The Deferred Acceptance Algorithm

A thesis presented for a Bachelor degree in Economics and Business

Luigi Croce

Roma, June 2016

LUISS Università Guido Carli

Contents

Chapter 1

What is a Matching Market

People's lives are full of choices. They have to choose how to maximize their utility, having unlimited needs but limited means. Economics is the science that studies the reasons for these choices and their outcomes. Therefore questions, such as which school people attend, which job they get, who they are married with, are of fundamental importance. It turns out that these questions do not depend exclusively on the choices people make, because they must be chosen back. To express this concept in a more formal way we could say that: while, in many markets, price is the only variable necessary to equate demand and supply, there are other markets in which price alone is not enough. The markets in which the things we choose must also choose us back are called "matching markets".

1.1 Matching Markets Analysis

The very first thing to say regards the conditions for such a market to exist:

1. The market must be thick, with an high enough number of partipants
2. Those participants must follow the market rules, whichever they are

In practice these two conditions are very difficult to materialize, the first one because people might not be aware of the existence of such a market or because they might not be able to signal properly their qualifications to the other party. The second condition often fails because people have an incentive to cheat, either when the rules are not strict or when they are not subject to punishment if they do not comply with such rules. Finding a job, applying to certain schools, or looking for a wife, are not markets that can be regulated, and therefore other mechanisms come into play.

In a matching market prices are still important, (you still have to pay tuition or buy a ring to your wife) but other factors, such as searching and wooing are important as well.
After this first stages of market formation, the analysis must switch towards the single participants of the market. In fact, while in commodity markets the same offer is made to all participants, in matching markets each transaction may have to be considered separately (for example each applicant will have his own interview, or each man his date). Single participants need time to evaluate proposals, and this might create congestion in the market, meaning that people wait long time for an answer.
Another problem might be that since people do not like to see competition will act in advance with respect to their competitors, and make exploding offers, meaning offers that have to be accepted or rejected before participants have time to receive offers from someone else.
Psychology as well plays a great role: sometimes a college might not offer a position to its best applicant, only because there is an high probability that the same student will be accepted into a more prestigious program and accept that offer.
In turn, market analysis will depend greatly on the details of the market, including the culture and the psychology of the participants.
In order to proceed towards a deeper and more complete analysis of reality I believe it might be of some use the analysis of the marriage model.

1.2 The Marriage Model

The model presents two groups of agents, represented by two disjoint sets of Men and Women. The purpose of each man and each woman is to be matched with their most preferred participant of the other set. The set of outcomes, defined as matches, can be of three type:

1. between a man and a woman
2. between a man and himself
3. between a woman and herself

The model can be conceptualized in the following way:
Suppose there are two disjoint sets $M = m_1, ..., m_n$ and $F = w_1, ..., w_n$ of men and women, each of whom has complete and quasi-transitive preferences over the set of individuals of different sex (plus the chance of being unmatched).

