University of Utrecht

Bachelor Thesis

Matching Theory and the Allocation of Kidney Transplantations

Kim de Bakker

Supervised by
Dr. M. Ruijgrok

14 June 2016

Introduction

Matching Theory has been around for many years. It was usually applied in the labour markets. Since the early 2000s it has also been applied to other problems for example paired kidney transplantations.

The study into matching theory started in 1962 with the article 'College Admission and the Stability of Marriage' by D. Gale and L.S. Shapley. This is the first appearance of matching theory in literature but matching theory was already being applied in hospitals. The hospitals were using an algorithm which matched interns with hospital programs. They did this without knowing that they were using matching theory.

It was only until the early 2000s that people saw a connection between matching theory and paired kidney transplantations. Alvin E. Roth, Tayfun Sönmez and M. Utku Ünver wrote the article 'Pairwise Kidney Exchange' in 2004. In this article they explained how matching theory could be applied and they talked about the possible restrictions.

In this thesis we start with explaining the basics of matching theory. We do this by starting with one-to-one matching in section 1. Here we use the most common example of one-to-one matching namely matching men and women. This is the type of matching which was first explained by Gale and Shapley in 'College Admission and the Stability of Marriage'. To explain this part of matching theory we used information from [2, Part I, p. 15-122],[3, Chapter 1, p. 1-36;55-66] and [4].

Next we discuss many-to-one matching in section 2. In this section we use hospitals and interns as the example of many-to-one matching. This is the most common example because it has been used in reality. We used information from [2, Part II, p. 123-186],[3, Chapter 1, p. 37-54] and [4].

These two types of matching theory are the most basic types of matching. There are other types, for example many-to-many matching or one-to-one matching with money as a continuous variable, but they aren't important for understanding the application of matching theory with paired kidney transplantations.

Section 3 is the part of the thesis which discusses the matching process of paired kidney transplantations. This kind of kidney transplantation is relatively new which means there isn't a huge amount of information about it. But it also means there is room for improvement which we'll talk about in the discussion. We used [5], [7] and [8] as the sources of the current matching process of paired kidney transplantations. We also used another article [6] which discusses a potential alternative matching process.

Contents

1 One-to-one matching

1.1 Matching men and women

One-to-one matching is the most basic matching where you match one person or group from a certain set S_1 with another person or group from a certain set S_2. If a group is matched then they are looked at as one entity otherwise it would be many-to-one matching or many-to-many matching.

Definition 1.1.1 A **matching** M is a bijection between the elements of S_1 and S_2

The most common application of one-to-one matching is the marriage model. In this model, there is one set (S_1) which is called 'men' and another set (S_2) which is called 'women'. One-to-one matching isn't only applicable for the groups men and woman. The groups could have been called firms and workers or something else. But the most common one to explain one-to-one matching is the one with men and women.

In this model the groups 'men' and 'women' are of equal size. Each person has a preference list which ranks all the members of the other sex. The men and women are matched together based on these preference list. These matchings can either be stable or unstable.

Definition 1.1.2 A matching M is **stable** if all matches in the matching are stable. A match (w_1, m_1) is stable when either w_1 (m_1) ranks nobody higher than m_1 (w_1) or if w_1 (m_1) ranks someone higher (for example m_2 (w_2)) than their current match then m_2 (w_2) should prefer their current match to w_1 (m_1).

If m_2 preferred w_1 to their current match then m_2 and w_1 should match together. Below is an example which will make everything clearer.

Example There are two groups of size $n = 5$. The preference lists of the men and women are in table 1 and 2.

An example of a stable matching is $\{(w_1,m_1)$, (w_2,m_3), (w_3,m_2), (w_4,m_5), $(w_5,m_4)\}$. An example of an unstable matching is $\{(w_1,m_2)$, (w_2,m_3), (w_3,m_4), (w_4,m_1), $(w_5,m_5)\}$. This is unstable for multiple reasons. w_1 would rather want m_1 and m_1 would rather want w_1. w_5 would rather want m_4 and m_4 would rather want w_5.

Table 1: Women's Preferences

	1	2	3	4	5
w_1 :	m_1	m_2	m_3	m_4	m_5
w_2 :	m_3	m_1	m_4	m_5	m_2
w_3 :	m_2	m_3	m_5	m_1	m_4
w_4 :	m_1	m_4	m_2	m_5	m_3
w_5 :	m_4	m_5	m_1	m_3	m_2

Table 2: Men's Preferences

	1	2	3	4	5
m_1 :	w_1	w_3	w_5	w_4	w_2
m_2 :	w_5	w_1	w_3	w_2	w_4
m_3 :	w_1	w_2	w_4	w_5	w_3
m_4 :	w_5	w_3	w_1	w_2	w_4
m_5 :	w_3	w_5	w_4	w_2	w_1

1.2 Gale-Shapley Algorithm

D. Gale and L.S. Shapley were the first people to find an algorithm which finds stable matchings. Later it was deducted that the same principle was applied in the 19th century for matching hospitals and interns. But in the 19th century they didn't know they were creating stable matches by applying matching theory. More on this in section 2.1.

The algorithm from Gale and Shapley which they found is a man-oriented algorithm.

Definition 1.2.1 The **man-oriented** algorithm is an algorithm which focusses on the preferences of the men and only looks at the preference of the women when there are multiple men who want one woman.

The algorithm goes as follows:

Initial stage:
There is a group of women and a group of men and they are both size n. Each woman is single and at her own house. The men are outside and single.

First stage part 1:
The men stand in front of the houses of the women who are at the top of their preference list.

First stage part 2:
If there are multiple men at one woman's door, she gets engaged to the one highest on her preference list and she dismisses the rest. If there is only one

man at her door, she gets engaged to him. If there is no one at her door, she stays single.
Second stage part 1:
The men who are dismissed go to their next preferred woman on their list.
Second stage part 2:
If there are multiple men at one woman's door, she chooses the one highest on her preference list. If this isn't the man she was already engaged to then she'll terminate the old engagement and gets engaged to the one who is higher on her list. She dismisses the rest. If there is only one man at her door, she get engaged to him. If there is no one at her door, she stays single.
Stage three and onward:
Stage two is repeated until there is one man at each house.

The matching which results from the man-oriented algorithm is stable. To prove this we'll first need to prove theorem 1 and 2.

Theorem 1. *There are a finite number of stages in the man-oriented algorithm.*

Proof. There are n women and n men. A man never goes to the same house because when a man is dismissed he goes to the next woman on his preference list. This means a woman can dismiss $n-1$ men. So every woman can dismiss at most $n-1$ men which mean there can be as many as $n(n-1) = n^2 - n$ stages which dismisses men. This means that the maximum stage where there isn't any dismisses is at stage $n^2 - n + 1$. This is a finite number of stages. □

We can now use theorem 1 to prove the following theorem.

Theorem 2. *The man-oriented algorithm can't end with a man being dismissed by all the women.*

Proof. To prove this it should be known that once a woman is engaged she will always be engaged because she can only get a better partner. It's isn't possible to choose to be single over being engaged because the women think that being engaged is always better than being single. So if a man (m_1) is dismissed, it is because the woman has found someone better. So if m_1 is dismissed by every woman that means every woman has someone else. If every woman has someone else and if there are n woman, that means that there are n other men besides m_1. This is impossible because there is a total of n men which means a man can't be dismissed by all women. The rule of equal sizes of groups is important for this. So no man is dismissed by every woman and there aren't multiple men at one woman's house which means that there is one man at each house. □

Now we can finally prove that the resulted matching is stable.

Theorem 3. *The matching which results from the man-oriented algorithm is stable.*

Proof. We can prove this with a contradiction. Suppose the algorithm gives the following unstable matching $\{(w_1,m_1), (w_2,m_2), (w_3,m_3)\}$. It is unstable because w_1 prefers m_2 over m_1 and m_2 prefers w_1 over w_2. If m_2 prefers w_1 over w_2 then m_2 would have been at the house of w_1 before he was at the house of w_2. If w_1 prefers m_2 over m_1, she would have dismissed m_1 instead of m_2. This is not what happened which means that you can't find a man who prefers another woman who prefers him. So in conclusion the Gale-Shapley algorithm will always end (theorem 1) and when it ends it finds a stable matching. □

Example There are two groups of size $n = 4$. The preference lists of the men and women are in table 3 and 4.

Table 3: Women's Preferences

	1	2	3	4
w_1 :	m_4	m_3	m_2	m_1
w_2 :	m_2	m_1	m_4	m_3
w_3 :	m_2	m_4	m_1	m_3
w_4 :	m_4	m_2	m_3	m_1

Table 4: Men's Preferences

	1	2	3	4
m_1 :	w_3	w_1	w_2	w_4
m_2 :	w_1	w_4	w_3	w_2
m_3 :	w_1	w_2	w_4	w_3
m_4 :	w_3	w_4	w_1	w_2

The matching by the Gale-Shapley algorithm is described in table 5.

Table 5: Example Gale-Shapley Algorithm

Stage	w_1	w_2	w_3	w_4	Dismissed/Single
0					m_1 m_2 m_3 m_4
1 part 1	m_2 m_3		m_1 m_4		
1 part 2	m_3		m_4		m_1 m_2
2 part 1	m_1 m_3		m_4	m_2	
2 part 2	m_3		m_4	m_2	m_1
3 part 1	m_3	m_1	m_4	m_2	

The matching which results from the Gale-Shapley algorithm is $\{(w_1,m_3)$, (w_2,m_1), (w_3,m_4), $(w_4,m_2)\}$. This is a stable matching.

