allocation system, we would like to optimize the total welfare of all the matched organs and patients. We would need to know how we determine this total welfare and what we mean by a 'good' match. When we defined these characterizations of the matching problems, we can try to find algorithms that try to optimize this welfare using different organ and patient lists. We will look into three online algorithms, which are assortative matching, expected matching, and posted price matching. These algorithms will be discussed in the next chapters.

2 Problem Statement

Organ allocation is a graph theory problem where a graph is denoted as a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$. A 2-partite graph is considered where V creates a partition of 2 classes such that every edge in E has its end in different classes. The bipartition is O, P, with O the class of organs and P the class of patients. A matching M needs to be found where each patient is matched to an organ. M is a set of independent edges in our graph G. If for a subset $U \subseteq V$, $u \in e$ for every vertex $u \in U$ and every edge $e \in M$, then M is called a matching for U. It can then be said that the vertices in U are matched by the matching M. Each edge has a weight that is given by the minimum value of the patient and organ on its ends such that the value of a match for a patient with a transplant is determined by the part that has the lowest quality. We want to match each patient to an organ such that the total sum of the weights is maximized.

For this project, integer values from 1 up to n will be worked with, given to patients as well as organs to represent their quality, where n is determined by the number of patients. The determination of these integer values is outside the scope of this research. For the offline setting, after receiving a list of patients and organs, they will be sorted by their integer values such that the patient and organ with the lowest integer values, have the lowest index. This gives a vector **p** and **o**, with the sorted values of the patient and organ lists respectively. For an online setting, the patient list is known beforehand and will also be sorted. The organ list is not known and therefore the organ vector is unsorted and the first element of the organ vector denotes the first incoming organ, and so on.

The value of a patient with index (or position) i is denoted by p_i, and the value of an organ with index i is denoted by o_i, where $i \in \{1, \dots, n\}$. The weights of the edges are then determined by $\min(p_i, o_i)$. Our goal is to maximize the total welfare, which is defined by $\sum_{i=1}^{n} \min(p_i, o_j)$ for p_i in P and o_j in O. This study then tries to find the matching M that maximizes $\sum_{i=1}^{n} \min(p_i, o_j)$ for p_i in P and o_j in O. To do this, there are three settings that can be looked at.

2.1 Setting 1

We start by assuming that we have an ordered list of n patients that all have a distinct integer value from 1 to n equal to their index. We denote this by $p_i = i$ for $i \in \{1, \dots, n\}$, or with the patient vector $\hat{\mathbf{p}}$. The incoming organs all have a value from 1 to n that is uniformly distributed. After drawing these values, the organs are relabelled such that $o_1 \leq o_2 \leq \cdots \leq o_n$ which gives the organ vector $\hat{\mathbf{o}}$. In this setting, the patient and organ lists have the same length, and thus the classes of organs and patients have the same size. This means that $|P| = |O| = n$.

2.2 Setting 2

In another setting, we can change the patient list to make it more realistic. We will assume that the patient list has the same length n as the organ list where each patient has a value from 1 to n that is uniformly distributed. After drawing a patient list from a uniform distribution between 1 and n, we relabel these such that $p_1 \leq p_2 \leq \cdots \leq p_n$. We denote this by the organ patient vector $\tilde{\mathbf{p}}$. The organ vector is denoted by $\tilde{\mathbf{o}}$ In this scenario, the patient value and their index do not necessarily have the same integer value. The length of both lists are still equal such that $|P| = |O| = n$.

2.3 Setting 3

In section 1, we discussed that there are numerous patients on the waiting list to receive an organ. This third setting takes the different lengths for the organ and patient list into account, where in this case the patient list is twice as long as the organ list such that $|P| = 2n > |O| = n$. The patients are still uniformly distributed between 1 and n as well as the organ list. To account for the unequal patient and organ list, n times the value 0 is added to the organ list such that the lists have the same lengths. After drawing 2n patients from a uniform distribution between 1 and n, they are relabeled such that $p_1 \leq p_2 \leq \cdots \leq p_{2n}$, which is given by the patient vector $\dot{\mathbf{p}}$. After drawing n organs from a uniform

distribution, they are given the distinct indexes from $\{n+1, \ldots, 2n\}$ and the first n values are given by $0 = o_1 = o_2 = \cdots = o_n$. This is denoted by the organ vector $\dot{\mathbf{o}}$.

3 Theoretical approach

3.1 Efficient matching

In an offline setting, where there is knowledge of all the available organs and patients, the total sum of the matches can be optimized by determining the sum for each possible combination of matches. To determine this matching faster, an approach that can find an optimal match directly for an arbitrary list of organs needs to be found. There are matching algorithms that for instance have a time complexity $\mathbb{O}(n^\alpha)$ for some $\alpha \geq 2$. One example is the Hungarian algorithm that has a time complexity of $\mathbb{O}(n^3)$ (Jain et al., 2005). Sorting algorithms are faster than this and can for instance give a time complexity of $\mathbb{O}(n \log_2 n)$ (Xiang, 2011) for a list of length n. This is why we would like to see whether a sorting algorithm is beneficial to determine a matching.

First, a patient and organ list with the same length from subsection 2.1 or subsection 2.2 are assumed. With minor adaptations, this can also be used for lists with different lists, which will be discussed later. Suppose there are n patients with values $p_1, \dots, p_n$ where $0 < p_1 \leq p_2 \leq \cdots \leq p_n$, denoted by **p** as in section 2. Suppose there are n organs with values $o_1, \dots, o_n$ where $0 < o_1 \leq o_2 \leq \cdots \leq o_n$, denoted by **o** as in section 2. $\sum_{i=1}^{n} \min(p_i, o_{\pi(i)})$ needs to be maximized for some permutation π.

