Stable Matching in Preference Relationships

by

Elizabeth Mary Philpin

submitted in part fulfilment of the requirements

for the degree of

Master of Science

in the subject

Mathematics

at the

University of South Africa

Supervisor: Prof. K. Swanepoel

Joint Supervisor: Prof. A. Barnard

November 2006

ABSTRACT

It is the aim of this paper to review some of the work done on stable matching, and on stable marriage problems in particular.

Variants of the stable marriage problem will be considered, and the similarities and differences from a mathematical point of view will be highlighted. The correlation between preference and stability is a main theme, and the way in which diluted or incomplete preferences affect stability is explored.

Since these problems have a wide range of practical applications, it is of interest to develop useful algorithms for the derivation of solutions. Time-complexity is a key factor in designing computable algorithms, making work load a strong consideration for practical purposes. Average and worst-case complexity are discussed.

The number of different solutions that are possible for a given problem instance is surprising, and counter-intuitive. This leads naturally to a study of the solution sets and the lattice structure of solutions that emerges for any stable marriage problem. Many theorems derive from the lattice structure of stable solutions and it is shown that this can lead to the design of more efficient algorithms.

The research on this topic is well established, and many theorems have been proved and published, although some published proofs have omitted the detail. In this paper, the author selects some key theorems, providing detailed proofs or alternate proofs, showing the mathematical richness of this field of study.

Various applications are discussed, particularly with relevance to the social sciences, although mention is made of applications in computer science, game theory, and economics.

The current research that is evident in this subject area, by reference to technical papers in periodicals and on the internet, suggests that it will remain a key topic for some time to come.

Keywords: Stable marriage, Matching algorithms, College admissions problem

TABLE OF CONTENTS

1 Background

This section gives a brief history of the stable marriage problem and its social applications in the last sixty years.

1.1 Hospitals and Residents

In 1950 the US government centralised the application system whereby graduating medical students (residents) were matched with hospitals. Prior to centralisation, the applications were processed by each hospital independently, with unavoidable contention and delay. Although centralisation was an improvement, and the preferences of both hospitals and residents were considered, there was still some dissatisfaction as the matching was not always stable; some hospitals did not get the residents they preferred, and were "preferred-by". Also, some students realised that they could benefit by declaring false preferences.

In 1952 the NRMP algorithm was introduced (National Residents Matching Program) that delivered a stable matching, favouring the hospitals. It was a considerable improvement in terms of administration and fairness (between hospitals) and it is still in use today, although it has recently been re-designed to moderate the bias against residents.

1.2 Colleges and Students

Ten years later, in 1962, David Gale and Lloyd Shapley published a paper (Gale, Shapley 1962) that addressed the same problem but this time it was between colleges and applicant students that were leaving high school.

Once again, the application whirlpool that accompanies independent approaches, pre-emptive offers, and ultimatums to accept was causing social havoc and, longer term, dissatisfaction with the resulting assignments. The need to produce a stable matching was very strongly motivated.

It appears that the previous NRMP work was not published nor widely recognised, and the Gale-Shapley paper with its published and proven theorems was regarded as seminal, and a breakthrough. It included a detailed procedure to derive a stable matching between two sets of disjoint elements. This breakthrough had two main aspects:

1. the procedure proved the existence of a stable matching for all instances of the problem.
2. a stable matching (one that would not break down because individuals could do better by private arrangement) could be generated and used for the college admission system if full preference lists were given by both students and colleges.

1.3 Main Contributors

Since that time much work has been done to analyse stable matchings, the stable marriage problem and variations thereof, algorithmic complexity in the derivation of stable solutions, and extensions into game theory, economics, and computer networks.

The work done by Donald Knuth, Dan Gusfield, and Robert Irving has been particularly important. Donald Knuth's book (1976) was devoted entirely to the stable marriage problem; it created a considerable amount of interest in this problem as a field of study for mathematicians and computer scientists, and for the design of efficient matching algorithms.

Gusfield and Irving (1989) extended Knuth's work to study the structure of solution sets, the application to economic and social sciences, and associated matching problems such as the "stable roommates problem". See section 5.3.

McVitie and Wilson (1971) were the first to obtain an algorithm to compute the full set of stable matchings for the one-to-one matching model, and in recent years researchers have developed algorithms for many-to-many models.

Alvin Roth pioneered the exploration of stable matching techniques from a game-theoretic point of view, investigating two-sided matching markets, particularly the labour market. The comprehensive book by Roth and Marilda Sotomayer (1990) has become a standard reference for other researchers.

In recent years, Bettina Klaus and Flip Klijn have worked extensively on the problems of median matchings and fair matchings, investigating fairness from both a procedural perspective and a justice perspective.

1.4 Today – Law Schools and Judges

The improvements made in the college admissions process have not been adopted by all institutions and we still have the model of 'application frenzy' in today's world.

A notable example is that of the law schools whose graduating students are pre-selected by judges with an impossibly short deadline given to 'choose or lose'.
An article in the New York Times of March 17th 1989 made the following observations: The once decorous process by which federal judges select their law clerks has degenerated into a free-for-all in which some of the judges scramble for the top law school students. The judges have increased the pressure on the hiring process, offering some jobs as early as February of the second year of law school, at which time there are fewer grades to support 'best choice'.

The association of American Law Schools agreed not to hire before September of the third year, but this is leading to a practice known in the clerkship vernacular as a "short fuse" or a "hold up", whereby offers are extended for only a few hours.
One judge offered a Yale student a clerkship at 11:35 and gave her until noon to accept ... At 11:55 he withdrew his offer!

2 Chapter 2 - Concepts and Definitions

2.1 Introduction

A *matching* between two input sets of the same size is one output set of unordered pairs. In each pair, one element comes from each input set, and each element appears exactly once in a pair. In graph theoretical terms, this would be seen as a *perfect* matching between the nodes of two sub-graphs. More informally, such a matching can be seen as a “marriage” matching between a set of men and a set of women.

NOTATION

Before proceeding, we introduce some notation that will be used throughout this paper:

n is the cardinality, the size of the set(s) to be matched

m is the male variable; *w* is the female variable

In the examples, specific males are denoted by *A*,*B*,*C*,*D*,…..

and specific females are denoted by *a*,*b*,*c*,*d*,…….