The original Gale-Shapley algorithm resulted in a man-oriented stable matching. You could also reverse the roles and find a woman-oriented matching.

Definition 1.2.2 The **woman-oriented** algorithm is a algorithm which focussed on the preferences of the women and only looks at the preference of the women when there are multiple women who want one man.

In the woman-oriented algorithm the men get approached by the women. This matching would also be stable because it works the same as the man-oriented matching which finds a stable matching. They will both find stable matchings but these matching aren't necessarily the same. This can be shown with the same example as in table 5 but with the roles reversed. The match-

Table 6: Example Woman-Oriented Algorithm

Stage	m_1	m_2	m_3	m_4	Dismissed/Single
0					w_1 w_2 w_3 w_4
1 part 1		w_2 w_3		w_1 w_4	
1 part 2		w_3		w_4	w_1 w_2
2 part 1	w_2	w_3	w_1	w_4	

ing which follows from the woman-oriented algorithm is $\{(w_1,m_3)$, (w_2,m_1), (w_3,m_2), $(w_4,m_4)\}$. If we compare this to the man-oriented algorithm, we see that m_2 and m_4 have switched positions. So they have resulted in different matchings but the matchings are both stable. We also see that some people might prefer one matching over the other matching. We can look at which position on the preference list the match of someone is. It would be in the best interest for the women to do the woman-oriented algorithm. w_3 and w_4 both end up with their first choice. While with the man-oriented algorithm w_3 and w_4 would have both ended up with their second choice. The men prefer the man-oriented algorithm. m_2 end up with his second choice and m_4 end up with his first choice. While with the woman-oriented algorithm m_2 ends up with his third choice and m_4 ends up with his second choice. This leads to theorem 4. We use the following definition in the theorem.

Definition 1.2.3 For two stable matchings A and B, $A \succeq^m B$ ($A \succeq^w B$) if all men (women) prefer matching A to matching B or they are indifferent between A and B.

Theorem 4. *The man-oriented matching M_m is the best possible stable matching for the men. So $M_m \succeq^m M'$ for $M' \in M \backslash \{M_m\}$ where M is a set containing every possible stable matching.*

Proof. Suppose, in matching M_m, m_i is matched with w_i. Now suppose there is another stable matching M_x where m_i is matched with w_j and m_i prefers w_j to w_i. This means that during the man-oriented algorithm w_j rejected m_i in favour of someone else (named m_j) because m_i started at the top of his preference list. m_j also prefers w_j because he was at her house when w_j rejected m_i. They both prefer each other compared to the matching they got in M_x. So M_x isn't a stable matching which means that M_m is the best outcome in a stable matching for every man. This means that $M_m \succeq^m M'$ for every possible stable matching M' other then M_m. $\square$

The same can be said for the women in the woman-oriented algorithm because it is just reversing the roles. Now that we know what the best outcome for the men/women is, we can also say something about the worst outcome.

Theorem 5. *The man-oriented matching M_m is the worst outcome in a stable matching for the women. So $M' \succeq^w M_m$ for $M' \in M \backslash \{M_m\}$ where M is a set containing every possible stable matching.*

Proof. w_k is matched with m_k in the man-oriented matching M_m. Suppose there is a stable matching M_y where w_k is matched with m_l and w_k prefers m_l less than m_k. The matching M_y can only be stable if m_k prefers his matching in M_y which he can't because all men prefer the matching in the man-oriented algorithm. So m_k prefers w_k and w_k prefers m_k which means M_y can't exist. So M_m is the worst possible stable matching for all women. This means that $M' \succeq^w M_m$ for every possible stable matching M' other then M_m. $\square$

We can again reverse the roles and say that the woman-oriented algorithm is the worst outcome in a stable matching for the men.
The man-oriented stable matchings and the woman-oriented stable matchings can also be seen as extremes. They are the ones that one of the groups likes the most and the other dislikes the most. There are also other stable matching which aren't as extreme.

Example There are two groups of size $n = 3$. The preference lists of the men and women are in table 7 and 8.

Table 7: Women's Preferences

	1	2	3
w_1 :	m_2	m_3	m_1
w_2 :	m_3	m_1	m_2
w_3 :	m_1	m_2	m_3

Table 8: Men's Preferences

	1	2	3
m_1 :	w_1	w_2	w_3
m_2 :	w_2	w_3	w_1
m_3 :	w_3	w_1	w_2

The man-oriented matching is given by $\{(w_1,m_1), (w_2,m_2), (w_3,m_3)\}$. The woman-oriented matching is given by $\{(w_1,m_2), (w_2,m_3), (w_3,m_1)\}$. There is another stable matching which is given by $\{(w_1,m_3), (w_2,m_1), (w_3,m_2)\}$. This is a stable matching which lies between the two extreme stable matchings. All the stable matchings form a lattice structure.

$$\begin{tikzcd}
1,2,3 \arrow[d, dash] \\
2,3,1 \arrow[d, dash] \\
3,2,1
\end{tikzcd}$$

This is a small structure because this model doesn't have many stable matchings. The top one is the man-oriented matching and the bottom one is the woman-oriented matching. The numbers are the numbers of the women where m_1, m_2, and m_3 are matched with respectively. The structure gets more complex when there are more stable matchings.

Example [2, chapter 1, p. 22] There is a marriage problem with $n = 4$ and preference lists like table 9 and 10. This model has 10 stable matchings. It

Table 9: Women's Preferences

	1	2	3	4
w_1 :	m_4	m_3	m_2	m_1
w_2 :	m_3	m_4	m_1	m_2
w_3 :	m_2	m_1	m_4	m_3
w_4 :	m_1	m_2	m_3	m_4

Table 10: Men's Preferences

	1	2	3	4
m_1 :	w_1	w_2	w_3	w_4
m_2 :	w_2	w_1	w_4	w_3
m_3 :	w_3	w_4	w_1	w_2
m_4 :	w_4	w_3	w_2	w_1

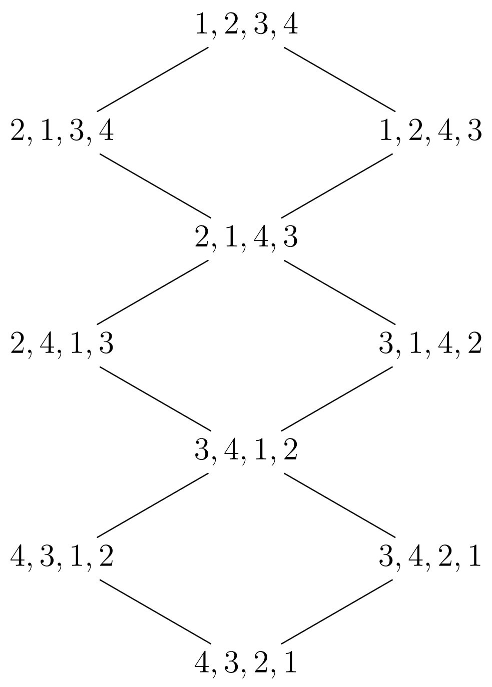
has the man-oriented matching, the woman-oriented matching and 8 other matchings in between. Those matchings make the lattice structure which is displayed in figure 1. The top matching is the man-oriented matching. Below that are two matchings which matches 2 men with their first choice and 2 men with their second choice. After that is a matching which matches all the men with their second choice. This goes on until the bottom one which is the woman-oriented matching which matches the men with their least preferred women and the women with their most preferred men. So when you're a man you'd rather want a matching at the top of the structure because a higher matching is either as well as a lower matching or it is better. When you're a woman you'd want the opposite.

1.3 Extended Gale-Shapley Algorithm

There is also an extended Gale-Shapley algorithm. It is called the extended algorithm because it doesn't just find one stable matching, it finds a list with all possible stable matches. The algorithm gives more information than the regular Gale-Shapley algorithm. It shortens the preference lists by taking out pairs which could never exist in a stable matching.

Again there is a man-oriented extended Gale-Shapley algorithm and a woman-oriented extended Gale-Shapley algorithm. These algorithms result in a man-oriented list and a woman-oriented list respectively. The overlap

Figure 1: Lattice Structure



between these two lists is the Gale-Shapley list which contains all the stable matches.

What the extended Gale-Shapley algorithm essentially does is it deletes all the women (men) on the men's (women's) preference list that are lower than the woman-oriented (man-oriented) matching because this was the worst possible outcome in a stable matching (theorem 5). It also deletes all the women (men) who are better than the men-oriented (women-oriented) matching because this was the best possible outcome in a stable matching (theorem 4).

The man-oriented algorithm gives the men their most preferred women from the Gale-Shapley list and the women their least preferred men. The women-oriented algorithm does the opposite. Below is the extended Gale-Shapley algorithm with i is a man and j is a woman in the man-oriented version and i is a woman and j is a man in the woman-oriented version.

Initial stage:
There is a group of women and a group of men and they are both size n. Everyone is not matched.

Stage 1:
i is the first person on $j's$ list. If i is already engaged (for example to j_1) then stop the engagement and engage i to j. If there is anyone worse (for example j_2) than j on i's list, delete i from j_2's list and delete j_2 from i's list.