Lemma 1. $\sum_{i=1}^{n} \min(p_i, o_i) \geq \sum_{i=1}^{n} \min(p_i, o_{\pi(i)})$ *for any permutation* π.

Proof. Suppose we have $b \in \{1, \dots, n\}$ where b is the highest value with $\pi(b) \neq b$, and $\pi(a) = b$ and $\pi(b) = c$ for some $a, c \in \{1, \dots, n\}$.

Then $p_b \geq p_a$ and $o_{\pi(a)} = o_b \geq o_{\pi(b)} = o_c$.
It is essential to show $\min(p_a, o_c) + \min(p_b, o_b) \geq \min(p_a, o_b) + \min(p_b, o_c)$. This is done for all cases;

- If $p_b \geq p_a \geq o_b \geq o_c$:
  $\min(p_a, o_c) + \min(p_b, o_b) = o_c + o_b \geq o_b + o_c = \min(p_a, o_b) + \min(p_b, o_c)$
- If $p_b \geq o_b \geq p_a \geq o_c$:
  $\min(p_a, o_c) + \min(p_b, o_b) = o_c + o_b \geq p_a + o_c = \min(p_a, o_b) + \min(p_b, o_c)$
- If $p_b \geq o_b \geq o_c \geq p_a$:
  $\min(p_a, o_c) + \min(p_b, o_b) = p_a + o_b \geq p_a + o_c = \min(p_a, o_b) + \min(p_b, o_c)$
- If $o_b \geq p_b \geq o_c \geq p_a$:
  $\min(p_a, o_c) + \min(p_b, o_b) = p_a + p_b \geq p_a + o_c = \min(p_a, o_b) + \min(p_b, o_c)$
- If $o_b \geq p_b \geq p_a \geq o_c$:
  $\min(p_a, o_c) + \min(p_b, o_b) = o_c + p_b \geq p_a + o_c = \min(p_a, o_b) + \min(p_b, o_c)$
- If $o_b \geq o_c \geq p_b \geq p_a$:
  $\min(p_a, o_c) + \min(p_b, o_b) = p_a + p_b \geq p_a + p_b = \min(p_a, o_b) + \min(p_b, o_c)$

So $\min(p_a, o_c) + \min(p_b, o_b) \geq \min(p_a, o_b) + \min(p_b, o_c)$. Since this applies for arbitrary $a, b, c \in \{1, \dots, n\}$ where b is the highest value with $\pi(b) \neq b$, a better matching can always be found by matching p_b to o_b.

So $\sum_{i=1}^{n} \min(p_i, o_i) \geq \sum_{i=1}^{n} \min(p_i, o_{\pi(i)})$ for any permutation π. □

This proves that when the full organ and patient list are available, both lists can be ordered and matched per index. When using the setting from subsection 2.3, the organ and patient lists are not of the same length, but lemma 1 can still be used. If the patient list is longer, organs can be added with a value of 0 until the lists have the same length. Each patient can then be matched to an organ with the same index, which means that the patients with the lowest values will be matched to a value of 0, indicating that there is no organ available for these patients.

3.2 Expected offline matching value

In the donation process, the value of each organ is not known beforehand. This can only be determined at the moment of arrival. Since it is assumed that the organs are coming in with a known independent distribution for their values, the expected value of an incoming organ can be found and therefore the expected value of a certain match. This will be useful for the offline matchings that will be investigated such as the posted price matching or expected matching. For this reason, it is necessary to determine the probability that an incoming organ has a certain value. Given lemma 1, it is interesting to take indices into account when determining the expected matching values.

Suppose there are n patients with values $p_1, \dots, p_n$ where $0 < p_1 \leq p_2 \leq \cdots \leq p_n$, denoted by the patient vector p. Suppose there are n organs with values $o_1, \dots, o_n$ where $0 < o_1 \leq o_2 \leq \cdots \leq o_n$ denoted by the organ vector o. The expected value can be calculated for each match $(p_1, o_1), \dots, (p_n, o_n)$.

Lemma 2. *Suppose $o_1, \dots, o_n$ from **o** as in section 2. Then*

$$\mathbb{P}(o_k = l) = \sum_{j=n-k+1}^{n} \left(\binom{n}{j} \left(\frac{n-l+1}{n}\right)^j \left(\frac{l-1}{n}\right)^{n-j} - \binom{n}{j} \left(\frac{n-l}{n}\right)^j \left(\frac{l}{n}\right)^{n-j} \right) \quad \text{for } k, l \in \{1, \dots, n\}$$

Proof. $\mathbb{P}(\text{exactly } j \geq l)$ has a binomial distribution with $p = \left(\frac{l-1}{n}\right) \implies$

$$\begin{aligned}
\mathbb{P}(o_k = l) &= \mathbb{P}(o_i \geq l \text{ for } i \geq k) - \mathbb{P}(o_i \geq l+1 \text{ for } i \geq k) \\
&= \sum_{j=n-k+1}^{n} \mathbb{P}(\text{exactly } j \geq l) - \sum_{j=n-k+1}^{n} \mathbb{P}(\text{exactly } j \geq l+1) \\
&= \sum_{j=n-k+1}^{n} \binom{n}{j} \left(\frac{n-l+1}{n}\right)^j \left(\frac{l-1}{n}\right)^{n-j} - \sum_{j=n-k+1}^{n} \binom{n}{j} \left(\frac{n-l}{n}\right)^j \left(\frac{l}{n}\right)^{n-j} \\
&= \sum_{j=n-k+1}^{n} \left(\binom{n}{j} \left(\frac{n-l+1}{n}\right)^j \left(\frac{l-1}{n}\right)^{n-j} - \binom{n}{j} \left(\frac{n-l}{n}\right)^j \left(\frac{l}{n}\right)^{n-j} \right)
\end{aligned}$$

□

The probability of an organ with index k in an ordered list is found to be equal to a number $l \in \{1, \dots, n\}$. This can be used to calculate the expected value of an organ that we match to a patient with the same index k. This is given by $\mathbb{E}[o_k] = \sum_{i=1}^{n} i \cdot \mathbb{P}(o_k = i)$ for every index $k \in \{1, \dots, n\}$.