M is the matching, the output set of unordered pairs

$p_M(w)$ is the partner of w in M

$p_M(m)$ is the partner of m in M

Analogously, in the college admissions problem we use:

c is the college variable; *s* is the student variable (ref. 2.3.1 and 5.1)

and in the hospital-residents problem we use:

h is the hospital variable; *r* is the resident variable (ref. 2.3.1 and 5.1)

Further notation will be introduced as we proceed.

The algorithms presented in this paper are set out in readable pseudo-code, as is often used in computer science to define algorithms that are implementation independent.

2.2 The Stable Marriage Problem

2.2.1 Stable Marriage - Definition

Having defined a "matching" in general terms in the introduction, it is now necessary to define a *stable* matching, which requires the notion of "preference" in the association of pair elements. A stable marriage is a representation of a stable matching, and a definition of the stable marriage problem may be given, thus:

Let there be *n* men and *n* women, each of whom has ranked all members of the opposite sex in order of preference from 1 to *n*, with preference lists of the form $\{w_1, w_2, ...w_n\}$ or $\{m_1, m_2, ...m_n\}$. If *n* marriage couples are formed, such that there are **no** pairs of the form $\{m_i, w_j\}$, $\{m_k, w_t\}$ where m_i prefers w_t to w_j **and** w_t prefers m_i to m_k then the set of marriages is said to be stable. Otherwise the marriages are unstable, and $\{m_i, w_t\}$ is said to be a **"blocking pair"**.

2.2.2 Stable Marriage – Example 2.1

We consider marrying 4 men (*A, B, C* and *D*) to 4 women (*a, b,c* and *d*).

Men	Order of Preference			
A	*c*	*b*	*d*	*a*
B	*b*	*a*	*c*	*d*
C	*b*	*d*	*a*	*c*
D	*c*	*a*	*d*	*b*

Women	Order of Preference			
a	*A*	*B*	*D*	*C*
b	*C*	*A*	*D*	*B*
c	*C*	*B*	*D*	*A*
d	*B*	*A*	*C*	*D*

The matching (*Aa, Bb, Cc, Dd*) is unstable because *A* and *b* prefer each other to their assigned partners. So *Ab* is a blocking pair. On the other hand, the matching (*Ad, Ba, Cb, Dc*) is stable since it does not contain any blocking pairs.

2.2.3 Stability Checking

Checking for stability is a systematic process whereby each member, of one sex only, is considered in turn. If the member is partnered with someone other than their first choice then the preferred choices, plus *their* partners, will be considered to see if a blocking pair is revealed. In the above example, if stability checking is being done from the male perspective, then couple *Ad* would lead to checking whether *c* prefers *A* to *D*, or *b* prefers *A* to *C*. If either condition were true then the matching would be unstable.

<u>A Stability Checking Algorithm</u>

```
for m = 1 to n do
        for each w that m prefers to  p_M(m)  do
                if w prefers m to  p_M(w)  then
                begin
                        report matching unstable
                        halt
                end
report matching stable
```

2.2.4 Stable Marriage - Existence

Gale and Shapley (1962) proved that at least one stable marriage exists for any choice of preference rankings. This is a rather surprising result since it is by no means intuitive. Depending on the preference lists there may be several stable marriage sets to be had from the same problem instance, but there will always be at least one, irrespective of preferences. (We show the proof of this in Section 2.3.3).

It is, perhaps, also surprising that *many, different,* stable matchings can be derived from one instance of preferences. We use an example to demonstrate multiple solutions.

Example 2.2

Assume that there are *n* men and *n* women, and that *n* is an even number. The dots (**. . . . .**) represent unspecified men or women in the lists. The only relevant entries for our purpose are the first two entries on the men's lists and the last entry on the women's lists.

Men	**Preferences**
A	$a\ b\ \ldots\ldots\ldots\ x_{n-1}\ x_n$
B	$b\ a\ \ldots\ldots\ldots\ x_n$
C	$c\ d\ \ldots\ldots\ldots\ x_n$
D	$d\ c\ \ldots\ldots\ldots\ x_n$
E	$e\ f\ \ldots\ldots\ldots\ x_n$
F	$f\ e\ \ldots\ldots\ldots\ x_n$
.	. . . .
.	. . . .
X_n	. . . .

Women	**Preferences**
a	$\ldots\ldots\ldots\ldots\ A$
b	$\ldots\ldots\ldots\ldots\ B$
c	$\ldots\ldots\ldots\ldots\ C$
d	$\ldots\ldots\ldots\ldots\ D$
e	$\ldots\ldots\ldots\ldots\ E$
f	$\ldots\ldots\ldots\ldots\ F$
.	. . . .
.	. . . .
x_n	. . . .

Suppose that each of the $n/2$ *pairs* of men *AB, CD, EF,* . . . . *both* marry their first choice, or, *both* marry their second choice, thus forming couples *{Aa, Bb}* or *{Ab, Ba}*. Each matching so obtained is stable, because:

i. a man marrying his first choice cannot be part of a blocking pair; by definition, he prefers no one else,
ii. each man is the *least* preferred man of his first choice woman, so she cannot form a blocking pair with him if he partners his second choice.

Since there are $n/2$ male pairs, and there are two options for each pair, it follows that $2^{n/2} = \left(\sqrt{2}\right)^n$ stable matchings can be obtained in this way.

2.2.5 Average and Maximum Numbers of Stable Matchings

The average number of stable matchings taken over all instances of size *n* is a function *g(n)* that is asymptotic to $e^{-1} n \ln n$. That is to say $\lim_{n \to \infty} \left(\frac{g(n)}{e^{-1} n \ln n} \right) = 1$, equivalently the percentage error → 0 for large *n*. The proof involves some advanced probability theory and is omitted from this paper. (Refer to Boris Pittel (1989) for the proof).

The function *g(n)* grows much more slowly than the corresponding function *f(n)* that gives the largest possible number of stable matchings from an instance of size *n*. It is known that the function *f(n)* has an exponential growth rate; in fact an exponential lower bound of $\left(\sqrt{2}\right)^n$ was demonstrated in 2.2.4.

The upper bound for the maximum number of stable matchings is still an open problem, but another lower bound for *f(n)* that establishes the exponential growth rate can be given, thus (Gusfield and Irving, 1989):

<u>Theorem 2.2.5 – a lower bound for maximum stable matchings</u>

Given stable marriage instances of sizes *p* and *q* with *x* and *y* stable matchings respectively, then there exists an instance of size *n*, $n = pq$, where the number of stable matchings is $\geq \max(xy^p, yx^q)$.