Stage 2:
Repeat the first stage until every i is matched.

This can be explain better with an example.

Example There are 4 women and 4 men with preference lists displayed in table 11. The woman-oriented extended Gale-Shapley algorithm is applied

Table 11: Women's Preferences (left) and Men's Preferences (right)

	1	2	3	4		1	2	3	4
w_1 :	m_4	m_3	m_2	m_1	m_1 :	w_1	w_2	w_3	w_4
w_2 :	m_1	m_4	m_3	m_2	m_2 :	w_3	w_1	w_4	w_2
w_3 :	m_4	m_1	m_2	m_3	m_3 :	w_1	w_4	w_2	w_3
w_4 :	m_1	m_2	m_4	m_3	m_4 :	w_3	w_4	w_1	w_2

in table 12.

The new woman-oriented preference list of the men and the women is displayed in table 13. Some impossible stable matchings were removed. The man-oriented extended algorithm is the same as the woman-oriented algorithm with the roles reversed. The man-oriented preference lists are displayed

Table 12: Example extended Gale-Shapley algorithm

Stage	m_1	m_2	m_3	m_4	Dismissed/Single
0					w_1 w_2 w_3 w_4
1	w_2 w_4			w_1 w_3	
1	w_2			w_3	w_1 w_4

Delete pairs (m_1, w') with $w' \prec w_2$ and pairs (m_4, w'') with $w'' \prec w_3$.

Stage	m_1	m_2	m_3	m_4	Dismissed/Single
1	w_2	w_4	w_1	w_3	

Delete pairs (m_2, w') with $w' \prec w_4$ and pairs (m_3, w'') with $w'' \prec w_1$.

Table 13: Women's Preferences (left) and Men's Preferences (right)

	1	2	3	4		1	2	3	4
w_1 :	m_3	m_2	m_1		m_1 :	w_1	w_2		
w_2 :	m_1				m_2 :	w_3	w_1	w_4	
w_3 :	m_4	m_2			m_3 :	w_1	m_4		
w_4 :	m_2	m_3			m_4 :	w_3			

in table 14. The Gale-Shapley list can be given by the overlap between table

Table 14: Women's Preferences (left) and Men's Preferences (right)

	1	2	3	4		1	2	3	4
w_1 :	m_4	m_3			m_1 :	w_2	w_4		
w_2 :	m_1				m_2 :	w_4			
w_3 :	m_4				m_3 :	w_1			
w_4 :	m_1	m_2			m_4 :	w_3	w_1		

13 and table 14. This is displayed in table 15.

Everyone has only one person on their Gale-Shapley preference list. This is because there is only one stable matching. The man-oriented algorithm and the woman-oriented algorithm result in the same stable matching namely $\{(w_1,m_3), (w_2,m_1), (w_3,m_4), (w_4,m_3)\}$.

1.4 Number of stable matchings

In section 1.2 we saw that there could be multiply stable matchings in a certain marriage problem. The question which could be asked now is: what is the maximum number of stable matchings for a certain instance n? This instance n is a marriage problem with n men and n women. This is a problem which still hasn't been solved. But it is possible to give a lower bound

Table 15: The Gale-Shapley Preference Lists

	1	2	3	4		1	2	3	4
w_1 :	m_3				m_1 :	w_2			
w_2 :	m_1				m_2 :	w_4			
w_3 :	m_4				m_3 :	w_1			
w_4 :	m_2				m_4 :	w_3			

to this maximum. When n rises the number of stable matching increases exponentially. If you want to find all the stable matchings by hand it might take a while. There are different ways to calculate the lower bound depending on what n is. These ways are discussed in theorem 6, 7 and 8.

Theorem 6. *If you have two stable marriage problems, one of size n and one of size m and with $g(n)$ and $g(m)$ stable matchings respectively then there is also a stable marriage problem of size nm with at least $max(g(n)g(m)^n$, $g(m)g(n)^m)$ stable matchings.*

Proof. Suppose there are two instances, one of size n and one of size m. The men in instance n are labelled $p_1, ..., p_n$ and the women are labelled $t_1, ..., t_n$. The men in instance m are labelled $q_1, ..., q_m$ and the women are labelled $s_1, ..., s_m$. There is an instance nm with men labelled (p_i, q_j) and with women labelled (t_i, s_j) with $i = 1, ..., n$ and $j = 1, ..., m$.

A man (p_i, q_j) from instance nm prefers woman (t_k, s_l) to $(t_{k'}, s_{l'})$ if q_j prefers s_l to $s_{l'}$ or if $l = l'$ and p_i prefers t_k to $t_{k'}$. This is the same for the women but with the roles reversed. So a woman (t_i, s_j) from instance nm prefers man (p_k, q_l) to $(p_{k'}, q_{l'})$ if s_j prefers q_l to $q_{l'}$ or if $l = l'$ and t_i prefers p_k to $p_{k'}$.

Now we call M equal to any stable matching in the marriage problem of size m and $M_1, ..., M_m$ is a sequence of stable matchings of the marriage problem of size n. M can be any matching of $g(m)$ and the sequence $M_1, ..., M_m$ can be $g(n)$ to the power of m. So the combination of M and $M_1, ..., M_m$ is $g(m) * g(n)^m$.

Suppose we've got the following matching $((p_i, q_j), (M(p_i), M_i(q_j)))$ where M and M_i are matchings. Now we need to prove that this is a stable matching. We can prove this by a contradiction. Suppose it isn't stable. This means that there is another matching $((p, q), (t, s))$ which is better. If this is better, one of the following preferences should be correct. Either (1) q prefers s to $M_i(q)$ or (2) $s = M_i(q)$ and p prefers t to $M(p)$. With one of these two, one of the following preferences should also be true. Either (3) s prefers q to $M_i(s)$ or (4) $q = M_i(s)$ and t prefers p to $M(t)$. 1 and 3 can't both be true because M_i is stable. 2 and 4 can't both be true because

M is stable. The other combinations aren't true because it impossible for those combinations to happen together. This means that that the matching $((p_i, q_j), (M(p_i), M_i(q_j))$ is a stable matching. This means the stable marriage problem has at least $g(m)g(n)^m$ stable matchings. This can also be done with the roles of n and m reversed. This will result in at least $g(n)g(m)^n$ stable matchings. This means that the lower bound of the maximum number of stable matchings is $max(g(m)g(n)^m, g(n)g(m)^n)$. $\square$

Theorem 7. *If there is a marriage problem with n men, n women and $g(n)$ stable matchings then there also is a marriage problem with $2n$ men, $2n$ women and at least $2[g(n]^2$ stable matchings.*

Proof. Suppose there is an instance of size n. The men are labelled as m_i and the women as w_i with $i = 1, ..., n$. So we have $M = \{m_1, ..., m_n\}$ and $W = \{w_1, ..., w_n\}$. Create a instance of size n with the men labelled m_j and the women labelled w_j with $j = n+1, ..., 2n$. So we have $M' = \{m_{n+1}, ..., m_{2n}\}$ and $W' = \{w_{n+1}, ..., w_{2n}\}$. Create this instant so that if m_i prefers w_k to w_l for $i \leq n$ then m_{i+n} prefers w_{k+n} to w_{l+n} and if w_i prefers m_k to m_l for $i \leq n$ then w_{i+n} prefers m_{k+n} to m_{l+n}. This way both instances will have the same number of stable matchings.

Now we need to create another instance but of size $2n$ with the men labelled m_a and the women labelled w_a with $a = 1, ..., 2n$. This way we'll have $M'' = \{m_1, ..., m_{2n}\}$ and $W'' = \{w_1, ..., w_{2n}\}$. The preference list of the last n people can be determine by the first n people and the preference list of the first n people can be determine by the last n people for either the men or the women.

Let's say z is a stable matching for the instance with M men and W women and z' is a stable matching for the instance with M' men and W' women. Now also have a matching z''_1 for the instance with M'' men and W'' women. This matching $z''_1(m)$ has the following definition.

$$z''_1(m) = z(m) \qquad \text{if m is from M}$$

and

$$z''_1(m) = z'(m) \qquad \text{if m is from M'}$$

This means that z''_1 is a stable matching because both z and z' are stable matchings. There can also be a stable matching z''_2 for the instance with M'' men and W'' women which has the following definition.

$$z''_2(w_i) = z'(w_{i+n}) \qquad \text{if } \ i \leq n$$

and

$$z''_2(w_i) = z(w_{i-n}) \qquad \text{if } \ i \geq n.$$

The matchings z and z' can each be $g(n)$ stable matchings (because both instances are of size n) which means the instance of size $2n$ can have at least $2g(n)^2$ stable matchings. $\square$

Theorem 8.

The maximum number of stable matchings is at least 2^{n-1} when $n = 2^k$ with $k \geq 0$.