The calculated expected organ values can be compared to the simulated average organ values to check our work and whether they match. By taking the difference between the calculation and the simulation, it is found that they differ by at most 0.01 as seen in Figure 1.

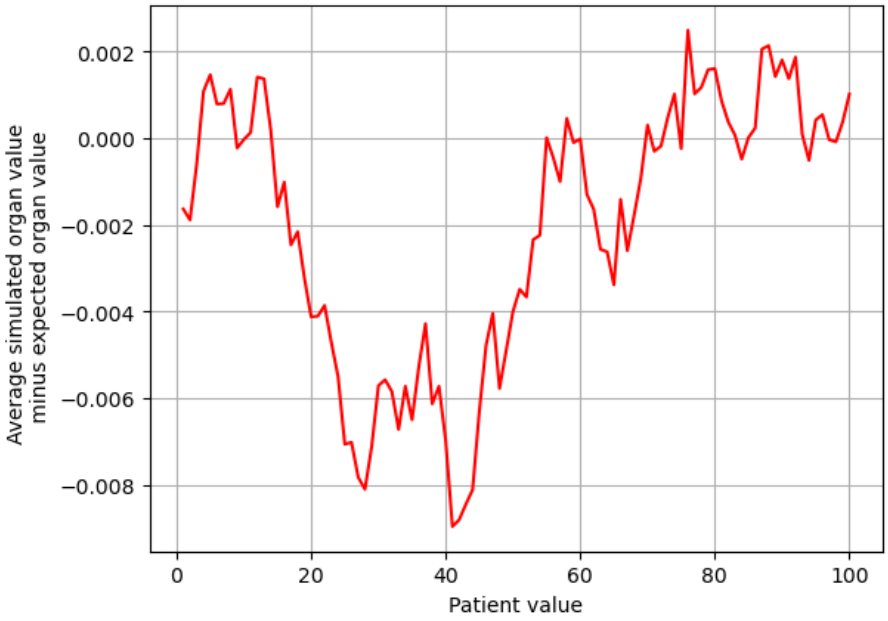

Figure 1: Comparison of expected organ value to simulated organ value with 1 million runs and setting 1.

The probabilities of the values for each incoming organ can also be used to calculate the expected match value which is given by the minimum value of the organ and patient value. For the setting where a completely known patient list from 1 to n as in subsection 2.1 is available, the following expected match value is obtained.

$$\mathbb{E}[match(p_k, o_k)] = \sum_{i=1}^{n} \min(i, k) \cdot \mathbb{P}(o_k = i) \text{ for } k \in \{1, \dots, n\}$$

The calculated expected match values can be compared to the simulated average match values. By taking the difference between the calculation and the simulation, it is clear that they differ by at most 0.005 as seen in Figure 2.

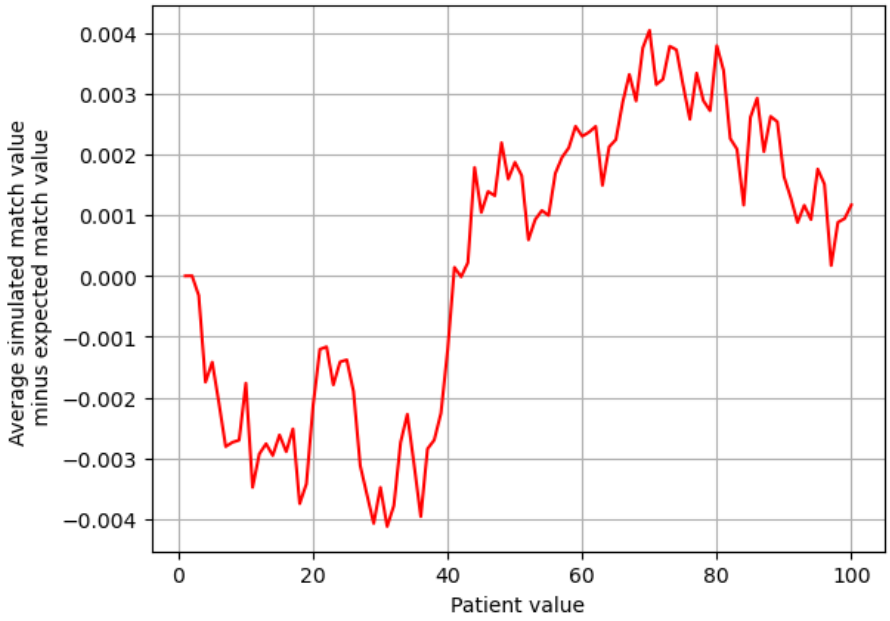

Figure 2: Comparison of expected match value to simulated match value with 1 million runs and setting 1.

With a setting as in subsection 2.2 where the patient list is also unknown, but from the same distribution, there are different calculations for the expected value. To determine the new expected match values, the probability of a patient with index k having a certain value needs to be determined, which is also uniformly distributed from 1 to n, similarly to the organ list. For this, lemma 2 can be used to find that

$$\mathbb{P}(p_k = l) = \sum_{j=n-k+1}^{n} \left(\binom{n}{j} \left(\frac{n-l+1}{n}\right)^j \left(\frac{l-1}{n}\right)^{n-j} - \binom{n}{j} \left(\frac{n-l}{n}\right)^j \left(\frac{l}{n}\right)^{n-j} \right) \text{ for } k, l \in \{1, \dots, n\}.$$

Then the expected patient value for a certain index $k \in \{1, \dots, n\}$ can be found which is given by $\mathbb{E}[p_k] = \sum_{i=1}^{n} i \cdot \mathbb{P}(p_k = i)$ for every index $k \in \{1, \dots, n\}$. This can be used to calculate the expected match value when both organ and patient lists are uniformly distributed between 1 and n for a total of n patients and n organs.

$$\mathbb{E}[match(p_k, o_k)] = \sum_{i=1}^{n} \sum_{j=1}^{n} \min(i, j) \cdot \mathbb{P}(o_k = i) \cdot \mathbb{P}(p_k = j) \text{ for } k \in \{1, \dots, n\}.$$

The calculated expected match values can be compared to the simulated average match values. This resulted in a difference of at most 0.006 between the calculation and the simulation as seen in Figure 3.