<u>Proof</u>

Let the men in the two original instances be labelled $A_1, A_2, ..., A_p$ and $B_1, B_2, ..., B_q$ and let the women be labelled $c_1, c_2, ..., c_p$ and $d_1, d_2, ..., d_q$.

We can construct an instance of size *pq* by repeating the *p*-size instance *q* times, and then differentiating the men/women by the particular repetition in which they occur. We shall call each repetition a "component instance". Therefore there are *pq* men, each labelled $(A_i, B_j), i = 1, ..., p, j = 1, ..., q$; and there are *pq* women, each labelled $(c_i, d_j), i = 1, ..., p, j = 1, ..., q$. The men's preferences from the original *q*-size instance are brought forward to the composite instance, so that for instance if man B_j originally preferred d_7 to

d_4 then, in the composite instance, all men in the B_j men's component prefer each women in the 7th women's component to every women in the 4th women's component. Women in the *same* component are compared according to the preferences of the original *p*-size instance. Formally, man (A_i, B_j) prefers woman (c_k, d_l) to woman $(c_{k'}, d_{l'})$ if B_j prefers d_l to $d_{l'}$, or if $l = l'$ and A_i prefers c_k to $c_{k'}$; similarly, woman (c_i, d_j) prefers (A_k, B_l) to $(A_{k'}, B_{l'})$ if d_j prefers B_l to $B_{l'}$, or if $l = l'$ and c_i prefers A_k to $A_{k'}$.

Having constructed a composite instance, we can lift the stable matchings from the original instances to the composite instance, thus:
let M_q be a stable matching of the original *q*-size instance, for which there are *y* solutions, and let $M_{p_1}, \ldots, M_{p_q}$ be *q* stable matchings (repetitions allowed) of the original *p*-size instance. Since there are *x* such stable matchings, there are x^q ways of choosing these *q* stable matchings with repetitions.

For the set of men $(A_i, B_j), i = 1, \ldots, p, j = 1, \ldots, q$, we can match (A_i, B_j) to $(p_{M_{p_j}}(A_i), p_{M_q}(B_j))$; this must be a matching because both M_q and M_{p_j} are matchings.
To show that it is stable we consider the conditions for a blocking pair $((A_i,B_j),(c,d))$: Either B_j prefers *d* to $p_{M_q}(B_j)$ and *d* prefers B_j to $p_{M_q}(d)$ or $d = p_{M_q}(B_j)$ and A_i prefers *c* to $p_{M_{p_j}}(A_i)$ and *c* prefers A_i to $p_{M_{p_j}}(c)$. The first condition is precluded by the stability of M_q, and the second condition is precluded by the stability of M_{p_j}.
Therefore the matchings so constructed on the composite instance are stable.

Recalling that there are *y* solutions for M_q, and *x* solutions for M_p that can be chosen with replacement for each of q component instances, we have constructed a composite instance with at least yx^q stable matchings.

By reversing the roles of the *p*-size and *q*-size original instances, we can construct another composite instance of size *pq* that has at least xy^p stable matchings.

Hence, our composite instance of size *pq* has at least $\max(xy^p, yx^q)$ stable matchings.

Corollary 2.2.5

For values of $n \geq 1$, *n* a power of 2, there is a stable matching instance of size *n* that has $\geq 2^{n-1}$ stable matchings.

Proof

We use induction on k, $n = 2^k$, together with the results of our theorem 2.2.5. The result is true for $k = 0$, $n = 1$, as the trivial instance has 1 stable matching. Assuming there exists a stable matching instance of size $n = 2^k$ that has at least $2^{n-1} = 2^{2^k-1}$ stable matchings, we will then show that there must exist a stable matching instance of size $n = 2^{k+1}$ which has at least $2^{n-1} = 2^{2^{k+1}-1}$ stable matchings.

We introduce a fixed instance of size 2 that has 2 stable matchings, thus:

Men		
A	*a*	*b*
B	*b*	*a*

Women		
a	*B*	*A*
b	*A*	*B*

There are two possible matchings, {(*Aa*),(*Bb*)} and {(*Ab*),(*Ba*)} and by inspection we see that both are stable.

Applying Theorem 2.2.5 using (our fixed example) $p = 2$ with $x = 2$, and (our assumption case) $q = 2^k$ with $y = 2^{2^k-1}$, we have that there must exist a composite instance of size $pq = 2.\ 2^k = 2^{k+1}$ that has $\geq \max(xy^p, yx^q) =$ $\max(2.(2^{2^k-1})^2, 2^{2^k-1}.2^{2^k}) = \max(2^{2^{k+1}-1}, 2^{2^{k+1}-1})$ stable matchings, which was required to prove.

We comment that this lower bound of 2^{n-1} for the maximum number of stable matchings from an instance of size *n* is better than the lower bound of $\left(\sqrt{2}\right)^n$ that was obtained earlier (if *n* is a power of 2).

As an illustration, for *n* = 32, the maximum number of stable matchings is greater than $2^{31} > 2 \text{ x } 10^9$.

2.2.6 For Better, For Worse …..

Example 2.2 has revealed another counter-intuitive aspect; it is quite possible, in a stable matching, for all the partners of one sex to be partnered with their worst choice. Consider the following:

Example 2.3

Men	**Order of Preference**			
A	*d*	*b*	*c*	*a*
B	*d*	*a*	*c*	*b*
C	*d*	*b*	*a*	*c*
D	*a*	*b*	*c*	*d*

Women	**Order of Preference**			
a	*A*	*B*	*C*	*D*
b	*B*	*C*	*D*	*A*
c	*C*	*D*	*A*	*B*
d	*D*	*A*	*B*	*C*

then the matching *(Aa, Bb, Cc, Dd)* is stable, even though every man is getting his least preferred woman, because the women are getting their first choice men and would not prefer anyone else. We shall see in section 3.1.2 that this is not atypical.

It is even possible for a stable matching to contain a pair in which both partners have their worst choice. Thus:

Example 2.4

Men	**Order of Preference**		
A	*a*	*b*	*c*
B	*a*	*b*	*c*

Women	**Order of Preference**		
a	*B*	*C*	*A*
b	*B*	*C*	*A*

C	*a*	*b*	*c*	*c*	*B*	*C*	*A*

This instance yields a stable matching *(Ac, Ba, Cb)* and yet *Ac* is a mutually worst choice pair.