Complete preferences are defined as binary relation R over a set $M \cup F$ if for all m and w in $M \cup F$, m is related to w or w is related to m (or both). In mathematical notation, this is $\forall m, w \in M \cup F \ (mRw \vee wRm)$
Quasi-transitive preferences are instead defined this way: If $m_1 \succ m_2$ and $m_2 \succ m_3$ then $m_1 \succ m_3$.
Preferences are represented as a list $P(m_i) = w_2, w_4, w_7, ..., m_i$ meaning that the agent m_i prefers w_2 to any other choice, w_4 to any choice but w_2, and so on, until he prefers to remain unmatched (denoted in the list as m_i).
Therefore when a proposal is received it may be considered acceptable or unacceptable whether it is present in the preference list of the women. Furthermore to express the fact that agent m_i prefers w_2 to w_4 we will use the following notation: $[w_2 \underset{m_i}{>} w_4]$ where the subscript m_i under the $>$ sign indicates the agent who presents a certain preference in his preference list.
Women preferences are conceptualized in the exact same way, meaning that in the list $P(w_i) = m_3, m_5, m_1, ..., w_i$ the female agent w_i prefers some male agents, and at some point she prefers to remain unmatched. Again to denote the preference that wi has toward certain males we use the notation $[m_3 \underset{w_i}{>} m_5]$.
An outcome of such a model is called a matching and is a one-to-one correspondence from the set $M \cup W$ into itself. This correspondence is of order two, meaning that if a man m is matched to a woman w, then the woman w is matched to the man m. Such a matching is expressed in the following way:
if $\mu(m) = w$ then it must be also true that $\mu(w) = m$. A matching can also be represented as a set of matched pairs: $\begin{matrix} w_1 & w_3 & w_5 & w_2 & w_4 & m_5 \\ m_2 & m_3 & m_1 & m_4 & w_4 & m_5 \end{matrix}$ where in this example $\mu(w_1) = m_2$ and w_4 and m_5 are matched to themselves. However matches may not always be stable. It might be the case that a *blocking pair* is formed by a pair of agents, if each one of them prefers to be matched with the member of the pair, instead of the one who he was matched with. In symbols: if the pair (m, w) has preferences such that $[w \underset{m}{>} \mu(m)]$ and $[m \underset{w}{>} \mu(w)]$ then they form a blocking pair.
Furthermore a match can be also *blocked by an individual* k if k prefers being single to being matched with $\mu(k)$. In symbols: $[k \underset{k}{>} \mu(k)]$.
Therefore it can be concluded that a matching μ is stable if it is not blocked by any individual or any pair of agents.
Having defined these characteristics now it comes into play the deferred acceptance algorithm, which is deemed to produce such stable matchings discussed above. The algorithm works in the following way:

- step 0:
  - If some preferences are not strict, then it must be specified a way that allows the algorithm to arbitrarily break such ties (this task is only needed if some agents are indifferent among one matching or another, and in this case the problem can be solved in two different ways. The algorithm might ask only for strict preferences, or break such ties at random).
- step 1:
  - Each man m proposes to his first choice.
  - Each women rejects any unacceptable proposal, and if she receives more than one proposal she only holds the most preferred one.
- step k:
  - Each man rejected at step $k - 1$ makes a new proposal to his first choice who has not yet rejected him. In case he will have no more women to propose to he will remain unmatched.
  - Each woman holds her most preferred offer to date, and rejects the rest.

The algorithm will stop when no further proposals can be made by any men. All the proposals held by women will form a match, and all the other women and men will be matched to themselves.

1.2.1 Optimal Properties of Stable Matches

This area has been extensively analyzed in previous papers.
In its paper of 2008, Roth states that: "The man-optimal stable matching μ_M is weakly Pareto optimal for the man in the set of all matchings."
This means that there cannot be a set of matches, even unstable, that makes all men better off. However this result about optimality is only true for the party proposing. This result must be kept in mind since it will be useful in the subsequent analysis about aggregate the welfare produced by the stable matches of the algorithm.

Chapter 2

Coding the Algorithm

The purpose of this chapter is to explain how, using a software like R, users can have the chance to solve complex optimization problem in a market that has the characteristics of a matching market.

2.1 R code:

In R we create a function with arguments $x, y, m.prefs, andw.prefs$. Among these four arguments we have arguments x and y which represent the number of males and females that constitute our experiment (x = number of males, y = number of females), and two matrices $m.prefs$ and $w.prefs$. These two matrices have the following structures: $m.prefs$ has as many columns as the number of males, with each row representing the preferences, from the highest to the lowest (in the first row we have the most preferred choice and so on) for each member. If a person would, at some point, prefer to remained unmatched we will insert zeros for the remaining females. The matrix $w.prefs$ has a specular structure, with each column representing a female, and with the preferences for that female ranked in the rows from the highest to the lowest, with the possibility of inserting zeros to represent the choice of remaining unmatched.
images to have an idea.
From this point onwards I will write lines of code with explanations regarding their meaning below:
daa1 < − function(x, y, m.prefs, w.prefs){
m.history < − rep(0, x)
w.history < − rep(0, y)
m.historyold < − rep(0,x)