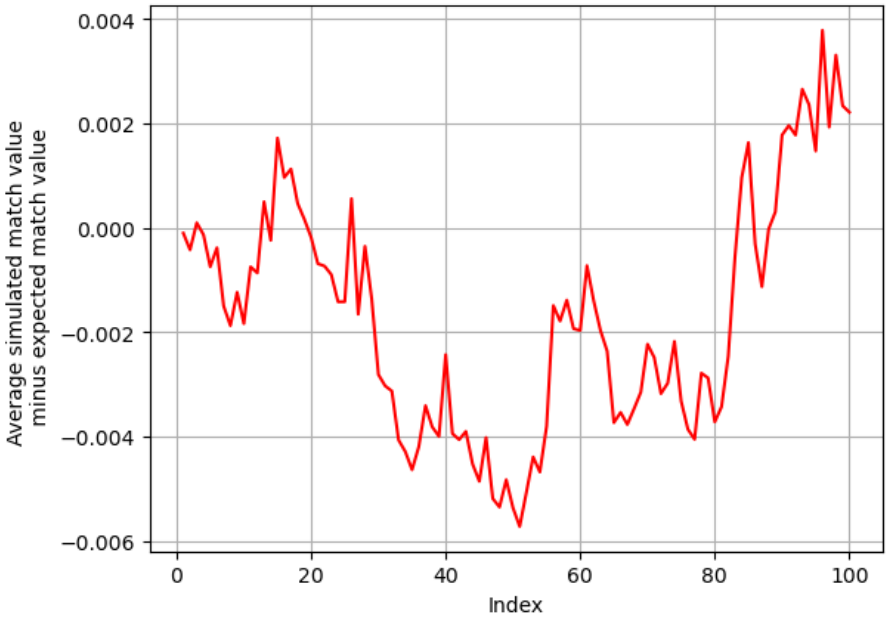

Figure 3: Comparison of expected match value to simulated match value with 1 million runs and setting 2.

In the third setting as in subsection 2.3, the length of the patient list is twice as large such that $|P| = 2n$. This means that the probability of a patient with index k having a certain value is different than for the other settings. To find this, lemma 1 can be used which gives

$$\mathbb{P}(p_k = l) = \sum_{j=2n-k+1}^{2n} \left(\binom{2n}{j} \left(\frac{n-l+1}{n}\right)^j \left(\frac{l-1}{n}\right)^{2n-j} - \binom{2n}{j} \left(\frac{n-l}{n}\right)^j \left(\frac{l}{n}\right)^{2n-j} \right) \text{ for } k \in \{1,\dots,2n\}, l \in \{1,\dots,n\}.$$

This gives the expected value for a patient with index k by $\mathbb{E}[p_k] = \sum_{i=1}^{n} i \cdot \mathbb{P}(p_k = i)$ for $k \in \{1,\dots,2n\}$. To be able to use lemma 1, the patient and organ lists are given the same length by adding n times the value 0 such that the organ vector is $\dot{\mathbf{o}}$ as in subsection 2.3. This gives $\mathbb{E}[o_k] = 0$ for $k \in \{1,\dots,n\}$ and from lemma 2 it is given that $\mathbb{E}[o_k] = \sum_{i=1}^{n} i \cdot \mathbb{P}(o_{k-n} = i)$ for $k \in \{n+1,\dots,2n\}$.

Together, this gives the expected match values

$$\mathbb{E}[match(p_k, o_k)] = \begin{cases} 0 & \text{for } k \in \{1,\dots,n\} \\ \sum_{i=1}^{n} \sum_{j=1}^{n} \min(i,j) \cdot \mathbb{P}(o_{k-n} = i) \cdot \mathbb{P}(p_k = j) & \text{for } k \in \{n+1,\dots,2n\} \end{cases}$$

The calculated expected match values can be compared to the simulated average match values. For the first 100 indexes of matches, the match difference is equal to 0 since these are never matched in both the simulated matching and the optimal offline matching. For the next 100 indexes, this comparison resulted in a difference of at most 0.01 between the calculation and the simulation as seen in Figure 4.

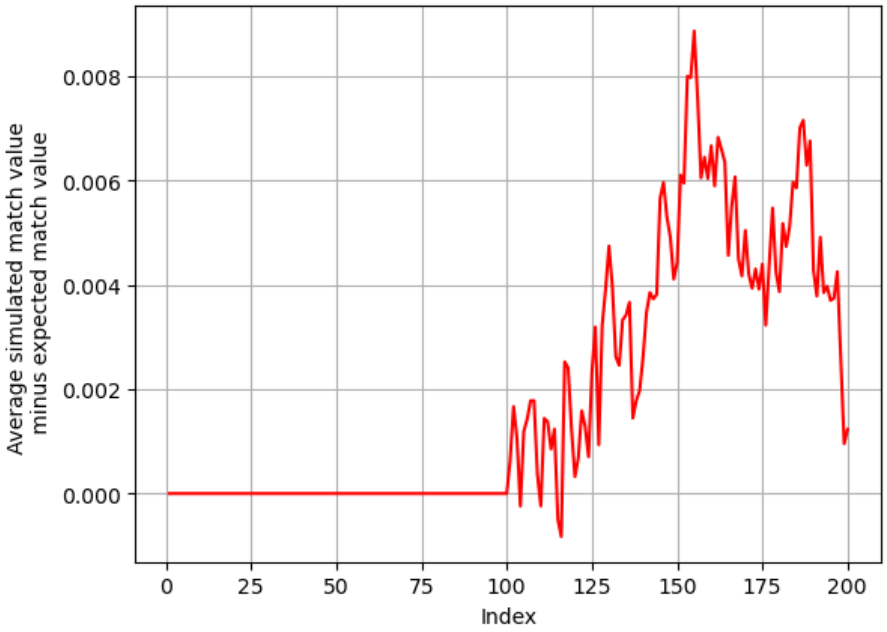

Figure 4: Comparison of expected match value to simulated match value with 1 million runs and setting 3.