2.3 Algorithms for Deriving a Stable Marriage

2.3.1 College Admissions – the Analogy to Stable Marriage

When Gale and Shapley (1962) published their groundbreaking paper their main focus was the "college admissions" problem, in which each college has a quota of places to offer preferred students and the students have ranking lists of preferred colleges. The two sets to be matched (college places and students) are typically of unequal size, and the matching is therefore partial. The definition of stability is therefore re-stated to deal with unequal sized sets, and the one (college) to many (students) situation.

Stable matching, one-to-many, unequal sized sets

A matching *M* between a set of colleges $\{c_1, c_2, ...\}$ and a set of students $\{s_1, s_2, ...\}$ is unstable if for some college *c* and student *s* the following three conditions hold:

1. *c* and *s* are not assigned to each other in *M*
2. *c* is either below quota, or prefers *s* to at least one of its assigned students
3. *s* is either unassigned, or prefers *c* to its assigned college

The college admissions problem can be seen as a polygamous (polyandrous) version of the stable marriage problem. Alternatively, the stable marriage problem can be seen as an instance of the college admissions problem in which each college has only one place to offer – a quota of unity – and there are an equal number of students and colleges.

We note that the "hospital-residents problem" is exactly the same problem mathematically as the college admissions problem. It carries a different name purely because of the different social application. We return to these "one-to-many" matchings in section 5.1.

2.3.2 The Gale and Shapley Algorithm

To address the college admissions problem, Gale and Shapley (1962) described an iterative procedure for deriving a stable marriage, not as an algorithm in pseudo code but in ordinary language without symbols. We paraphrase it here.

Stage 1: Each boy makes one proposal to his favourite girl. Any girl who has more than one proposal rejects all but the best one.

Stage 2: Any boy who has been rejected makes a proposal to the next girl on his list. Any girl who has more than one proposal (including one from stage 1) rejects all but the best proposal so far.

Stage 2 is repeated until the last girl receives a proposal. This is sure to happen at some stage because whilst *n* boys are proposing to *less* than *n* girls there must be rejections and new proposals. Eventually, the *n*-th girl will receive a first proposal.

End: When this happens, the "courtship" process is over and each girl accepts the best proposal she has received during the process.

Minimum and maximum stages

It is possible for a stable matching to be achieved in one round of proposals, if each boy's first choice is a different girl. Gale and Shapley note that the maximum number of stages is $n^2 - 2n + 2$, and this will be explored in Section 2.4.

We note that Gale and Shapley's algorithmic procedure has some degree of parallelism, in that a girl considers multiple proposals in one stage, and there may be many "rejected" men proceeding to their next choice at the same time.

2.3.3 Knuth's Algorithm for a Stable Marriage

By contrast, Knuth (1976) sets out an algorithm in Algol-style pseudo code that also derives a stable matching between two equal-sized sets of men and women.

It should be noted that Knuth's (1976) algorithm will yield exactly the same result as Gale and Shapley's (1962) iterative procedure. However, the sequence followed is somewhat different in that Knuth (1976) processes one man at a time, and only allows for one man to be in a rejected state at any time.

A strong feature of both procedures is the principal of "deferred acceptance" whereby each girl/woman keeps her options open in the hope that a more favourable proposal may happen. This is necessary to ensure stability.

We re-state Knuth's (1976) algorithm here, in our own words. For ease of reference we have given each statement a line number, and we have indicated the processing loops within the algorithm by means of a shaded side bar.

The algorithm uses three variables: *k, X* and *x,* and two constants *n* and Ω.

n : number of men, number of women

k : number of provisionally engaged couples

X : the man who will make the next proposal

x : the woman to whom a proposal is being made

Ω: an extra, imaginary man that all women like less than any of the real men, and to whom all the women are initially "engaged"

<u>Knuth's algorithm to derive a stable marriage</u>

1 $k \leftarrow 0$;
2 while $k < n$ do
3   begin $X \leftarrow (k + 1)$-st man ;
4   while $X \neq \Omega$ do
5   begin $x \leftarrow$ best choice remaining on X's list ;
6   if x prefers X to her fiancé then
7   begin engage X and x ;
8 $X \leftarrow$ preceding fiancé of x
9   end ;
10   if $X \neq \Omega$ then withdraw x from X's list
11   end ;
12 $k \leftarrow k + 1$
13   end ;
14 celebrate n weddings

couple/decouple (lines 6–9)

inner loop – repeat until a new woman is asked for the first time (lines 4–11)

outer loop – repeat until the *n*-th man is selected (lines 2–13)

Knuth (1976) uses this algorithm to give an independent proof of Gale and Shapley's (1962) theorem for the existence of a stable matching between any two same-sized sets with any, complete, preference lists.

We summarise the proof into two parts, a) and b):

a) the algorithm produces a matching
b) the matching produced is stable

Proof of a)

Statements #7 and #8 are either both obeyed or neither obeyed, subject to #6. Therefore a woman cannot become engaged to a (new) man and remain engaged to her (old) man. Therefore no woman will appear more than once in the final set of couples.

The inner loop, statements #4 through #11, is performed until $X = \Omega$; in other words, it is performed until a previously un-engaged woman gets engaged for the first time. This must always happen because no woman can appear in more than one engaged pair (above) and the introduction of a new man in line #3 ensures that a new woman must eventually have a first proposal in line #5.

The outer loop, statements #2 through #13, is performed *n* times, once for each man. This is enforced by #2 and #12, which increment *k* by 1 each time that the loop is performed, and stops the loop when *n* men have been processed.

Therefore, each time the outer loop is performed, a 'new' man is selected and the inner loop is performed until a 'new' woman is engaged. Thus *n* different men are engaged to *n* different women. Therefore there is a matching when the algorithm terminates.

Proof of b)

We note that with each performance of the inner loop beginning at line #5 the proposer selects the next candidate fiancée from his list, strictly in order of preference. Suppose, to the contrary, that the algorithm yields a result set that is **not** a stable matching. Then the result set contains two couples *Ab* and *Ca* such that *Aa* is a blocking pair. That is, *A* and *a* mutually prefer each other to their assigned partners.

Therefore, *A* prefers *a* to *b*. It follows that *A* must have proposed to *a* before proposing to *b* (Line #5). Hence *a* must have rejected *A* in favour of her existing fiancé, or a later proposer *X*. But the rejection process entails that *a* prefers *C* to *A*.