The first line creates the function and specifies which arguments must be put into it. These are x and y, the numbers of males and females respectively, and preferences matrices for males and females. The following three lines of code create three vectors, with the first two representing the history of each male and female, meaning that for each man and women we specify with whom they are currently matched. Notice that since it is just the beginning of our process these two vectors are vectors of zeros, because there are no matches yet. In these two vectors each row represent the current match of each agent. The third row represents a vector, still filled with zeros, but that serves another purpose that I will explain later on.

```
m.singles < − 1:x
w.singles < − 1:y
```

In this second step we create two vectors, representing the singles men and women, that at this point will be constituted of all the males and the females.

```
fun1 < − function(w.history, m.singles, m.history, w.singles, m.mat,
stay.single) {
 m.history < − m.historyold[m.singles]
 m.history < − sapply(m.history, function(m.history) m.history+1 )
m.historyold[m.singles] < − m.history
```

Now we create a new function inside the original function. The reason is that the algorithm we want to write consists of a repetition of a certain process over a list of elements. Therefore we specify first which is the process and later how this process will be applied. This new function will take as arguments many vectors that we have previously specified. Inside this function we will start the process we are really interested in. Our first goal is to change the vector *m.history* in order for it to reflect the current round of the males proposing. Therefore we exctract from the vector *historyold* the value of the history of single males, then we add 1 to each element of the vector and we store back the informations in the *m.historyold* vector. This process allows males which might have been matched and then unmached to propose to the right woman.

```
if(any(m.history>length(m.prefs[,1]))){
 m.singles < − m.singles[m.history<=length(m.prefs[,1])]
 m.history < − m.history[m.history<=length(m.prefs[,1])]
 }
```

Now we need an "if" statement to evaluate whether inside the vector *m.history* there is a number corresponding to a round which is higher than the maximum round a single male can reach. If this is the case we will eliminate the male from the vector of singles and eliminate his preference from the preference list.

```
 if(length(m.singles)==0|length(m.history)==0){
```

return(list(w.history=w.history, m.singles=m.singles, m.history=m.history, w.singles=w.singles, m.historyold=m.historyold))

}

Here we insert two conditions (there are no more singles or no more people in m.history) for which the function should end and return a list of vectors.

offers < − rep(0, length(m.singles))

for (i in 1:length(m.singles)) {

offers[i]< − m.prefs[m.history[i],m.singles[i]]

}

if (any(offers==0)) {

stay.single < − m.singles[offers==0]

m.singles < − m.singles[stay.single!=m.singles]

offers < − offers[offers!=0]

Now that I have solved the issues regarding the history, I will start to construct a vector of offers made by all the singles of this round. I will first create a vector of zeros, then a "for" loop to fill this vector of zeros with the right offer of this round, and later an "if" statement to eliminate the offers=0, that in our model represent a person who prefers to remain unmatched.

if (length(offers)==0){

return(list(w.history=w.history, m.singles=m.singles, m.history=m.history, w.singles=w.singles, stay.single=stay.single))}

Here I specify that if there are no offers it means that the algorithm has ended.

approached < − unique(offers)

proposers < − m.singles

Now I define the two vectors of females and males that will actually take part in the proposition and refusal process in this round. Those are the females approached by at least one man, and the m.singles of this round.

for(j in approached) {

for (k in 1:length(proposers)){

Here I start two for loops with the intention of evaluating each girl and each man in the two vectors above. The females will be called j and the males will be called k.