Proof. This can be done by induction. The theorem is correct for $k = 0$. If $k = 0$ then $n = 2^0 = 1$ and $2^{n-1} = 2^0 = 1$. This means if there is only one man and one woman then there is only one stable matching which is obvious. Now we say that the theorem is correct for $n = 2^i$. We need to prove that it is correct for $n = 2^{i+1}$ with $2^{n-1} = 2^{2^{i+1}-1}$ stable matchings. To prove this we'll need what we proved in theorem 6. We have $n = 2^i$ and $g(n) = 2^{2^i-1}$ and we'll use $m = 2$ with preference lists in table 16 and 17. This instance has

Table 16: Women's Preferences

	1	2
w_1 :	m_2	m_1
w_2 :	m_1	m_2

Table 17: Men's Preferences

	1	2
m_1 :	w_1	w_2
m_2 :	w_2	w_1

two stable matchings $\{(w_1, m_2), (w_2, m_1)\}$ and $\{(w_1, m_1), (w_2, m_2)\}$. This means that $g(m) = 2$. We have proven in theorem 6 that there should also be another stable marriage problem with size $nm = 2^i * 2 = 2^{i+1}$ and with at least $max(g(n)g(m)^n, g(m)g(n)^m) = max(2^{2^i-1} * 2^{2^i}, 2 * (2^{2^i-1})^2) = max(2^{2^{i+1}-1}, 2^{2^{i+1}-1}) = 2^{2^{i+1}-1}$ stable matchings. So there exists a stable marriage problem with size $2^i * 2 = 2^{i+1}$ and with at least $2^{2^{i+1}-1}$ stable matchings. $\square$

In table 18 are the lower bounds from theorem 8 for a few examples of n. Here we can see that the lower bound grows exponentially when n rises. The actual maximum can be much higher than the values shown in the tables because they are lower bounds.

Table 18: Theorem 8

i	$n = 2^i$	2^{n-1}
0	1	1
1	2	2
2	4	8
3	8	128
4	16	32768
5	32	2147483648

1.5 Groups of unequal sizes and other variations

In this section we will discuss different variations of the marriage model from section 1.1. It isn't always the case that you're matching people from groups of equal sizes. It could be possible that there are more men than women or the other way around. Some people won't get matched and will stay single. We will still assume that people would rather be married than to be single. The Gale-Shapley algorithm from section 1.2 can still be applied here but it is slightly altered. The definition of a stable matching is also slightly different.

Definition 1.5.1 A match (w_x, m_x) is **stable** if for each m (w) that w_x (m_x) prefers to their current match, m (w) prefers their own match to w_x (m_x). Also if someone is single that means nobody prefers them to their current partner.

So if there are more women than men, the Gale-Shapley algorithm (which is the man-oriented algorithm) will stop when all men are engaged to different women. The difference between the number of men and the number of women is the number of single women. This matching is still stable and it is the best matching for the men and the worst matching for the women. The women-oriented algorithm will also stop when all the men are engaged. There will still be single women but the stable matching is the best outcome in a stable matching for the women. If a woman is single after the woman-oriented algorithm then she is single in every stable matching. When there are more men than women then the algorithms will stop when all the women are engaged and there will be single men left.

Example Suppose there is a marriage problem with 3 women and 5 men. Their preference lists are shown in table 19 and 20. This marriage problem has only one stable matching namely $\{(w_1, m_5), (w_2, m_1), (w_3, m_2)\}$. This is the only stable matching because the men-oriented and the women-oriented

Table 19: Women's Preferences

	1	2	3	4	5
w_1 :	m_5	m_4	m_1	m_2	m_3
w_2 :	m_1	m_5	m_3	m_4	m_2
w_3 :	m_5	m_2	m_4	m_1	m_3

Table 20: Men's Preferences

	1	2	3
m_1 :	w_1	w_2	w_3
m_2 :	w_2	w_1	w_3
m_3 :	w_3	w_2	w_1
m_4 :	w_1	w_3	w_2
m_5 :	w_2	w_1	w_3

algorithm result in the same matching. The men m_3 and m_4 are single.

Another variation of the stable marriage problem is when people would rather be single than to be married to certain people. This makes the problem a bit more realistic because you might not like everyone who is available. This scenario might end up having both a group of single men and a group of single women no matter how many men and women there are. This problem again has a slightly different definition of a stable matching.

Definition 1.5.2 A match (w_x, m_x) is **stable** if for each m (w) that w_x (m_x) prefers to their current match, m (w) prefers their own match to w_x (m_x) or m (w) prefers being single. If someone is single then everyone they prefer to being single likes their current partner more or likes being single more.

The man-oriented algorithm and the women-oriented algorithm change because the preference lists of the men and women change. The preference lists get an extra element named x-single (with x is the name of the person who is single). When w is above the element m-single on the preference list of m that means m prefers w to being single. When w is below the element m-single on the preference list of m that means m prefers to be single to being with w.

The algorithm will stay the same but for it to work a fake person should be added for each person that can be single. This will become clearer with the help of the following example.

Example There is a marriage problem with 2 women and 3 men. They have preference lists which are displayed in table 21 and 22. The Gale-

Table 21: Women's Preferences

	1	2	3	4	5
w_1 :	m_1	m_2	w_1-single	m_3	w_2-single
w_2 :	m_2	m_3	w_2-single	m_1	w_1-single
m_1-single:	m_1	m_2	m_3	w_1-single	w_2-single
m_2-single:	m_2	m_1	m_3	w_1-single	w_2-single
m_3-single:	m_3	m_1	m_2	w_1-single	w_2-single

Table 22: Men's Preferences

	1	2	3	4	5
m_1 :	w_2	w_1	m_1-single	m_2-single	m_3-single
m_2 :	w_1	m_2-single	w_2	m_1-single	m_3-single
m_3 :	w_1	w_2	m_3-single	m_1-single	m_2-single
w_1-single:	w_1	w_2	m_1-single	m_2-single	m_3-single
w_2-single:	w_2	w_1	m_1-single	m_2-single	m_3-single

Shapley algorithm is applied in table 23. The resulted stable matching is $\{(w_1, m_1), (w_2, m_3), (m_2\text{-single}, m_2)\}$. This means that m_2 stays single. He could have had w_2 but he liked staying single more than getting engaged to w_2. This result is already clear after step 3 part 1 of the Gale-Shapley algorithm but the algorithm goes until everyone is matched with someone else. This doesn't change the outcome.

It is also impossible for someone to be in a stable matching with the single version of someone else. This is because i-single always has i at the top of its preference list and i prefers i-single to every other x-single. The order of the preference list from x-single doesn't matter as long as x is on the top of the list.

The last variation of the marriage model that we will discuss is the case of indifference. Someone is indifferent when they don't prefer one over the other. So m is indifferent between w and w' if he doesn't want w more (or less) than w'. It can also be that someone is indifferent between someone else and being single.

With indifference there are different types of stability depending on how strict you want to be. The first type of stable matching is super-stable matchings. The matching is unstable if you like someone else at least as

Table 23: Example Gale-Shapley algorithm with single people

Stage	w_1	w_2	m_1-s	m_2-s	m_3-s	Dismissed
0						m_1 m_2 m_3 w_1-s w_2-s
1 part 1	m_2 m_3 w_1-s	m_1 w_2-s				
1 part 2	m_2	w_2-s				m_1 m_3 w_1-s
2 part 1	m_1 m_2	m_3 w_1-s w_2-s				
2 part 2	m_1	m_3				m_2 w_1-s w_2-s
3 part 1	m_1 w_2-s	m_3 w_1-s		m_2		
3 part 2	m_1	m_3		m_2		w_1-s w_2-s
4 part 1	m_1	m_3	w_1-s w_2-s	m_2		
4 part 2	m_1	m_3	w_1-s	m_2		w_2-s
5 part 1	m_1	m_3	w_1-s	m_2	w_2-s	

much as your current partner and if they like you at least as much as their current partner. When you have this type of stability it is possible to have no stable matchings. An example of this is when everyone is indifferent to each other.

The second type of stability is one called strongly-stable matchings. The matchings are unstable if you like someone more than your current partner and they like you at least as much as their current partner. It is again possible with this type of stability to have no stable matchings.

The third kind of stability is a weaker kind of stability which is easier to achieve. The matchings are unstable if you like someone strictly more than your current partner and they like you strictly more than their current partner.

Stable matchings with indifference can be found by choosing randomly if one is better than the other when someone is indifferent and then applying a certain algorithm. Each time you choose someone else in a situation with indifference it might result in a different stable matching.

1.6 Strategy and other problems

A question which could arise from one-to-one matching is: are there any strategies which you could apply to improve your outcome? Is there anything you could do which results in a more preferred partner? The answer to this is "yes" depending on your situation. It all depends on if you're a man or a woman in the man-oriented or the woman-oriented situation. If you're a woman in the man-oriented situation or if you're a man in the woman-oriented situation you can get a better partner if you apply a strategy. This strategy is lying about your preference list. This way you might have to dismiss someone for someone else who you like less. But when you do this you might end up with someone better. It is not possible for a man in a man-oriented situation or for a woman in a woman-oriented situation to do this. For example if they were single without a strategy then they can't apply a strategy to suddenly get married with someone that they like. There is an example below which makes it a bit clearer.