So *A* and *a* do **not** mutually prefer each other to their assigned partners, which contradicts the assumption. Therefore *Aa* is **not** a blocking pair, and the matching produced by the algorithm is stable.

2.3.4 Gale and Shapley vs. Knuth

A comparison between the two procedures is interesting in terms of:

1. theorem proving – using an algorithm to prove a theorem is more established in computer science but is relatively recent in mathematics
2. complexity – in this context we mean "time complexity", measuring the workload inherent in the algorithm, and therefore the computability of the algorithm, and the likelihood of computing a stable matching in polynomial time
3. non-determinism – the fact that the sequence in which men propose does not affect the outcome

2.3.5 The Extended Algorithm

An extended algorithm is now described where the same procedure is used for proposing, but every time an acceptance happens the opportunity is taken to remove redundant entries from the preference lists. In a man-oriented extended algorithm, when *m* proposes to *w* and is accepted, then we know that *m* cannot do better than *w*, so any w' that precedes *w* in *m*'s list is removed from his list, and *m* is removed from the lists of all w'. Similarly, we know that w cannot do worse than m, so any m' that follows m in w's list is removed from her list, and w is removed from the lists of all m'.

One consequence of this approach is that a proposal is always accepted, because w only keeps on her list the men that are better/equal to her current fiancé.

The author here presents her own extended algorithm. The differences from the standard Knuth algorithm previously shown in 2.3.3 are indicated by highlighting the modifications.

Extended algorithm to derive a stable marriage

1 $k \leftarrow 0$;

2 while $k < n$ do

3 begin $X \leftarrow (k + 1)$-st man ;

4 while $X \neq \Omega$ do

5 begin $x \leftarrow$ best choice remaining on X's list ;

line 6 removed, the proposal will always be accepted

6 if x prefers X to her fiancé then

7 begin engage X and x ;

insert process to reduce the lists during the course of the algorithm

7a **begin for each successor X' of X in x's list**

7b **remove X' from x's list and**

7c **remove x from X''s list**

7d **end**

7e

8 $X \leftarrow$ preceding fiancé of x

9 end ;

line 10 removed, withdrawal of x from X's list is already done in new lines 7a-e

10 if $X \neq \Omega$ then withdraw x from X's list

11 end ;

12 $k \leftarrow k + 1$

13 end ;

14 celebrate n weddings

couple/decouple

inner loop – repeat until a new woman is asked for the first time

outer loop – repeat until the n-th man is selected

The benefit of this exercise is not just one of optimising the algorithm, although that does result. The additional benefit is that we can reduce the preference lists to explore all possible stable matchings, and not just the optimal matching produced by the algorithm.

2.3.6 The Reduced Preference Lists

When the extended algorithm halts, each agent (man or woman) has a preference list that has been reduced by the deletion of (some of the) impossible partners.

We follow the notation of Gusfield and Irving (1989) in referring to the reduced preference lists arising from the extended algorithm with men as proposers as the *man-oriented Gale-Shapley lists,* or *MGS-lists* for short; if the algorithm were run with women as proposers the reduced lists would be different and we say that the women-oriented algorithm will produce the *WGS-lists.*

Further, if for each person we take the intersection of his/her MGS-list and WGS-list, then the final reduction is the *GS-list* of that person. It follows from the rules for deriving the GS-lists that no man (or woman) can be partnered in any stable matching with a partner who is *not* in his/her GS-list. However it is possible for the GS-lists to contain a man/woman that cannot be partnered in any stable matching. The GS-lists are a necessary condition for stability, but not a sufficient one.

One way of obtaining the GS-lists is to apply the extended algorithm with men as proposers, and then use the MGS-lists so obtained to run the algorithm again with women as proposers.

The GS-lists can be very revealing, and also vary considerably per instance in the degree of reduction. For instance, in the case where there is only one stable matching, then the man-optimal and woman-optimal matchings coincide, and the GS-lists will contain only one entry per person. Alternatively, if the man-optimal solution yields a first choice for every man and a last choice for every woman and the woman-optimal solution yields a first choice for every woman and a last choice for every man, then the GS-lists will not be reduced at all, they will be full lists as per the initial preferences.

An example GS-list is given in Gusfield and Irving (1989) based on an instance of size 8. The author here provides the detailed processing and derivation of this example by showing the entries that will be deleted during the man-optimal algorithm, highlighted thus: X , and the ones deleted during the consequent woman-optimal algorithm, highlighted thus X. The initial preference lists with consequent deletions highlighted are:

Men	Order of Preference							
A	*e*	*g*	*a*	*b*	*f*	*h*	*d*	*c*
B	*b*	*c*	*g*	*e*	*d*	*a*	*h*	*f*
C	*h*	*e*	*a*	*d*	*f*	*b*	*c*	*g*
D	*c*	*b*	*g*	*d*	*a*	*f*	*h*	*e*
E	*g*	*b*	*e*	*a*	*c*	*f*	*h*	*d*
F	*a*	*f*	*g*	*e*	*h*	*d*	*b*	*c*
G	*b*	*e*	*g*	*f*	*c*	*d*	*h*	*a*
H	*c*	*h*	*d*	*e*	*g*	*b*	*f*	*a*

Women	Order of Preference							
a	*E*	*C*	*G*	*F*	*A*	*B*	*H*	*D*
b	*H*	*F*	*C*	*E*	*G*	*B*	*A*	*D*
c	*A*	*E*	*F*	*B*	*D*	*H*	*G*	*C*
d	*H*	*G*	*C*	*B*	*D*	*A*	*E*	*F*
e	*F*	*D*	*G*	*C*	*H*	*A*	*B*	*E*
f	*B*	*H*	*E*	*C*	*D*	*F*	*G*	*A*
g	*G*	*E*	*B*	*A*	*H*	*F*	*D*	*C*
h	*G*	*D*	*A*	*E*	*B*	*C*	*F*	*H*

The residual GS-lists are then revealed to be as follows:

Men	GS-List				
A	*e*	*h*	*c*		
B	*c*	*h*	*f*		
C	*h*	*e*	*a*	*f*	*b*
D	*f*	*h*			
E	*g*	*b*	*a*		
F	*a*	*e*			
G	*b*	*e*	*g*		
H	*d*				

Women	GS-List			
a	*E*	*C*	*F*	
b	*C*	*E*	*G*	
c	*A*	*B*		
d	*H*			
e	*F*	*G*	*C*	*A*
f	*B*	*C*	*D*	
g	*G*	*E*		
h	*D*	*A*	*B*	*C*

We can summarise the properties of GS-lists in the following Theorem.