if(w.history[j]==0 && any(w.prefs[,j]==proposers[k]) && offers[k]==j) {

w.history[j] < − proposers[k]

m.singles < − m.singles[! m.singles $\%in\%$ **proposers[k]]**

w.singles¡-w.singles[! w.singles $\%in\%$ **j] }**

This "if" statement specifies a first case of the algorithm, in which the woman j is not matched to another man already. Further I have to specify that the proposer k is somewhere on the preference list for the woman j, and lastly I have to be sure that the offer that k is making at this round is actually toward j. After this conditions are met a match will occur, therefore in the vector w.history in the position of the woman j the proposer k will be inserted and will represent a match. Then the single male will be eliminated from the vector *m.singles*, as well as the woman from w.singles.

else if (w.history[j]!=0 && any(w.prefs[,j]==proposers[k]) && offers[k]==j && match(proposers[k] , w.prefs[,j])¡match(w.history[j], w.prefs[,j])){
m.singles $<-$ c(m.singles,w.history[j])
w.history[j] $<-$ proposers[k]
m.singles $<-$ m.singles[! m.singles %in% proposers[k]]
w.singles $<-$ w.singles
}
}
}

Now the other case, in which the woman has already a match. If this is the case k will still propose to her, and therefore the if statement will still require the conditions that k is in j's preference list and that at this round k is actually proposing to j, but now another condition to met is that the proposer k of this round is actually higher on the preference list of j, than the previous man matched with her. If this conditions are met then the man formerly matched with the woman j will return single, and all the other steps are the same as above, since a match will occur.

output $<-$ list(w.history=w.history, m.singles=m.singles, m.history=m.history, w.singles=w.singles) return(output) }

At this point the function has evaluated the match of this round and will return a list of vectors.

iter=function(lista)
{return(do.call(fun1, lista))}

Here I specify another function called iter, which performs iteration of the function do.call over the list I will give it.

lres=list(w.history, m.singles, m.history, w.singles, m.mat)
while (sum(lres[[2]])¿0) {
lres=iter(lres)
}

Now I just have to combine these two processes and tell R to perform the iter function over the list **lres** that I have defined. However this repetition cannot be carried out forever, in fact here I specify that it is only while m.singles is not empty.

return(list("matches"=lres[[1]], "wsingles"=lres[[4]])) }

Now my process is over and I can take the vectors I want to show and tell the function to display them.

2.1.1 Processing Capacity

Depending on the number of participants the iterations performed by the algorithm vary, therefore I believe it might be useful to assess the magnitude of data that can be processed through this algorithm. With a standarc PC it takes only seconds to perform iterations among two groups of 100 people, however time increases exponentially with the number of players. It takes 10 minutes, for example, to perform iterations among one group of 1000 and one group of 500. This is because the number of preferences in the two matrices is 500.000 in this case, while it was only 10.000 for two groups of 100 individuals each.

Chapter 3

Social Welfare Application

Now that I have defined the algorithm and I have proposed a solution to program it using the R software, my analysis will shift the focus toward welfare problems connected with the proposed algorithm. The algorithm produces stable results because each of the participants could not find a better result by himself, given that all were all given perfect information. Therefore a question that might be of interest would be: How big is the loss in social welfare that results from stable matches, as opposed to optimal matches. This task poses some important challenges, both in the definition of concepts and in the interpretation of results. Furthermore we must specify which measure we are going to use to proxy the welfare of society; which utility function we can attribute to each individual and which are the more relevant aspects we want to take under consideration. These issues must be kept in mind to interpret the result of the study in the correct way.

3.1 Price of Anarchy

A sensible and quick measure of the analysis described above can be provided by the price of anarchy, which is defined in the following way:
Consider a game $G = (N, S, u)$, defined by a set of players N , strategy sets S_i for each player and utilities u_i : S $\rightarrow \mathbb{R}$(where $S = S_1 ... S_n$ also called set of outcomes). We can define a measure of efficiency of each outcome which we call welfare function $W : S \rightarrow \mathbb{R}$.
This function must be meaningful for the particular game being analyzed. In this particular case the sum of players utilities $W(s) = \sum_{N=1}^{N} u_i(S)$ is the function to be maximized.