Example There are 4 men and 3 women. The preference list of the women is described in table 24 and that of the men is in table 25. Some men and some women prefer being single to being with some people. This is explained in section 1.5. A stable matching is $\{(w_1, m_2), (w_2, m_1), (w_3, w_3\text{-single}),$

Table 24: Women's Preferences

	1	2	3	4	5	6	7
w_1 :	m_1	m_2	m_3	m_4	w_1-s	w_2-s	w_3-s
w_2 :	m_2	m_1	m_3	m_4	w_2-s	w_1-s	w_3-s
w_3 :	m_3	m_1	w_3-s	m_4	m_2	w_1-s	w_2-s
m_1-s:	m_1	m_2	m_3	m_4	w_1-s	w_2-s	w_3-s
m_2-s:	m_2	m_1	m_3	m_4	w_1-s	w_2-s	w_3-s
m_3-s:	m_3	m_1	m_2	m_4	w_1-s	w_2-s	w_3-s
m_4-s:	m_4	m_1	m_2	m_3	w_1-s	w_2-s	w_3-s

Table 25: Men's Preferences

	1	2	3	4	5	6	7
m_1 :	w_2	w_1	w_3	m_1-s	m_2-s	m_3-s	m_4-s
m_2 :	w_3	w_1	w_2	m_2-s	m_1-s	m_3-s	m_4-s
m_3 :	w_2	m_3-s	w_1	w_3	m_1-s	m_2-s	m_4-s
m_4 :	w_1	w_2	w_3	m_4-s	m_1-s	m_2-s	m_3-s
w_1-s:	w_1	w_2	w_3	m_1-s	m_2-s	m_3-s	m_4-s
w_2-s:	w_2	w_1	w_3	m_1-s	m_2-s	m_3-s	m_4-s
w_3-s:	w_3	w_1	w_2	m_1-s	m_2-s	m_3-s	m_4-s

$(m_3\text{-single}, m_3), (m_4\text{-single}, m_4)\}$ when we apply the Gale-Shapley algorithm. Both w_1 and w_2 are matched with their second choice partners and m_1 and m_2 are matched with their first and second choice respectively. w_3, m_3 and m_4 are all single.

w_2 could get a better partner if she acts like she has a different preference list. A preference list she can use is ($m_2(1)$, $m_3(2)$, $m_4(3)$, $m_1(4)$, w_2-s(5),

w_1-s(6), w_3-s(7)). So she changes it by moving m_1 to a lower spot on her list. This will cause w_1 to get m_1 and to dismiss m_2. m_2 will then move to w_2. This is a way for w_2 to get her first choice. The resulted matching is $\{(w_1, m_1), (w_2, m_2), (w_3, w_3\text{-single}), (m_3\text{-single}, m_3), (m_4\text{-single}, m_4)\}$. This matching is still a stable matching even with the original preference list from w_2. Nobody can get a better partner than their current partner because the people they like more than their current partner don't like them more. This is because the Gale-Shapley algorithm only stops when a stable matching is made.

The question we are left with is: is there a way to avoid this strategy?

Theorem 9. *There is no algorithm which avoids the strategy to lie.*

Proof. Suppose there is a marriage problem with 2 men and 2 women. We can show that for every stable matching process a man or a woman can do better by lying about their preference list. The women and the men have the preferences lists displayed in table 26 and table 27 respectively.

Table 26: Women's Preferences

	1	2
w_1 :	m_2	m_1
w_2 :	m_1	m_2

Table 27: Men's Preferences

	1	2
m_1 :	w_1	w_2
m_2 :	w_2	w_1

This marriage problem has two stable matchings. The stable matchings are $M_1 = \{(w_1, m_1), (w_2, m_2)\}$ and $M_2 = \{(w_1, m_2), (w_2, m_1)\}$. This means that each algorithm which produces a stable matching must result in one of these matchings.

Suppose an algorithm results in M_1. This would be bad for w_1 and w_2. Now what if w_2 changed her preference list and says she would rather be single than be with m_2. The only possible stable matching which could result from this changed preference is M_2. The same could be said for m_2 instead of w_2 if an algorithm choose M_2. We can also assume without loss of generality that this is true for a larger instance n. In conclusion there is not an algorithm where it is the best for everyone to be honest. □

Another problem which can arise with one-to-one matching is when the people that get matched together are in the same group. For example when you want to match 2 people together to work on a project. If this is the case then there is one group (with men and/or women) and each person has a preference list which contains all other people from the group. This model can cause problems because it isn't certain that there is always a stable matching. This will be shown in the next example.

Example There are 4 people (it doesn't matter if they're men or women) who will be matched together. Their preference list is displayed in table 28. There are no stable matchings. There are 3 possible matchings. The first one is $\{(p_1, p_2), (p_3, p_4)\}$. This matching isn't stable because p_2 prefers p_4 to p_1 and p_4 prefers p_2 to p_3. The next matching is $\{(p_1, p_3), (p_2, p_4)\}$. This matching isn't stable because p_3 prefers p_2 to p_1 and p_2 prefers p_3 to p_4. The last matching is $\{(p_1, p_4), (p_2, p_3)\}$. This isn't a stable matching because p_4 prefers p_3 to p_1 and p_3 prefers p_4 to p_2.

Table 28: People's Preferences

	1	2	3
p_1 :	p_2	p_3	p_4
p_2 :	p_3	p_4	p_1
p_3 :	p_4	p_2	p_1
p_4 :	p_2	p_3	p_1

2 Many-to-one matching

2.1 Hospitals and interns

The algorithm that Gale and Shapley invented was already being applied in some way before they thought of it. It was mostly being applied to match interns and hospital programs. This wasn't exactly the same as the Gale and Shapley method because this is many-to-one matching instead of one-to-one matching.

There are still two sets, one finite set of interns (I) and one finite set of hospital programs (H), which are matched together. Now it's the case that many intern can be matched to one hospital program. Each intern has a preference over the hospital programs and each hospital program has a preference over the interns. It is also possible that an intern would rather wait a year (or do something else) than to join some program. A hospital can also keep some intern positions empty instead of hiring someone they don't like. This resembles wanting to be single in the marriage problem from section 1.5.

Definition 2.1.1 A matching is **stable** when item 1 is true combined with either item 2, 3 or 4.

1. Intern i is not matched with hospital program h
2. If intern i prefers h to their current situation then h either has all their spots filled with more acceptable interns or would rather have an empty spot than have intern i.
3. If hospital program h prefers i to their current situation then i either is matched with a more acceptable hospital program or would rather be with no hospital program than to be with h.
4. Intern i does not prefer h to their current situation and h does not prefer i to their current situation.

Just like the marriage problem there are different algorithms to find stable matchings. Again there are two extreme matchings. There is the hospital-oriented algorithm and the intern-oriented algorithm. The hospital-oriented algorithm works almost the same as the Gale-Shapley (man-oriented) algorithm with the interns as the women and the hospitals as the men. The interns reject every hospital program except the one they prefer the most. The intern-oriented algorithm works almost the same as the woman-oriented algorithm. The Hospital programs starts to reject people when the amount of applications surpass the amount of places. There is also another algorithm which is called the NIMP algorithm. This is an algorithm which was actually used to match interns and hospital programs.

2.1.1 The NIMP Algorithm

First we'll discuss the NIMP algorithm. This is an algorithm which was applied in 1951 before the existence of the Gale-Shapley algorithm. NIMP stands for 'National Intern Matching Program'. This is an algorithm which was actually used to match interns with hospital programs. This algorithm worked as follows:

Initial Phase

The interns and the hospitals sent their preference list to the central clearing-house where their preference list is altered. If an intern i didn't see a certain hospital program h as an acceptable option then i would be removed from

h's preference list. The same would be done to the interns' preference lists if a hospital program didn't think they were an acceptable option. Begin with the matching phase.

Matching Phase

Stage 1

Look at all the hospital programs' first choices. If there is any hospital program that is also the first choice of the intern then assign them together and start with the tentative-assignment-and-update phase. If there is no hospital program which is also the first choice of the intern then go to the next stage.

Stage 2

Look at all the hospital programs' first choices. If there is any hospital program that is the second choice of the intern then assign them together and start with the tentative-assignment-and-update phase. If there is no hospital program which is the second choice of the intern then go to the next stage.

$\vdots$

Stage m

Look at all the hospital programs' first choices. If there is any hospital program that is the mth (with m is the maximum number of hospitals of an intern's preference list) choice of the intern then assign them together and start with the tentative-assignment-and-update phase. If there is no hospital program which is the mth choice of the intern then stop the algorithm. Each intern is matched with the hospital which is at the bottom of their updated preference list.

Tentative-Assignment-and-Update Phase

Look at all the assignment you've just made. If the interns have any hospital programs on their preference list that they prefer less than their assigned hospital program then those programs should be removed from the interns' preference lists. Also remove all the assigned interns from all the other hospital programs' preference lists if the interns list these hospital programs lower than their current hospital program. Restart the matching phase with the updated preference lists when this phase is finished.

Theorem 10. *The NIMP algorithm will produce a stable matching no matter what the preference lists are.*

Proof. When the algorithm stops, every hospital program h_k is matched with c_k (for capacity) interns who are all at the top of the updated preference lists of the hospital programs. The algorithm couldn't have ended if this isn't true because there could have been another match. What if there was a hospital program h_a which preferred intern i_b. This means i_b was removed from h_a preference list because i_b was assigned to a hospital program which they preferred more that h_a. This means that there aren't any interns who are more preferred than their assigned interns who also prefer them more than their assigned hospital program. This means that the matching is stable. □

2.1.2 The Hospital-Oriented Algorithm

The hospital-oriented algorithm is the same as the man-oriented algorithm from section 1.2 but then with many-to-one matching instead of one-to-one matching. The hospital programs represent the men and the interns represent the women. The intern can get conditionally accepted to a hospital program instead of getting engaged. If an intern isn't assigned to any program then they aren't single but they are free and the hospital programs can be undersubscribed.