Thus far, this study has concentrated on the expected patient value based on their index since this gives a way to optimally match patients to organs. In this situation, when for instance multiple patients have the same value, their expected value is determined by the random order that these same-valued patients are placed in. The expected match value can also be calculated per patient based on the value of the patient instead of its index. In this situation, a patient does not need to know where on the list it is placed, but it can determine its expected value based on its current value regardless of the values or positions of other patients. Patients with the same value are then also given the same expected value. These expected values can be calculated by doing multiple runs with a different patient list and a different organ list for each run.

For the setting from subsection 2.2, with 10 patients and 10 organs, the following expected patient values are given for 100 runs:

Patient value α	Expected match value for patient $p_i = \alpha$
1	1.0
2	1.73
3	2.55
4	3.46
5	4.30
6	5.20
7	6.12
8	7.12
9	8.29
10	9.28

Since lemma 1 is based on the indexes of the patient and organs, these new expected match values are further from the expected match values gained by the offline optimal matching.

4 Matching algorithms

In lemma 1, it is shown that matching per index maximizes the total sum of the values. In an offline setting, this would translate to sorting both the patient list and the organ list and matching them pairwise per index, given the assumption that both lists have the same length. If they do not have the same length, the value 0 can be added multiple times to the smallest list until the lists have the same length. This way, the left-over patients or organs will be matched to a value of 0, indicating that they did not find a match from the other list. The offline matching for all three settings from section 2 is given by Algorithm 1.

Algorithm 1 Offline matching

Input: organ_list, patient_list
Output: match_list, sum
  $sum \leftarrow 0$
  $o \leftarrow$ sort(organ_list)
  $p \leftarrow$ sort(patient_list)
  for index i in o **do**
    match_list$[i] = (p_i, o_i)$
    $sum+ = \min(p_i, o_i)$
  end for

In an online setting, the full patient or organ list is not known beforehand, but the previously calculated expected values can be used to get close to the optimal offline matching since our expected values are based on this offline matching. With these calculations, the matching can be used that is called expected matching where the difference between the value of the current match and the expected match value is minimized. Another online matching is the posted price matching where the calculated expected values are also used. Another online matching that will be discussed is assortative matching, where matches are not made based on expected values but based on the difference between the organ and patient values. To evaluate how well these online matchings work, they need to be compared to the offline setting and to each other.

4.1 Assortative matching

Assortative matching, as well as the expected matching, is similar to the matching from (Mattei et al., 2017) where the difference between two values is minimized. In the current study, for each incoming organ, we would try to minimize the difference between that organ's value and the values of the patients on the waiting list. Only an organ list and a patient list is needed. For this assortative matching, the pseudocode from Algorithm 2 is used.

Algorithm 2 Assortative matching

Input: organ_list, patient_list
Output: match_list, sum
  $sum \leftarrow 0$
  unmatched_patient_list ← patient_list
  for o_i in organ_list **do**
    for p_j in unmatched_patient_list **do**
      find difference between o_i and p_j
    end for
    match o_i to p_k in unmatched_patient_list with minimal difference
    match_list.append($(p_k, o_i,)$)
    $sum+ = \min(p_k, o_i)$
    remove p_k from unmatched_patient_list
  end for

Since the values of the matched patients and organs are close together, we use the values of the patients and organs optimally. Matching a patient with a value of 100 to an organ with a value of 10 gives an addition of only 10 to the total welfare, which means that we lose 90 from the value of the patient therefore matching the patient and organ values closely could be beneficial to the total welfare. However, we can be in a situation where we have a higher-value patient coming later when all the high-valued organs are already matched. If this happens often, we also lose a lot of value and it would increase the total welfare to not have all the high-valued patients matched in the beginning. For this reason, it might be interesting to look at the expected values calculated in section 3.

4.2 Expected matching

In another online matching algorithm, the expected values calculated in section 3 can be used to have the benefit of assortative matching, while also taking into account the organs that will be donated in the future and still have to be matched. An organ list, a patient list, and a list of the expected patient values are needed. Then, for each incoming organ, it is tried to minimize the difference between the match values with the patients and the expected values. For this expected matching, the pseudocode from Algorithm 3 is used.

Algorithm 3 Expected matching

Input: organ_list, patient_list, expected_match _value_list
Output: match_list, sum
  $sum \leftarrow 0$
  unmatched_patient_list ← patient_list
  for o_i in organ_list **do**
    for p_j in unmatched_patient_list **do**
      determine match value: $\min(o_i, p_j)$
      find minimal difference between $\min(p_j, o_i)$ and expected match values
    end for
    match o_i to p_k in unmatched_patient_list with minimal difference
    match_list.append((p_k, o_i))
    $sum+ = \min(p_k, o_i)$
    remove p_k from unmatched_patient_list
  end for

With this matching, we try to match the patient and organ value as closely as possible, but we account for a situation that we do not account for with assortative matching. We match based on these expected values and therefore try to decrease the chance that we have to match a high-valued organ to

a lot lower-valued patient or vice versa. Another matching where we can use the calculated expected values is the posted price matching.