We can define a subset $E \subseteq S$ to be the set of strategies in equilibrium (for example stable matches).
The Price of Anarchy is then defined as the ratio between the optimal solution and the 'worst equilibrium':

$$PoA = \frac{\min_{s \in E} W(s)}{\max_{s \in S} W(s)} \tag{3.1}$$

3.2 Studies

Therefore I will try to construct and conduct a study, taking the variation in the price of anarchy to formulate certain hypothesis:

1. What if I change the party proposing?
2. What if I increase the number of participants?
3. What if the party proposing and the party that receives the proposals have different utility functions?
4. What if the number of proposers and the number of receivers are different?

Case #1

What if I change the party proposing?
In order to analyze the changes in social welfare I will first define a simple utility function, equal for all participants, and look at the loss in social welfare in the two cases. $U = 1/x$ where x is the position of the match in the preference list of the individual.
Let's examine a situation with given set of preferences:

m_1	m_2	m_3	m_4	m_5
1	4	4	1	1
3	3	2	4	3
2	2	1	2	4
4	1	3	0	0

Male Preferences

w_1	w_2	w_3	w_4
2	5	3	1
3	4	1	4
1	1	2	5
4	2	4	2
5	3	0	0

Female Preferences

Therefore the set of stable and optimal mathces with male proposing are:

μ_1	μ_2	μ_3	μ_4	μ_5
1	2	3	4	5
1	3	2	4	0

Stable Matches

μ_1	μ_2	μ_3	μ_4	μ_5
1	2	3	4	5
1	4	3	0	2

Optimal Matches

Following from these results, the utility from stable mathes is $U = 4,11$, while

$Uopt = 5,08$ and therefore:
$PoA = 0,81$

When instead females propose these are the stable matches:

μ_1	μ_2	μ_3	μ_4	μ_5
1	2	3	4	5
4	1	3	2	0

Stable Matches

μ_1	μ_2	μ_3	μ_4	μ_5
1	2	3	4	5
1	4	3	0	2

Optimal Matches

Following the same procedure, the utility from stable matches is
$U = 3,86$ and therefore the $PoA = 0,76$.

The **conclusion** to be drawn from these example is that the set of stable matches might change, and so does the welfare of participants.
In particular what I believe it is important to notice, is that the utility of the party proposing is much higher:
In particular for men it changes from 2,75 to 1,33, less then half.
For women it changes from 1,36 to 2,53.

Case #2

What if I increase the number of participants?
In general increasing the number of participants seems to have a negative effect on the price of anarchy.
However the study here becomes more interesting, since the results will differ depending on the preference list of people. In the case in which participants might dislike certain participants of the other group the loss of social welfare is in fact magnified.
This happens because when everybody has the possibility to be matched with everybody else under the algorithm, (meaning every participant would accept someone of the other group instead of remaining unmatched) it becomes more difficult to increase social welfare.

Lets consider these as the two preference matrices:

m_1	m_2	m_3	m_4
3	1	2	4
4	2	3	3
1	3	4	2
2	4	1	1

Male Preferences

w_1	w_2	w_3	w_4
1	3	1	2
0	2	2	3
0	1	3	4
0	4	4	1

Female Preferences

μ_1	μ_2	μ_3	μ_4	μ_5
1	2	3	4	0
3	4	2	0	1

Stable Matches

μ_1	μ_2	μ_3	μ_4	μ_5
1	2	3	4	0
3	4	2	0	1

Optimal Matches

In this case stable matches are equal to optimal matches, and so are the utility generated. Thereore the **price of anarchy** in this case is **equal to** 1.
Such a price of anarchy means that if people follow their preferred alternative, then the collective welfare is maximized.
If we increase the number of participants, with preferences as follow:

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
2	1	1	3	3	5	6	8
3	5	6	2	8	1	7	5
5	2	5	1	6	2	8	6
1	7	3	5	5	3	2	3
4	8	2	6	0	4	1	2
0	0	4	0	0	0	3	1
0	0	8	0	0	0	0	0
0	0	0	0	0	0	0	0

Male Preferences

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
1	3	5	7	1	8	5	2
2	1	6	3	2	7	3	5
3	2	1	8	5	6	2	3
4	0	3	1	7	5	1	4
5	0	2	0	3	4	6	1
6	0	0	0	0	3	0	0
0	0	0	0	0	2	0	0
0	0	0	0	0	1	0	0

Female Preferences

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	μ_{11}
1	2	3	4	5	6	7	8	0	0	0
2	1	5	0	3	0	0	6	4	7	8

Stable Matches

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}
1	2	3	4	5	6	7	8	0	0
2	1	5	0	3	0	4	6	7	8

Optimal Matches

Stable matches generate utility of $U = 7,87619$
Optimal matches instead $U = 8,5333$
Price of anarchy therefore is $= 0,9229$ (3.1) as defined in the previous equation.

Now the case with random preferences over each individual of the other group:

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
8	4	4	1	6	7	5	3
2	5	3	7	7	1	3	4
7	1	5	4	2	3	4	8
6	3	6	8	4	5	6	1
1	8	1	2	1	8	1	2
4	7	8	6	8	2	8	5
5	6	7	5	5	4	2	7
3	2	2	3	4	6	7	6

Male Preferences

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
4	3	2	1	6	3	4	3
3	1	6	2	1	6	3	4
2	4	5	3	3	5	8	1
6	5	4	6	2	4	7	7
8	2	1	5	7	2	5	8
7	6	3	8	8	8	6	2
5	7	8	7	5	1	1	5
1	8	7	4	4	7	2	6

Female Preferences

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
1	2	3	4	5	6	7	8
8	4	3	1	6	7	5	2

Stable Matches

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
1	2	3	4	5	6	7	8
8	4	6	1	2	7	5	3

Optimal Matches

Stable matches $U = 9,875$
Optimal matches $U = 10,209524$

Price of anarchy $= 0,9672$

Here we can see that for people who accept every person of the other group as a match, the PoA is higher: $0,96$ compared with $0,92$.

Case #3

What if the party proposing and the party that receives the proposals have different utility functions? Lets try here to model the game having as utility functions the usual $1/x$ and $1/2x$, therefore the second utility function produces much less utility. Now lets see the variation in the price of anarchy and derive some results.

Male Preferences

m_1	m_2	m_3	m_4	m_5	m_6
1	5	5	5	3	5
2	3	6	1	2	3
5	6	1	2	1	4
4	2	3	3	5	1
6	4	4	6	4	2
3	1	2	4	6	6

Female Preferences

w_1	w_2	w_3	w_4	w_5	w_6
6	6	4	6	5	2
3	1	6	1	1	4
5	5	3	5	3	3
1	2	2	3	6	5
2	3	5	2	2	6
4	4	1	4	4	1

Stable Matches

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6
1	2	3	4	5	6
1	6	5	3	2	4

Optimal matches in this case are the same if the utility function is the same.

However in the case there are different utility functions the optimal matches are:

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6
1	2	3	4	5	6
1	6	5	2	3	4

Optimal Matches

U of stable matches in different utilities functions case:
$U = 5$
$U.opt = 5,475$
The price of anarchy becomes therefore: $0,981$
Which are the conclusions to be drawn? First of all it appears that stable matches are very efficient, and they depend on the random allocation of preferences. The further consideration is that they are extremely effienct in the case in which the utility functions are equal for all participants. This is a strong assumption to be made and not so true in real life. However it is also true that it is impossible to ascertain which person derives a higher utility from a certain event. This said I believe that once again the deferred acceptance algorithm produces very good results in term of welfare for the society. Furthermore each participants behaves in a way that is actually plausible because it just respects its own preferences. This has notorious positive implications in practice.