Theorem 2.3.6 - Properties of GS-lists

For a given instance of a stable matching problem:

1. all stable matchings are contained in the GS-lists
2. no matching contained in the GS-lists can be blocked by a pair that is not in the GS-lists
3. in the man-proposing (woman-proposing) stable matching, each man is partnered by the first (last) woman on his GS-list, and each woman by the last (first) man on hers.

The proofs of these properties are deferred to section 3.1.3 so that we can use the principle of optimality that is explored in section 3.1

In section 4.2 we return to the topic of GS-lists and show how they can be used to investigate the solution set of stable matchings from a given instance.

2.4 Worst Case Complexity

2.4.1 Comparing Maximum Stages with Maximum Proposals

The time complexity of the algorithms have a direct correspondence to the amount of work entailed in deriving a stable matching, expressed as the number of iterations of some core component in the procedure or algorithm. It is convenient to express this in terms of *n*, the size of the sets being matched.

Knuth (1976) gives a worst case for the number of proposals (using his algorithm) of $n^2 - n + 1$. Gale and Shapley (1962) state that the worst case for the number of stages in their iterative procedure is $n^2 - 2n + 2$.

The apparent discrepancy is not clarified in Gusfield and Irving (1989). The author now shows that it can be explained when it is understood that Knuth is measuring 'proposals' and Gale and Shapley are measuring 'stages' – in which there may be more than one proposal.

Proposals

Matching stops when the *n*-th woman receives her first proposal. The worst case for the man who is proposing to the *n*-th woman is that he is proposing to the last woman on his list. Only one man can make proposals to all *n* women. The other (*n* – 1) men can make as many as (*n* – 1) proposals each, which means that they obtain their next-to-last choices. Therefore, the worst case for the number of proposals is $n + (n-1)^2$, which is $n^2 - n + 1$. This applies to both the Gale and Shapley (1962) iterative procedure and the Knuth (1976) algorithm.

Stages

However, if we evaluate the workload of Gale and Shapley's (1962) iterative procedure in terms of stages, we see that the parallel nature of the procedure allows for more than one proposal to be made in one stage. In fact, each stage has as many proposals as there are free men (at that time).

So the first stage has, of necessity, *n* proposals – each man making his first proposal to his first choice. Thereafter, the following stages may have as little as one proposal each, indicating only one rejected / free man who is compelled to drop down his preference list and propose to the next woman.

Therefore, if the maximum number of proposals is stretched out to spread over the maximum number of stages, we have:-

stage 1 - n proposals

stage 2 onwards - 1 proposal each

After the first stage, (n – 1) of the men may make another (n – 2) proposals and one of the men may make another (n – 1) proposals.

Thus, the total number of stages $= 1 + (n-1)(n-2) + (n-1)$

$$= 1 + n^2 - 3n + 2 + n - 1$$

$$= n^2 - 2n + 2 .$$

We could also say that the worst case for the number of stages corresponds to the worst case for the number of proposals by adjusting for stage 1, by subtracting n proposals and adding 1 stage.

Thus $(n^2 - n + 1)$ ………… max proposals

$(-n + 1)$ ………… adjustment for stage 1

$= n^2 - 2n + 2$ ………… max stages.

2.4.2 Author's Example for maximum stages

An example for maximum stages in the Gale and Shapley procedure is given, for n = 4, as follows: [The numeric suffix after each woman in the men's preference lists indicates the stage in which the proposal is made.]

Men	**Order of Preference**			
A	*a.1*	*c.4*	*b.8*	*d*
B	*b.1*	*a.3*	*c.7*	*d*
C	*c.1*	*b.2*	*a.6*	*d.10*
D	*c.1*	*b.5*	*a.9*	*d*

Women	**Order of Preference**			
a	*D*	*C*	*B*	*A*
b	*A*	*D*	*C*	*B*
c	*B*	*A*	*D*	*C*
d	.	.	.	.

Thus, for n = 4, we have maximum number of stages $= n^2 - 2n + 2$ = 10, and we see that man C proposes to the last woman d in the tenth stage.

We can generalise the case for maximum stages to arbitrary n as follows:

First stage : $n - 2$ of the men propose to $n - 2$ different women, and the other two men (call them *A* and *B*) propose to the same woman.

The only contention is between *A* and *B*, so, depending on the women's preferences, either *A* or *B* will have to make a second proposal in the second stage.

Second stage : Either *A* or *B* proposes to his second choice, which must be one of the $n - 2$ women already proposed to in the first stage. The existing fiancé, say *C*, is rejected and has to propose to his next choice in the next stage.

Next and subsequent stages : Proposals continue as per the second stage, always with the existing fiancé being rejected in favour of the new proposer, until all $n - 1$ women have been proposed to by all n men.

Final stage : the last fiancé to be rejected is compelled to propose to the n-th woman.

2.4.3 Author's Example for maximum proposals

An example for the maximum proposals being required in the Knuth algorithm is given, for $n = 4$, as follows: [The numeric suffix after each woman in the men's preference lists indicates the proposal sequence.]

Men	Order of Preference			
A	*a.1*	*c.7*	*b.11*	*d*
B	*b.2*	*a.6*	*c.10*	*d*
C	*c.3*	*b.5*	*a.9*	*d.13*
D	*c.4*	*b.8*	*a.12*	*d*

Women	Order of Preference			
a	*D*	*C*	*B*	*A*
b	*A*	*D*	*C*	*B*
c	*B*	*A*	*D*	*C*
d	.	.	.	.

Thus, for $n = 4$, we have maximum number of proposals $= n^2 - n + 1 = 13$, and we see that man *C* proposes to the last woman d with the thirteenth proposal.

2.4.4 General Example for Maximum Stages and / or Proposals

The author constructs below an instance of generalised preference lists to demonstrate both cases of maximum stages and maximum proposals, for an arbitrary problem size *n*. We note that:

1. the **.** represents an unspecified man/woman; the tables could be expanded by inserting more men/women between *H/h* and *X/x*.
2. in Gale and Shapley terms, the first column of the men's table represents the first *stage*, and thereafter there is one stage for each proposal, for rows from 1 to *n* and columns from 2 to *n* – 1, before a final stage for the proposal by the first man to his n^{th} woman .
3. in terms of Knuth's algorithm, the first column of the men's table represents *n* proposals, and thereafter it is the same as Gale and Shapley, one stage = one proposal.
4. in principle, in the j^{th} column of proposals each man before *X* makes one accepted proposal to the j^{th} woman on his list, each man to a different woman; then *X* upsets things by proposing to the same woman as *A*, which starts another column of proposals.
5. the woman *x* is last choice for every man and so the matching procedure will only stop when *A* proposes to *x*.