Before the algorithm is started, the unacceptable matches are deleted from the preference lists. This means if a hospital h would rather have an empty spot than to hire intern i, h should be removed from i's list and i should be removed from h's list. The algorithm goes as follows.
Initial stage:
There is a group of interns and a group of hospital programs. Everyone is not matched. This means that all the interns are free and all the hospital programs are undersubscribed. Begin stage 1 for each hospital program h.
Stage 1:
Intern i is the first person on hospital program h's list. If i is already conditionally assigned (to h_1) then break the bond between i and h_1 and conditionally assign i to h. If there is anyone worse (for example h_2) than h on i's list, delete i from h_2's list and delete h_2 from i's list. Go to stage 2.
Stage 2:
Repeat the first stage until one of the following statement isn't true anymore. (1) A hospital program h is undersubscribed. (2) There exist a intern i on h's preference list who is not conditionally assigned to h.

This algorithm results in a hospital-oriented list where the hospital (with c spots) has assigned the top c interns of their list. If an intern i isn't on hospital program h's list and h isn't on i's list, that can mean multiple thing.

Lemma 11. *(i) (i, h) is an unstable match. (ii) The intern i prefers the hospital programs on their preference list to h. (iii) The matching (i, h) doesn't cause the other matchings on the reduced list to be unstable.*

Proof. (i) Assume the match (i, h) was the first stable match to be removed. The match (i, h) was removed from the list which means i was assigned to someone (for example h_1) who they prefer more than h. The hospital program h_1 has c_{h_1} spots. The number of interns i_1 who are matched with

h_1 and that are more preferred than i should be less than the capacity of h_1 because we assumed (i, h) was the first stable matching to be removed. So the hospital program h_1 either has some place left or it has an intern who they prefer less than i. We also said that i preferred h_1 to h. This means that (i, h) is unstable because (i, h_1) is better. So (i, h) can't be stable.

(ii) The intern i only removes hospital programs from their list when those programs are less preferred than the program that they are conditionally assigned to. This means that h is always less preferred than the hospital programs on the new preference list.

(iii) This can be proven with statement (ii). We know that intern i prefers the hospital program on their new list to h. This means that (i, h) can't cause any matching from the new list to be unstable. That can only happen when i prefers h to their current situation and when h prefers i to their current situation. □

From lemma 11 and because the intern are matched with the last hospital program on their hospital-oriented list we can also conclude that the intern is matched with their worst possible stable matching. This means that, just like the man-oriented algorithm from section 1.2, the hospital gets their best possible stable match and the interns get their worst possible stable match.

2.1.3 The Intern-Oriented Algorithm

The resident-oriented algorithm isn't just the hospital-oriented algorithm with the roles reversed. The hospital programs only start to reject interns when the number of interns that apply exceeds the number of available places. This is because the interns who were not good enough were already removed from the preference list. This means that everyone who applies is acceptable and only when there are too many interns then the least preferred ones will be denied. The intern-oriented algorithm goes as followed.

Initial stage:

There is a group of interns and a group of hospital programs. Everyone is not matched. This means that all the interns are free and all the hospital programs are undersubscribed. Begin stage 1 for each intern i.

Stage 1:

Hospital program h is the first program on intern i's list. If h is undersubscribed, go to stage 2. If h doesn't have any empty places, go to stage 3.

Stage 2:

Conditionally assign i to h. If h doesn't have any empty places left, go to stage 4. If there are no free interns or an intern has an empty list, stop the algorithm. Otherwise begin stage 1 for the next intern.

Stage 3:

Look at the interns who are conditionally assigned to h. Intern i_1 is the least preferred intern who is conditionally assigned to h. Reassign i_1 to be free. Conditionally assign i to h. Go to stage 4.

Stage 4:

Intern i_2 is the worst intern from the interns who are conditionally assigned to h. This can be i or another intern. If there is any intern who is less preferred (for example i_3) than i_2 on h's preference list, remove i_3 from h's preference list and remove h from i_3's preference list. If there are no free interns or an intern has an empty list, stop the algorithm. Otherwise go to stage 1 for the next intern.

Theorem 12. *(i) All the conditionally assignments, which still exist after the algorithm stops, are stable. (ii) The interns are matched with their best possible stable match.*

Proof. (i) This can be proven by a contradiction. Suppose there is an intern i who is conditionally assigned to h at the end of the algorithm. Now suppose that this match is unstable. This would mean that there is another hospital program h_1 which is more preferred than h and h_1 is either undersubscribed or h_1 prefers i to their worst intern. But if this was true, i would have tried to apply to h_1 before applying to h and h_1 would have conditionally accepted i. This means that all the conditionally assignment, which result from the algorithm, are stable.

(ii) The interns go down their preference list when the algorithm is applied. This means that if a intern isn't matched with their first choice, it is because someone else was better. This would also be impossible for any other algorithm because you can't get a better result than to start at the top of the preference list. This means that the interns are matched with their best possible stable matching. □

It can be concluded from this second statement that an intern will always be free if that intern is free at the end of the intern-oriented algorithm. This is because interns can't do better than the result of the intern-oriented algorithm.

It isn't necessarily true that the hospital programs are matched with the worst possible stable matches. There are two different possibilities either the hospital program end up undersubscribed or it doesn't have any available places left. If the hospital program ends up undersubscribed after an algorithm then it will have the same set of interns after every stable algorithm. In this case, one result won't be worst than another result. If the hospital program ends up without any open places after the intern-oriented algorithm then it doesn't necessarily consist of the worst possible stable matchings. The first case is shown in theorem 13 and the second case is shown in an example.

Theorem 13. *(i) The hospital program h has the same interns in stable matching M_2 as in intern-oriented stable matching M_1 if h is undersubscribed in M_2 and (ii) every hospital program has the same number of interns in each stable matching.*

Proof. (i) Suppose M_1 is the stable matching after the intern-oriented algorithm and M_2 is any stable matching. Hospital program h is undersubscribed in M_2. If intern i is matched with h in M_1 but not with h in M_2 that would make M_2 unstable. This is because h is undersubscribed and i prefers h to every other stable matching partner because it is the result of the intern-oriented algorithm. So if h is undersubscribed in M_2 then it should have the same interns as in M_1.

(ii) M_1 is the matching which is the result of the intern-oriented algorithm and M_2 is another stable matching. The number of interns matched with hospital programs in M_2 can't be larger than the number of interns matched with hospital programs in M_1. This is because if a intern is free in M_1 then they should always be free because they can't get a better result than the intern-oriented result. The number of interns matched with hospital programs in M_2 also can't be smaller than the number of interns matched with hospital programs in M_1. We already proved in part (i) that every intern who is matched with h in M_1 should also be matched with h in M_2 when h is undersubscribed in M_2. This means that the number of intern matched with hospital programs in M_2 can't be smaller than that of M_1. So it can't be smaller or larger which means it is the same. □

Statement (i) and (ii) can be combined to conclude that if a hospital program is undersubscribed in any stable matching then it is matched with the same interns in every stable matching. This also means they won't prefer one stable matching over another because the result is the same. Now we'll look at an example which will look at the preference of the hospital program between stable matchings when the hospital program isn't undersubscribed.

Example [2, chapter 1, p. 50] There are 3 hospital programs and 7 interns. The preference lists are displayed in table 29 and 30.

These preference lists can be used to find the intern-oriented matching and the hospital-oriented matching. This can be done with the algorithms from section 2.1.2 and 2.1.3. The matching which results from the intern-oriented algorithm is: $\{(h_1,\{i_2, i_5, i_8, i_{10}\}), (h_2, i_7), (h_3, \{i_1, i_4, i_6\}), (h_4, \{i_3, i_9\}), (h_5, i_{11})\}$ The matching which results from the hospital-oriented algorithm is: $\{(h_1,\{i_4, i_5, i_9, i_{11}\}), (h_2, i_7), (h_3, \{i_3, i_6, i_8\}), (h_4, \{i_1, i_2\}), (h_5, i_{10})\}$. Hospital program 2 always gets intern 7 because h_2 is undersubscribed. This was proven in theorem 13. This model also has other stable

Table 29: Interns' Preferences

	1	2	3	4	5
i_1 :	h_3	h_1	h_5	h_4	
i_2 :	h_1	h_3	h_4	h_2	h_5
i_3 :	h_4	h_5	h_3	h_1	h_2
i_4 :	h_3	h_4	h_1	h_5	
i_5 :	h_1	h_4	h_2		
i_6 :	h_4	h_3	h_2	h_1	h_5
i_7 :	h_2	h_5	h_1	h_3	
i_8 :	h_1	h_3	h_2	h_5	h_4
i_9 :	h_4	h_1	h_5		
i_{10} :	h_3	h_1	h_5	h_2	h_4
i_{11} :	h_5	h_4	h_1	h_3	h_2