Case #4

What if the number of proposers and the number of receivers are different?

m_1	m_2
3	2
2	1
1	3

Male Preferences

w_1	w_2	w_3
2	2	1
1	1	2

Female Preferences

μ_1	μ_2	μ_3
1	2	0
3	2	1

Stable Matches

μ_1	μ_2	μ_3
1	2	0
3	2	1

Optimal Matches

U stable $= 3, 5$
U optimal $= 3, 5$
Price of anarchy $= 1$
Now lets increase the number of participants:

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
3	2	2	5	5	5	2	1
5	3	3	2	4	3	4	4
2	1	4	4	3	4	3	2
4	5	5	3	2	1	1	5
1	4	1	1	1	2	5	3

Male Preferences

w_1	w_2	w_3	w_4	w_5
1	2	5	2	6
5	3	7	5	3
9	1	9	1	4
4	7	10	7	10
3	6	8	8	5
6	9	3	6	8
2	5	2	3	9
8	4	1	9	7
7	10	4	10	1
10	8	6	4	2

Female Preferences

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	
1	2	3	4	5	6	7	8	9	10	Stable Matches
1	2	0	0	4	5	3	0	0	0	

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}	
9	2	1	5	6	3	4	7	8	10	Optimal Matches
1	2	3	4	5	0	0	0	0	0	

U from stable matches is $= 6,4$
U from optimal matches is $= 6,958$
Price of anarchy $= 0,919$ Here the PoA with few players is 1 and decreases to 0,91

with more participants.
In the case in which we had same number of participants, PoA started at 1 and deceased to 0,96.
Therefore on average the disparity in the number of participants is negative from a social welfare perspective, when stable matches are formed.

3.3 Final Conclusions

In this paper I have proposed a brief overview of matching markets, with a special focus on the marriage model.
I have programmed the algorithm in order to have immediate results, to be used to analyze the properties of such algorithm.
By doing so I was able to empirically analyze the welfare properties of stable matchings, and provide some empirical conclusions.

Welfare Properties There are four variables that determine the loss in social welfare coming from stable matchings, if social welfare is thought of as the sum of utilities.
Those are:

1. Depending on which party is proposing to the other, results change, and benefit, in term of aggregate utility, the party proposing.

2. The number of participants: on average the higher the number of participants the higher the loss.

3. The difference in utility functions of participants: on average the more different the utility functions the higher the loss.

4. The disparity in the number of men and women: on average the higher the disparity in the number of men and women, the higher the loss.

Bibliography

[Bodine-Baron et al.(2011)Bodine-Baron, Lee, Chong, Hassibi, and Wierman] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman. Peer effects and stability in matching markets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6982 LNCS, pages 117–129, 2011. ISBN 9783642248283. doi: 10.1007/978-3-642-24829-0_12.

[Demange and Gale(1985)] G. Demange and D. Gale. The Strategy Structure of Two-Sided Matching Markets. *Econometrica*, 53(4):873–888, 1985. ISSN 00129682. doi: doi:10.2307/1912658. URL citeulike-article-id:7176937$\delimiter"026E30F$nhttp://dx.doi.org/10.2307/1912658.

[Gale and Shapley(2013)] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 120(5):386–391, 2013. ISSN 00029890. doi: 10.4169/amer.math.monthly.120.05.386.

[Klaus and Klijn(2007)] B. Klaus and F. Klijn. Paths to stability for matching markets with couples. *Games and Economic Behavior*, 58(1):154–171, 2007. ISSN 08998256. doi: 10.1016/j.geb.2006.03.002.

[Kojima et al.(2013)Kojima, Pathak, and Roth] F. Kojima, P. A. Pathak, and A. E. Roth. Matching with couples: Stability and incentives in large markets. *Quarterly Journal of Economics*, 128(4):1585–1632, 2013. ISSN 00335533. doi: 10.1093/qje/qjt019.