4.3 Posted price matching

The posted price matching is based on the matching algorithm from (Feldman, Gravin, & Lucier, 2014), where the setting is based on an auction instead of organ matching. This matching pre-calculates a price τ_p for every patient in p. This price is based on our calculated expected patient values such that $\tau_{p_i} = \frac{1}{2}\mathbb{E}[match(p_i, o_i)]$. Then each incoming organ from o is matched to the available patient from p that is preferred with this price given that such a patient is available. This matching tries to maximize the total weight of the matches minus the price given by $\min(p_i, o_i) - \frac{1}{2}\mathbb{E}[match(p_i, o_i)]$. The pseudocode for this matching is shown in Algorithm 4.

Algorithm 4 Posted price matching

Input: organ_list, patient_list, expected_match_value_list
Output: match_list, sum
  $sum \leftarrow 0$
  unmatched_patient_list ← patient_list
  for p_j in patient_list **do**
    determine the prize list for p_j (half the expected value)
  end for
  for o_i in organ_list **do**
    for p_j in unmatched_patient_list **do**
      determine the difference between min(p_j, o_i) and prizes
    end for
    if all differences negative **then**
      leave o_i unmatched
    else
      match o_i to p_k with biggest positive difference
    end if
    match_list.append((p_k, o_i))
    $sum+ = \min(p_k, o_i)$
    remove p_k from unmatched_patient_list
  end for

With this matching algorithm, the patients have a higher standard since not every organ is accepted. It will not accept a match that is too far from the expected match value. For the current setting where individual welfare is disregarded, this matching algorithm might not perform as well as the other matching algorithms, but it is interesting to see how much this individual preference influences this total welfare.

For the third setting, from subsection 2.3, the prize for the first n patients is set at 0 since the expected match values are equal to 0. Only the patients with index n+1 till 2n are expected to find a match when there are n organs.

5 Results & comparisons

To compare the algorithms, multiple runs are done that each output the matching sum for the offline and all online matchings, given the same organ and patient list per run. This can be done for each setting from section 2.

For the first setting where $|P| = |O| = n$, the patient vector is $\hat{\mathbf{p}}$ and the organ vector is $\hat{\mathbf{o}}$, it can be seen in Figure 5 how well the online matchings perform in comparison to the offline setting where all information is known beforehand. 100 runs are done where all four matching algorithms use the same patient and organ list per run.

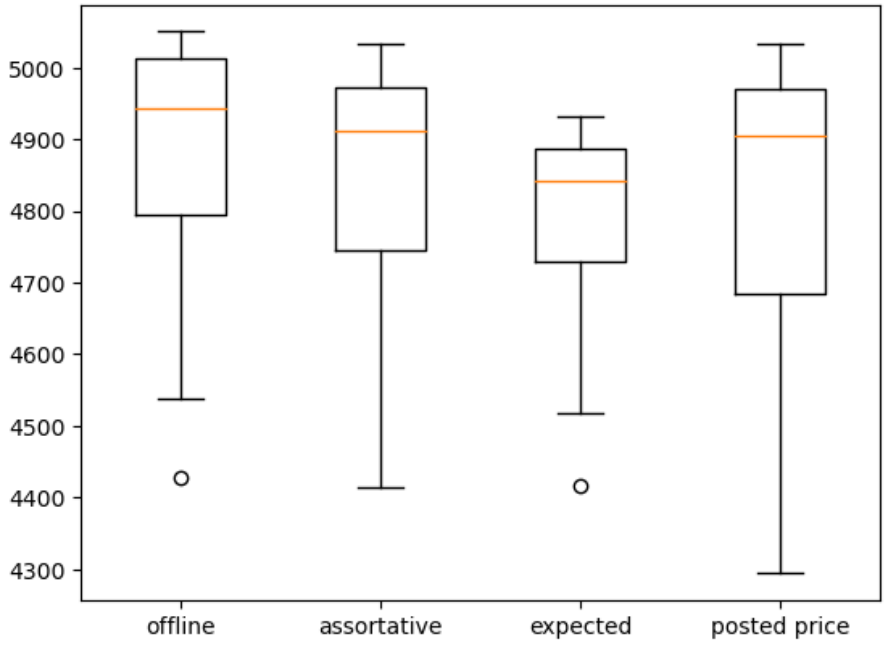

Figure 5: Boxplot comparison between offline, assortative, expected and posted price matching for 100 runs for setting 1.

The range of the total sums of the matchings is between 4300 and 5100. When Figure 5 is looked at, it can be seen that the mean values for the total sums appear to be unequal. To check whether the means are equal, a Wilcoxon signed-rank test can be done that does not need the assumption of Normality. It tests the null hypothesis that two related paired samples come from the same distribution. When running the test for each online algorithm against the offline algorithm, all of them give a p-value below 0.05. This means that the null hypothesis should be rejected that stated that the samples come from the same distribution.