Table 30: Hospital Programs' Preferences

	1	2	3	4	5	6	7	8	9	10	11	# Places
h_1 :	i_3	i_7	i_9	i_{11}	i_5	i_4	i_{10}	i_8	i_6	i_1	i_2	4
h_2 :	i_5	i_7	i_{10}	i_6	i_8	i_2	i_3	i_{11}				3
h_3 :	i_{11}	i_6	i_8	i_3	i_2	i_4	i_7	i_1	i_{10}			3
h_4 :	i_{10}	i_1	i_2	i_{11}	i_4	i_9	i_5	i_3	i_6	i_8		2
h_5 :	i_2	i_4	i_{10}	i_7	i_6	i_1	i_8	i_3	i_{11}	i_9		1

matchings which aren't the intern-oriented matching or the hospital-oriented matching. All the stable matchings are displayed in table 31 with M_1 the intern-oriented matching and M_7 the hospital-oriented matching. The intern-

Table 31: All Stable Matchings

Matching	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}
M_1 :	h_3	h_1	h_4	h_3	h_1	h_3	h_2	h_1	h_4	h_1	h_5
M_2 :	h_1	h_3	h_4	h_3	h_1	h_3	h_2	h_1	h_4	h_1	h_5
M_3 :	h_3	h_1	h_5	h_3	h_1	h_3	h_2	h_1	h_4	h_1	h_4
M_4 :	h_1	h_3	h_5	h_3	h_1	h_3	h_2	h_1	h_4	h_1	h_4
M_5 :	h_5	h_3	h_3	h_4	h_1	h_3	h_2	h_1	h_1	h_1	h_4
M_6 :	h_5	h_4	h_3	h_1	h_1	h_3	h_2	h_3	h_1	h_1	h_4
M_7 :	h_4	h_4	h_3	h_1	h_1	h_3	h_2	h_3	h_1	h_5	h_1

oriented matching (M_1) doesn't match the hospital programs with their least preferred interns. For example h_1 is matched with $\{i_2, i_5, i_8, i_{10}\}$ not with $\{i_1, i_2, i_6, i_8\}$. The hospital programs do have a certain preference between matchings. For example h_1 is matched with $\{i_2, i_5, i_8, i_{10}\}$ in M_1 and with $\{i_1, i_5, i_8, i_{10}\}$ in M_2. The only difference between these two matchings is that they either have i_1 or i_2. The hospital program h ranks i_1 higher than i_2. This means that h prefers matching M_2 over M_1. The matching for h in M_1 is the same as in M_3 and the matching in M_2 is the same as in M_4. So h doesn't mind if they are in M_1 (M_2) or M_3 (M_4). This isn't the case for all the hospital programs. For example h_5 is indifference between M_1 and M_2; M_3 and M_4; M_5 and M_6.

3 Paired Kidney Transplantations

Matching theory has also been applied to kidney transplantations. The idea of using matching theory with kidney transplantations started in the 80's but it started getting popular in 2004-2005 with the start of kidney exchange programs. Matching with paired kidney transplantations works as follows:

Each recipient has one or more donors who is incompatible with the recipient. This can be a family member who wants to donate or someone else the recipient knows. This recipient and incompatible donor form a pair. A recipient can also have no donor who they know and who is willing to donate. There are also some donor who donate to strangers without knowing a recipient.

Paired kidney transplantations is a kind of one-to-one matching. Each pair of recipient and incompatible donor is seen as one and they are matched with another pair or person. Each recipient has a preference list with all donors on it. The recipients can either have a 0-1 preference, this is the 'American' way, or a continuous preference, this is the 'European' way. In the 'American' way a kidney can either be compatible (1) or incompatible (0). There isn't a difference between compatible kidneys. In the 'European' way there can be a difference between compatible kidneys. Here the preference increases depending on the tissue type match. Both kinds of preferences were used in practice but surgeons preferred to use the American way.

The preference of kidneys can depend on multiple factors. First there is blood type. Someone is either blood type A, B, AB or O. If someone is blood type O then they can donate a kidney to everyone no matter the blood type but they can only receive a kidney from someone with blood type O. If someone is blood type A or B then they can donate a kidney to someone with the same blood type or with blood type AB. They can receive a kidney from someone with blood type O or from someone with the same blood type. If someone is blood type AB then they can only donate a kidney to someone with the same blood type. They can receive a kidney from every blood type. This means that people, who are blood type O, are the best donor and people, who are blood type AB, are the best recipients.

The recipient always prefers their own incompatible donor to another donor who is also incompatible. Another reason for the incompatibility of a kidney is a defect in someone's gene. A recipient can have certain antibodies which make a kidney incompatible even though it is from someone with a compatible blood type.

Sometimes surgeons name a maximum that the number of kidney allocations at the same time can't exceed. This is because if you match 20 donors with 20 recipients then you'll have to perform 40 surgeries at the same time. If you do them at different times then a donor might choose not to do the surgery after their recipient received their kidney.

It is also possible for someone to have a incompatible donor and to be matched with a kidney from someone who has passed away. This match can either be allowed or not allowed. Recipients with blood type O and without a incompatible donor might not get a match because of this. These people already have a small chance of getting a kidney and this might decrease this chance even more. This is the reason why surgeons prefer to not include matches between a recipient who knows an incompatible donor (a friend/family member) and a kidney from someone who has passed away.

There usually is a priority list with all the recipients. This list ranks all the recipients in order of who needs a kidney the most. If someone is at the top of the list then it can mean that are the most likely to die. But it can also be that someone is higher on the priority list because they have a small change of finding a compatible kidney. The person with the highest priority will be matched before anyone else. When that person is matched the next person on the priority list is matched and this is repeated until everyone is matched. This way of matching is called the Simple Serial Dictatorship (SSD).

If someone has multiple incompatible donors who would like to help then there can be a priority list between the donors. For example if you need a donor and your parents and siblings would like to donate but they are incompatible then they might prefer that every option with the parent as donor is considered before the option with the sibling as donor is considered. This might cause them to withheld a possible donor (the sibling) until it is absolutely necessary. It is better to reveal all the possible incompatible donors because it gives you a better chance to be match with a compatible donor.

3.1 Two Donors and Two Recipient

Example There needs to be a few assumptions made before we start an example.

- We'll use the 0-1 preference of the 'American' way. This is the most preferred way by surgeons.
- There can only be a match between at most two donors and two recipients. This way the maximum number of surgeries that need to happen at the same time is 4.
- There can be no match between a recipient and incompatible donor pair and a donor who has passed away. This is to increase the chances of a recipient with blood type O and without a incompatible donor.

There are five people who need a kidney. From these five recipients there are four recipients who have one or more incompatible donors and there is one recipient who has no-one who is willing to donate. There is one person who has passed away and wanted to donate their organs. There is no-one who wants to just donate a kidney to a stranger without knowing someone who needs one.

The donors and the recipients have the blood types which are displayed in table 32. Some people could be a match when you look at the blood type but sometimes they can't be a match because of a defect. This defect would reject the kidney if it were transferred.

The preference list of the recipients is displayed in table 33. The parentheses indicate indifference. The donors can be divided into three groups.

Table 32: Blood Types of the Donors and the Recipients

Recipient/Donor	Blood Type
r_1	O
d_1^1	A
d_1^2	B
r_2	A
d_2^1	B
r_3	B
d_3^1	A
d_3^2	AB
d_3^3	O
r_4	AB
d_4^1	A
r_5	O
d_6	O

The first group is the group who in compatible with the recipient. This is the most preferred group. Next are the donors who participate because of the specific recipient. These donors are more preferred than other incompatible donors. The last group, who is least preferred, consists of the other donors.

Table 33: Recipients' Preferences

	1	2	3	4	5	6	7	8
r_1 :	d_3^3	$(d_1^1$	$d_1^2)$	$(d_2^1$	d_3^1	d_3^2	d_4^1	$d_6)$
r_2 :	$(d_1^1$	d_3^1	d_3^3	$d_4^1)$	d_2^1	$(d_1^2$	d_3^2	$d_6)$
r_3 :	$(d_1^2$	$d_2^1)$	$(d_3^1$	d_3^2	$d_3^3)$	$(d_1^1$	d_4^1	$d_6)$
r_4 :	$(d_1^2$	d_3^1	d_3^2	$d_3^3)$	d_4^1	$(d_1^1$	d_2^1	$d_6)$
r_5 :	d_6	$(d_1^1$	d_1^2	d_2^1	d_3^1	d_3^2	d_3^3	$d_4^1)$

Some recipients have a higher priority than others of getting a kidney. This can be because they have a smaller chance of getting a kidney (for example a recipient who has blood type O) or because they have a higher chance of dying. This priority list is displayed in table 34. The recipient r_5 has the highest priority of getting a kidney. If we use the Simple Serial Dictatorship method then that means that r_5 is the first person to possibly get a kidney. They can be matched with donor d_6 who isn't connected with any recipient. The next person on the priority list is r_1. They are only compatible with d_3^3. This can only work if one of the donor of r_1 is compatible with r_3. The donor d_1^2 is compatible with r_3. So recipient r_1 gets a kidney from donor d_3^3 and recipient r_3 get a kidney from d_1^2. The next person on

Table 34: Priority List

1 :	r_5
2 :	r_1
3 :	r_3
4 :	r_2
5 :	r_4

the priority list is r_2. They are compatible with donors from one, three and four. Recipients one and three are already match which leaves r_4 as the only option. The problem is that r_2 doesn't have a donor who is compatible with r_4. This means that they can't be matched and they have to wait until more kidneys gets available. The final stable matching is $\{(r_1, d_3^3), (r_2, \emptyset), (r_3, d_1^2), (r_4, \emptyset), (r_5, d_6)\}$.