[Roth(1988)] A. E. Roth. *The Shapley Value*. 1988. ISBN 1483295052. doi: 10.1017/CBO9780511528446. URL https://books.google.com/books?id=JK7MKu2A9cIC{&}pgis=1.

[Roth and Sotomayor(1983)] A. E. Roth and M. Sotomayor. THE COLLEGE ADMISSIONS REVISITED. *Econometrica*, 51(3), 1983. ISSN 0012-9682. URL http://search.ebscohost.com/login.aspx?direct=true{&}db=bth{&}AN=4998993{&}site=ehost-live{&}scope=site.

[Roth and Sotomayor(1992)] A. E. Roth and M. Sotomayor. Chapter 16 Two-sided matching, 1992. ISSN 15740005.

[Roth et al.(1990)Roth, Vate, and Vande Vatei] A. E. Roth, J. H. V. Vate, and J. H. Vande Vatei. Random Paths to Stability in Two-Sided Matching. *Source: Econometrica Econometrica*, 58(6):1475–1480, 1990. ISSN 00129682. doi: 10.2307/2938326. URL http://www.jstor.org/stable/2938326$\delimiter"026E30F$nhttp://www.jstor.org/page/info/about/policies/terms.jsp.

.1 Code

```
daa1 <- function(x, y, m.prefs, w.prefs){
 m.history <- rep(0, x)
w.history <- rep(0, y)
m.historyold <- rep(0,x)
m.singles <- 1:x
w.singles <- 1:y
fun1 <- function(w.history, m.singles, m.history, w.singles, m.mat,
stay.single) {
 m.history <- m.historyold[m.singles]
 m.history <- sapply(m.history, function(m.history) m.history+1 )
m.historyold[m.singles] <- m.history
if(any(m.history>length(m.prefs[,1]))){
 m.singles <- m.singles[m.history<=length(m.prefs[,1])]
 m.history <- m.history[m.history<=length(m.prefs[,1])]
 }
 if(length(m.singles)==0|length(m.history)==0){
return(list(w.history=w.history, m.singles=m.singles,
m.history=m.history, w.singles=w.singles, m.historyold=m.historyold))
}
offers <- rep(0, length(m.singles))
for (i in 1:length(m.singles)) {
 offers[i]<- m.prefs[m.history[i],m.singles[i]]
}
 if (any(offers==0)) {
 stay.single <- m.singles[offers==0]
 m.singles <- m.singles[stay.single!=m.singles]
 offers <- offers[offers!=0]
 if (length(offers)==0){
            return(list(w.history=w.history, m.singles=m.singles,
m.history=m.history, w.singles=w.singles, stay.single=stay.single))}
 approached <- unique(offers)
 proposers <- m.singles
 for(j in approached) {
 for (k in 1:length(proposers)){
 if(w.history[j]==0 && any(w.prefs[ ,j]==proposers[k]) && offers[k]==j)
{
```

```
w.history[j] < − proposers[k]
m.singles < − m.singles[! m.singles %in% proposers[k]]
w.singles¡-w.singles[! w.singles %in% j] }
else if (w.history[j]!=0 && any(w.prefs[ ,j]==proposers[k]) && offers[k]==j && match(proposers[k] , w.prefs[ ,j])¡match(w.history[j], w.prefs[ ,j])){
m.singles < − c(m.singles,w.history[j])
w.history[j] < − proposers[k]
m.singles < − m.singles[! m.singles %in% proposers[k]]
w.singles < − w.singles
}
}
}
output < − list(w.history=w.history, m.singles=m.singles, m.history=m.history, w.singles=w.singles) return(output) }
iter=function(lista)
{return(do.call(fun1, lista))}
lres=list(w.history, m.singles, m.history, w.singles, m.mat)
while (sum(lres[[2]]) > 0) {
lres=iter(lres)
}
return(list("matches"=lres[[1]], "wsingles"=lres[[4]])) }
```