The algorithms can then be compared for the second setting where $|P| = |O| = n$, but now the patient vector $\tilde{\mathbf{p}}$ and the organ vector $\tilde{\mathbf{o}}$are used. Again, 100 runs are done and the same patient and organ list are used per run for all four matching algorithms. The result can be seen in Figure 6

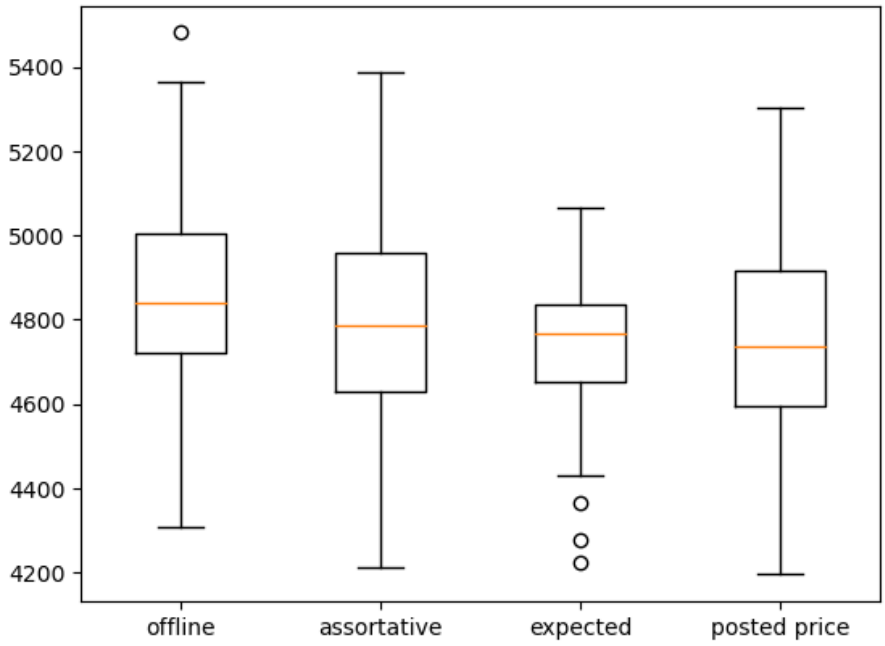

Figure 6: Boxplot comparison between offline, assortative, expected, and posted price matching for 100 runs for setting 2.

Based on Figure 6, the distributions of the four algorithms seem similar and they appear to have a mean close to each other. To test this, again, the Wilcoxon signed-rank test can be used since not all four algorithms show a normal distribution. Doing the Wilcoxon signed-rank test for all the possible combinations of two algorithms gives a p-value below the significance level $\alpha = 0.05$, except for the combination of the posted price matching and the expected matching. This means that for all other combinations, the null hypothesis must be rejected and it can be concluded that the distributions or means are statistically different, but the posted price and expected matching still fall under the hypothesis that they are not statistically different at a confidence level of 5%.

It is interesting to find out why the outcomes of the matching algorithms differ and in what cases a certain matching works significantly worse. One important observation to note is that the posted price matching leaves organs unmatched in certain situations for setting 1 and setting 2. The posted price matches an incoming organ only when there is a patient available that is preferred at the price given to it. In the settings used for this research, matching an organ to any patient always increases the total welfare rather than leaving organs unmatched. The other online matchings always match organs to patients. Most of the time that the posted price matching performs less than the other online matching algorithms, the unmatched patient list with the posted price algorithm is not empty. For the first setting, however, the posted price algorithm still has a slightly higher total mean value than for the expected matching algorithm based on Figure 5. So overall, the higher standards for the patients do not necessarily imply a worse outcome.

For the third setting where $|P| = 2n > |O| = n$, the organ vector $\dot{\mathbf{o}}$, and the patient vector $\dot{\mathbf{p}}$, it can be seen in Figure 7 how well the matchings perform compared to each other.

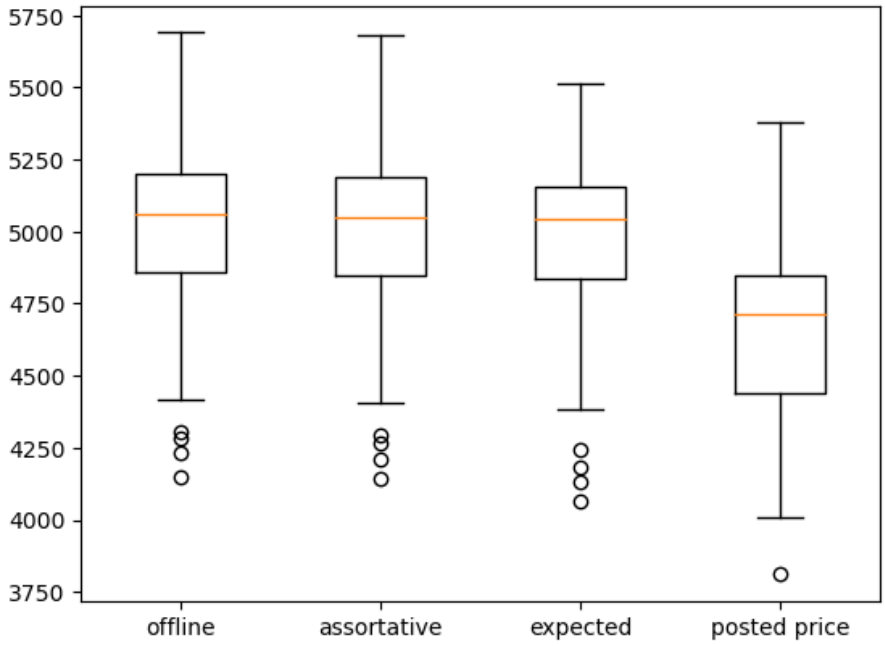

Figure 7: Boxplot comparison between offline, assortative, expected, and posted price matching for 100 runs for setting 3.

The assortative matching and expected matching appear to perform similarly to the offline matching. By performing the Wilcoxon singed-rank test, it is shown that they are statistically different from each other and the offline matching algorithm. Clearly, the posted price mechanism performs a lot worse than all the other matching algorithms. In this case, this is not clearly caused by leaving organs unmatched, but observation is that it often matches organs to a lower-valued patient because it expects higher-valued organs to come in later. The way this works can be shown with a simple example for 6 patients and 3 organs. Suppose we have the patient vector $\dot{\mathbf{p}} = (1, 1, 1, 1, 2, 2)$, then the price vector, determined by half times the expected match value per index, is given by $(0, 0, 0, 0.64, 0.95, 1.30)$. If the first incoming organ o_1 has value 3, it is matched to patient p_5 with value 2. The second incoming organ $o_2 = 3$ is matched to $p_1 = 1$. The third incoming organ $o_3 = 1$ is matched to $p_2 = 1$. This gives a sum for the total match values equal to 4. Simply matching $o_2 = 3$ to the still unmatched patient $p_6 = 2$ would already increase the sum of the total match values. Both the assortative and expected matching algorithms would return a higher sum for the total match values in this situation.