3.2 Many Donors and Many Recipient

Example In this example we'll look at a different situation which allows a match to exist between more than two recipients and two donors. If there is someone in a match who donates a kidney without knowing someone who needs a kidney then all surgeries won't necessarily need to happen at the same time. This is because the match is a chain which start with one donor and not a closed loop of donations. The following assumptions are made.

- We'll use the 0-1 preference of the American way. This is the most preferred way by surgeons.
- There can be a match between more than two donors and two recipients.
- There can be no match between a recipient who has an incompatible donor and a donor who has passed away. This is to increase the chances of a recipient with blood type O and without a incompatible donor.

There are four recipients who need a kidney. Each recipient has one or more incompatible donors. These donors are incompatible because of their blood type or because there is another reason that increases the chance of rejection. There is also another donor who is willing to donate their kidney without wanting anything in return. There is no donor who has passed away which means there are only donors who are alive. The blood types of the donors and the recipients are displayed in table 35.

These blood types combined with a compatibility test results in the preference list in table 36. We assume that a recipient won't reject a compatible kidney.

Table 35: Blood Types of the Donors and the Recipients

Recipient/Donor	Blood Type
r_1	B
d_1^1	A
d_1^2	AB
r_2	A
d_2^1	B
r_3	A
d_3^1	AB
d_3^2	A
r_4	AB
d_4^1	B
d_5	O

Table 36: Recipients' Preferences

	1	2	3	4	5	6	7
r_1 :	$(d_4^1$	$d_5)$	$(d_1^1$	$d_1^2)$	$(d_2^1$	d_3^1	$d_3^2)$
r_2 :	$(d_1^1$	d_3^2	$d_5)$	d_2^1	$(d_1^2$	d_3^1	$d_4^1)$
r_3 :	$(d_1^1$	$d_5)$	$(d_3^1$	$d_3^2)$	$(d_1^2$	d_2^1	$d_4^1)$
r_4 :	$(d_2^1$	d_3^1	d_3^2	$d_5)$	d_4^1	$(d_1^1$	$d_1^2)$

There is also a priority list (table 37) which decides who the first person should be to get a kidney. Recipient r_3 has the highest priority in this exercise. If we assumed that there could only be a maximum of two recipients

Table 37: Priority List

1 :	r_3
2 :	r_2
3 :	r_4
4 :	r_1

and donors matched together then r_3 would get a kidney from d_5 and the other recipients wouldn't get a kidney. But this isn't the case if we don't restrict the maximal number of people in one match. Again recipient r_3 gets the first choice. They get a kidney from d_5. The next recipient r_2 gets a kidney from the incompatible donor d_3^2 from r_3. Then recipient r_4 will get a kidney from d_2^1 and lastly r_1 gets a kidney from d_4^1. This is a stable matching because nobody objects to the matching.

This chain was made possible by the donor d_5 who decided to give his kidney away to a stranger without expecting anything back. If donor d_5 was connected to a recipient who needed a donor then it would be a lot more difficult. All the surgeries would have to be perform at the same time to prevent anyone from backing out. But there is someone who donates without wanting anything in return. First d_5, r_3 and d_3^2 would be operated on. Then r_2 and d_2^1 followed by r_4 and d_4^1 and lastly r_1. The donor of r_1 would only need to donate if there were more recipients who they could donate to. This would also make the chain longer.

3.3 Successes and Problems

3.3.1 Problems

Before the start of the kidney exchange programs there were a few problems which needed to be solved. The main problem was: is it Legal? It isn't legal to buy organs with money. At least it isn't legal in the countries where they were developing kidney exchange programs. It isn't legal because it gives the wealthy an unfair advantage with respect to the less wealthy. This is essentially the same as what happens with the kidney exchange programs. You're chances of getting a kidney is increased if you know someone who is willing to donate but is incompatible with you. This means you have a higher chance of getting a kidney compared to someone who is alone. In 2007 the U.S. Senate declared kidney exchange to be legal. In some countries it is still illegal to exchange kidneys because they believe a recipient should have some kind of emotional connection with a donor if the recipient gets a kidney from a donor who is alive.

In section 3.2 we saw that it was possible for a longer exchange chain to exist. The reason why this was possible, was because someone donated their kidney out of kindness without expecting anything back. This is a problem because there aren't many people who do this. That's why kidney exchanges usually consists out of two recipients and two donors.

3.3.2 Successes and Improvements

The use of kidney exchanges has a positive influence on a lot of factors. It doesn't just result in more kidney transplants but it also decreases the medical costs. This is because the recipients won't need dialysis anymore. The wait time has also decreased because the chance of getting a kidney has increased.

The kidney exchange programs have had multiple success cases. For example in 2015 there was a kidney exchange chain in America which existed of 34 donors and 34 donors. This chain started with a donor who gave their kidney to a stranger and it ended 3 months later when a 77-year-old woman got the last kidney. There were a lot of people in the chain who had a genetic defect which made it difficult to get a kidney via the regular transplant list. The kidney exchange program made it possible for them to get a kidney.

In section 3.1 and 3.2 we discussed two examples which uses the priory list to find a match. The one with the highest priority gets the first chance at a kidney. The article 'Kidney Paired Donation and Optimizing the Use of Live Donor Organs' [6] show a different way which results in a better match. They used an optimized algorithm which looks at all possible combination of recipients and donors and chooses the best one. This experiment shows a few things. First it shows that a larger percentage of the group is matched. If also shows that the people, who are matched, have a smaller chance of rejecting the kidney. It also takes travel time into account and it shows a decrease in travel time. Finally it shows an even greater decrease in medical cost compared to the previous method. Even though it is just a theoretical experiment it shows that there could be improvements made to the current kidney matching programs which will improve the results from the program.

4 Conclusion

We've seen the basics of matching theory. We now know how matching works, when a matching is stable and some variations of the standard matching models. We have shown that in some cases it isn't always possible to get a stable matching and in some cases there is always at least one stable matching.

These basic matching problems led to the use of matching theory with paired kidney transplantations. First it caused a lot of discussion about legitimacy of kidney exchanges and about the restriction which should be in place. Now that matching theory has been applied during kidney exchange programs, the programs have shown very positive result. A lot of transplantations have been made possible because of the programs. The programs also show room for improvement and for further research.

5 Discussion

There are a few improvements which can be made to either paired kidney transplantations or matching theory in general. In section 1.4 we constructed a lower bound for the maximum numbers of stable matchings in one-to-one matching. There isn't a formula for the actual maximum number of stable matchings. This is still a problem which could possibly be solved.

In section 1 we saw a man-oriented and a woman-oriented algorithm. The algorithm is positive for one of the groups while it is negative for the other group. A question which could be asked is: 'Is there a algorithm which gives both groups the same kind of result'? This result would be the most fair towards both groups. This algorithm hasn't been found yet.

The last improvement which we are going to discuss is the improvement in the matching process used with paired kidney transplantations. At the end of section 3.3.2 we mentioned a article [6] which discusses a possible improvement for the kidney exchange programs. This is a theoretical experiment which means it might be different in reality but it does show room for improvement. It shows that there might be a better way to match paired kidney transplantations which also results in a stable matching. This is a study which can be looked into further.

References

[1] Gale, D. and L.S. Shapley, (1962), 'College Admission and the Stability of Marriage', *The American Mathematical Monthly*, 69(1), pp. 9-15.

[2] Gusfield, Dan and Robert G. Irving, (1989), *The Stable Marriage Problem: Structure and Algorithm*, The MIT Press.

[3] Maschler, Michael, Eilon Solan and Shmuel Zamir, (2013), Stable matching, *Game Theory* (pp. 884-915), Cambridge University Press

[4] Roth, Alvin E. and Marilda A. Oliveira Sotomayor, (1990), *Two-sided matching: A study in game-theoretic modeling and analysis*, Cambridge University Press.

[5] Roth, Alvin E., Tayfun Sönmez and M. Utku Ünver, (2004), 'Pairwise Kidney Exchange', *Journal of Economic Theory*, 125(2), pp. 151-188. [Online] DOI:10.1016/j.jet.2005.04.004 (Accessed: 19 May 2016)

[6] Segev, Dorry L., Sommer E. Gentry, Daniel S. Warren, Brigitte Reeb, Robert A. Montgomery, (2005), 'Kidney Paired Donation and Optimizing the Use of Live Donor Organs', *JAMA*, 293(15), pp. 1883-1890. [Online] DOI:10.1001/jama.293.15.1883 (Accessed: 23 May 2016)

[7] Sönmez, Tayfun, (2013), *How Does Matching Theory Improve Our Lives?*, Presentation, Arne Ryde Mini-Course on Economic Design. Available at: https://www2.bc.edu/tayfun-sonmez/MatchingApplications-Lund.pdf (Accessed: 25 May 2016)

[8] *Longest Kidney Chain Ever Completed Wraps up at UW Hospital and Clinics*, (2015) Available at: http://www.uwhealth.org/news/longest-kidney-chain-ever-completed-wraps-up-at-uw-hospital-and-clinics/45549 (Accessed: 23 May 2016).