Eventually, there are $n^2 - n + 1$ proposals and $n^2 - 2n + 2$ stages.

Men	**Order of Preference**									
A	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	.	*x*
B	*b*	*c*	*d*	*e*	*f*	*g*	*h*	.	*a*	*x*
C	*c*	*d*	*e*	*f*	*g*	*h*	.	*a*	*b*	*x*
D	*d*	*e*	*f*	*g*	*h*	.	*a*	*b*	*c*	*x*
E	*e*	*f*	*g*	*h*	.	*a*	*b*	*c*	*d*	*x*
F	*f*	*g*	*h*	.	*a*	*b*	*c*	*d*	*e*	*x*
G	*g*	*h*	.	*a*	*b*	*c*	*d*	*e*	*f*	*x*
H	*h*	.	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*x*
.	.	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*x*
X	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	.	*x*

Women	**Order of Preference**									
a	*B*	*C*	*D*	*E*	*F*	*G*	*H*	.	*X*	*A*
b	*C*	*D*	*E*	*F*	*G*	*H*	.	*X*	*A*	*B*
c	*D*	*E*	*F*	*G*	*H*	.	*X*	*A*	*B*	*C*
d	*E*	*F*	*G*	*H*	.	*X*	*A*	*B*	*C*	*D*
e	*F*	*G*	*H*	.	*X*	*A*	*B*	*C*	*D*	*E*
f	*G*	*H*	.	*X*	*A*	*B*	*C*	*D*	*E*	*F*
g	*H*	.	*X*	*A*	*B*	*C*	*D*	*E*	*F*	*G*
h	.	*X*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
.	*X*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	.
x	.	.	.	.	.	.	.	.	.	.

2.4.5 Algorithmic Complexity – Upper Bound

In making the above observations with respect to complexity in terms of both 'stages' and 'proposals' we note that Gale and Shapley (1962) state the formula for maximum stages without proof, and that Gusfield and Irving (1989) bypass both proofs in pursuit of merely an $\boldsymbol{O}(n^2)$ upper bound for proposals.

The algorithm may be efficiently implemented by defining arrays *mp* and *wp* of the preference lists where

$mp[m,i] = w \Leftrightarrow$ woman w is in position i in the list of man m ;

$wp[w,i] = m \Leftrightarrow$ man m is in position i in the list of woman w .

Further, for a quick assessment of whether w prefers m to m', two ranking arrays can be constructed from the preference arrays *mp* and *wp*, thus:

$wr[w,m] = i \Leftrightarrow$ man m occupies position i in the list of woman w ;

$mr[m,w] = i \Leftrightarrow$ woman w occupies position i in the list of man m .

The preference and ranking arrays can be constructed from the preference lists in $\boldsymbol{O}(n^2)$ time at the start of the algorithm.

During the course of the algorithm the number of operations is bounded by a constant times the number of proposals, which we have shown has a maximum of $(n^2 - n + 1)$. So the complexity for the standard algorithm is $\boldsymbol{O}(n^2)$.

The complexity for the extended algorithm is still of the same order. In the worst case, no reduction of preference lists will take place. In other cases, the extra work involved in changing the arrays is offset by the reduced number of proposals, and the fact that a proposal is always accepted. Either way, the work to be done on the main loop of the extended algorithm still correlates to the number of proposals, so the upper bound is still $\boldsymbol{O}(n^2)$.

3 Chapter 3 - Some Theorems Re-Proved

3.1 Optimality and Pessimality

3.1.1 The Man Optimal Solution

The algorithms of Gale and Shapley (1962) and Knuth (1976) both have an element of non-determinism in that the choice of 'next man' is arbitrary. However, irrespective of the sequence in which the proposers (men) are considered, the algorithms give the same result: a stable matching in which every man has the best partner that he can have in any stable matching from this instance of preferences. Thus, the stable matching that is produced by the algorithm is called the "man-optimal" stable matching, denoted by M_0. It also clearly follows that if the roles of the men and women are switched so that the women are the proposers, the algorithm would yield the "woman-optimal" stable matching, W_0.

The proposer-optimal property of the "deferred acceptance" algorithm was first stated as a theorem by Gale and Shapley (1962, page 14, theorem 2) for the college admissions (one to many) version of the algorithm. An outline of the proof by induction was given.

Both Knuth (1976) and Gusfield & Irving (1989) offer proof sketches by contradiction for the marriage (one to one) version, mentioning induction on the algorithm, but leaving out details of the induction.

The author here provides her own detailed proof, using infinite descent.

<u>**Theorem 3.1.1**</u>

The Algorithm Produces the Man-Optimal Solution

<u>Proof</u>

Let *M* be the matching produced by the algorithm.

We define a *rejection pair* to be a pair (*m*,*w*) such that *w* rejects *m* in the course of the algorithm.

Let *S* = $\{(m, w): w$ rejects *m* in the course of the algorithm, yet (*m*,*w*) occurs as a pair in some other stable matching from this instance }

The set S can be linearly ordered by the sequence of rejections in the algorithm, and supposing that $S \neq \emptyset$, we take (m,w) to be the earliest rejection pair in S. Let M' be a stable matching that contains the pair (m,w).
Therefore, w rejected m in the course of the algorithm, and at some stage during the algorithm w is engaged to m', where m' is the man for whom w rejected m. We know, then, that in w's preference list the man m' must appear before the man m.
That is, w's preference list looks like $[\, w : \ldots\ldots m' \ldots m \ldots]$ where the ellipses $(\ldots)$ represent none, one or many instances of other men.

Let m' be matched to w', say, in M'. M' is stable, therefore m' prefers w' to w, otherwise (m', w) would be a blocking pair. (See 2.2.1) Therefore, the preference list for m' must look like $[\, m' : \ldots\ldots w' \ldots w \ldots]$.