6 Conclusion

With each setting, this study gets a bit closer to a real-life organ allocation, where the third setting, subsection 2.3, is closest to this real-life organ allocation system where a lot of patients are on a waiting list. With the results of this study, we tried to maximize the total welfare of the organ allocation system and to give an insight into what kind of algorithms can help maximize this welfare.

The results show that for all three settings from section 2, the online algorithms perform differently. In the first setting, subsection 2.1, the expected matching algorithm has a lower total welfare, but the variance is smaller than for the posted price and assortative matching algorithms. In the second setting, subsection 2.2, the mean total welfare for the assortative matching is slightly higher than for the expected and posted price matching. The latter two matching algorithms are even statistically indifferent according to the Wilcoxon signed-rank test. However, the expected matching algorithm has more outliers and a smaller variance than both the posted price and assortative matching algorithms. For the third setting, subsection 2.3, the assortative and expected matching algorithms are closest to the outcome of the offline matching and have similar variance and some outliers, while the posted price matching performs noticeably worse than all the other matching algorithms.

In general, for the three settings, the assortative matching performs closest to the optimal offline matching. This is on the one hand surprising since this algorithm does not take into account the values of the future incoming organs. On the other hand, since we try to optimize the sum for the minimum values between patients and organs, the assortative matching loses less of this minimum value at the start of the matching since the values of the patients and organs lie close to each other.

7 Discussion

For this study, the outcome of a certain algorithm is greatly determined by the definition of the organ and patient values, the match values, and the total welfare of the system. The patient and organ values were determined by integer values between 1 and n, but in reality, determining these values might be a difficult task and are hard to express in a list of integers. Next to that, in organ allocation, a certain organ might have a negative impact on the life expectancy of a patient if for instance the blood types of the organ and patient do not match. This study disregards the case where leaving an organ unmatched is better than matching all the organs.

The value of a match was determined by $\min(p_i, o_j)$ for p_i in P and o_j in O. This was based on the idea that if the body of a patient, aside from the organ in need of transplantation, has a life expectancy of x, and the new organ has a life expectancy of y, then the life expectancy of the patient with this organ is as high as it's worst working part. Another determination of the value of a match would result in completely different expected match values since the lemma that is used to calculate these, lemma 1, uses this definition of the match values. It might be interesting to consider a different definition for the value of a match.

In this study, the goal was to find the matching M that maximizes $\sum_{i=1}^{n} \min(p_i, o_j)$ for p_i in P and o_j in O from the point of view of the total welfare. A different goal would change the algorithms that are interesting to look at. Next to that, the best interest of an individual patient is disregarded in this study as it only looks at the total welfare. This was investigated a bit with the posted price matching algorithm, but the goal was not unchanged. In a real-life setting, patients might have a say in which organs they want to accept or not and they might have even higher standards than what is now used in the posted price matching algorithm.

For the third setting from subsection 2.3, there was only looked at a length difference of n between the organ and patient list. It might be interesting to see at which difference the online matching algorithms perform close to the offline matching algorithm and for what difference it performs a lot worse.

Most of the expected values were calculated based on the index of a patient, but at the end of section 3, it was explored what the expected match values were based on the patient values instead of their index. This can be explored more and it could also be implemented for the online algorithms.

One of the observations for setting 1 and 2 was that the posted price matching usually performs worse than the assortative or expected matching when it leaves patients and organs unmatched. A way to improve the total sum for the posted price matching is to first look at the preferred organ with the set price, and otherwise find the next best organ. This could be looked into for future research on using the posted price algorithm for organ allocation.

For future research, it is interesting to look into other online algorithms that could optimize the total welfare even more. This can then also be compared to the algorithms used in this study. All the algorithms were now compared to the offline algorithm based on the mean total sum values. Aside from looking at different values for n or doing more runs, another way of determining how well a matching performs can also be investigated.

References

Burra, P., & De Bona, M. (2007). Quality of life following organ transplantation. *Transplant International*, *20*(5), 397–409.

Feldman, M., Gravin, N., & Lucier, B. (2014). Combinatorial auctions via posted prices. In *Proceedings of the twenty-sixth annual acm-siam symposium on discrete algorithms* (pp. 123–135).

Freeman, P. (1983). The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, 189–206.

Jain, A. K., Zhou, Y., Mustufa, T., Clif Burdette, E., Chirikjian, G. S., & Fichtinger, G. (2005). Matching and reconstruction of brachytherapy seeds using the hungarian algorithm (marshal). *Medical physics*, *32*(11), 3475–3492.

Karp, R. M., Vazirani, U. V., & Vazirani, V. V. (1990). An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual acm symposium on theory of computing* (pp. 352–358).

Li, H. (2008). Assortative matching. *The New Palgrave Dictionary of Economics. Second Edition, Palgrave Macmillan*.

Mattei, N., Saffidine, A., & Walsh, T. (2017). Mechanisms for online organ matching. In *Ijcai* (pp. 345–351).

NTS. (2023, Jan). *Jaarcijfers 2022: Hoogste aantal orgaandonoren in één jaar.* Retrieved from https://www.transplantatiestichting.nl/nieuwsartikel/jaarcijfers-2022-hoogste-aantal-orgaandonoren-in-een-jaar

Xiang, W. (2011). Analysis of the time complexity of quick sort algorithm. In *2011 international conference on information management, innovation management and industrial engineering* (Vol. 1, pp. 408–410).