If we consider these two preference lists in terms of the algorithm, it means that m' proposed to w', and was rejected, ***before*** m' became engaged to w.
But we also know that m' became engaged to w ***before*** w rejected m.
Since (m', w') is a pair in the stable matching M', and (m', w') is a rejection pair in the algorithm, we have that (m', w') must occur in the set S. But we have also shown that w' rejected m' before w rejected m, which contradicts our assumption that (m,w) is the earliest rejection pair.
Therefore $S = \emptyset$, and there can be no stable matching that gives m a partner that rejected him in the algorithm.
Therefore the algorithm produces the man-optimal solution.

3.1.2 Man Optimal = Woman Pessimal

Having established that the algorithm produces a stable matching that is proposer/man optimal, it is then fairly straightforward to prove that the stable matching so produced is "woman-pessimal"; that is, in any stable matching that can be constructed from the given instance, each woman gets a partner that is equal or superior to the one assigned by the algorithm. The proof is by contradiction of the man-optimal property: Knuth (1976, p 14).

Theorem 3.1.2

The Man-Optimal Solution is the Woman-Pessimal Solution

Proof

We consider men *A* and *B* and women *a* and *b*. Suppose the man-optimal stable matching M_O contains the pair *Aa*, and another stable matching M_1 contains the pairs *Ba* and *Ab*, where *a* prefers *A* to *B*. (That is, we suppose that M_O is not woman-pessimal.) This means that *A* prefers *b* to *a*, otherwise *Aa* would form a blocking pair for M_1. This contradicts the premise that M_O is man-optimal. Therefore, the man-optimal stable matching M_O must also be woman-pessimal.

Corollary

If the algorithm were reversed, processing the women as proposers, then the stable matching produced would be woman-optimal and man-pessimal.

3.1.3 Properties of GS-lists

Now that we have established the man-optimality and woman-pessimality properties of the algorithm, we return to the topic of GS-lists to prove the properties stated in Theorem 2.3.6. The theorem is restated here.

Theorem 2.3.6 - Properties of GS-lists

For a given instance of a stable matching problem:

1. all stable matchings are contained in the GS-lists
2. no matching contained in the GS-lists can be blocked by a pair that is not in the GS-lists
3. in the man-proposing (woman-proposing) stable matching, each man is partnered by the first (last) woman on his GS-list, and each woman by the last (first) man on hers.

Proof of 1

In deriving the MGS-lists, the man-oriented algorithm produces the woman-pessimal stable matching. So if, in the algorithm, man *m* proposes to woman *w* then *w* can do no worse than *m*. So any man coming after *m* in *w*'s list cannot be a partner for *w* in any stable matching. Similarly for the WGS-lists. Hence any pair that is deleted in either the man-oriented or woman-oriented extended algorithm cannot be partners in any stable matching. Therefore all stable partners that could occur in any stable matching are contained in the GS-lists.

Proof of 2

It follows that all pairs that are left in the final GS-lists are preferred, by either the man or the woman, to any pair that has been deleted. So any matching compiled from the GS-list cannot be blocked by a deleted pair.

Proof of 3

We have shown that the man-oriented algorithm produces the man-optimal / woman-pessimal solution. See theorems 3.1.1 and 3.1.2 above. When a man *m* proposes to woman *w* it is his best choice at that time, and in the extended algorithm previous women on his list are deleted. Thus, in the extended algorithm man *m* is always proposing to the first woman on his list, and his final proposal is his partner in the man-optimal stable matching. Similarly, after a woman *w* accepts a proposal in the extended algorithm the less preferred men are deleted from her list, so her final proposal is from the last man on her reduced list, which is her partner in the woman-pessimal stable matching.

3.2 Sets of Unequal Size

3.2.1 Definition

When the two sets being matched have different numbers of elements it is still possible to obtain a stable matching, even though it is a partial matching. The definition of stability is extended to meet this case, as follows.

A matching M is unstable if there is a man *A* and a woman *a* such that the following three properties hold:

i. *A* and *a* are not partners in *M*;
ii. *A* is either unmatched in *M*, or prefers *a* to his partner in *M*;
iii. *a* is either unmatched in *M*, or prefers *A* to her partner in *M*.

(In the context of stable marriage this notion of stability rests on the assumption that people prefer to be married rather than remain single, an assumption that may place mathematics at odds with sociology!)

3.2.2 The "Wallflower" (Author's own example and proof)

In more traditional times of ballroom dancing, the woman who never got asked to dance was left to 'decorate the wall', sitting on the side while the other women were partnered.
We use this term to illustrate the following result for the case where there is one extra woman.

The Wallflower
If *n* men are to be paired in stable matching with some *n* of a set of *n* + 1 women, then the woman that is unmatched in the man-optimal solution is also unmatched in all other stable matchings.
Proof
In the man-optimal stable matching M_O the wallflower w_f is unmatched. Suppose that there is another stable matching M_x that contains the pair (m, w_f); that is, w_f is matched. Then, in M_x some woman w_x is unmatched. But w_x was matched in M_O , which contradicts the fact that M_O is woman-pessimal.

This example is of particular interest because it is counter-intuitive, unless you are a die-hard pessimist. Even if the women are proposing, and the "wallflower" makes the first proposal, the rules of stability are such that, by the end of all the proposals, the "wallflower" will be unmatched.

In fact this result extends to greater inequalities in the set sizes, not just the case where there is one extra woman (or man).

Theorem 3.2.3 In all stable matchings that can be derived from a problem instance of unequal sized sets the smaller set is always matched to the same subset of the larger set. (This theorem was stated by McVitie and Wilson, 1970, and also by Roth, 1982.)

Proof A convenient proof is by referral to the properties of GS-lists, see 3.1.3 above. If we apply the extended algorithm to derive the optimal matching for the larger set, say women, then the WGS-lists for the unmatched women will be empty, and this is the best they can do. It follows from the properties of the GS-lists that any woman with an empty WGS-list cannot be matched in any stable matching.

3.2.3 Algorithm for Unequal Sized Sets

We can still use Knuth's algorithm to derive a stable matching for unequal sized sets. One way of doing this is to create an imaginary person to make up numbers on the short set, thus reducing the problem to one of equal sized sets. This imaginary person, **Ω**, has the quality that he/she is most disliked by all the members of the other sex, and appears at the bottom of their preference lists, repeated if necessary.

At the start of the algorithm, every member of the larger set is assumed to be partnered with **Ω.** At the end of the algorithm, any person that is partnered with **Ω** is considered to be 'unmatched'.

With this construction we can use the standard algorithm, with either the smaller or larger set as proposers.

Alternatively, the algorithm can work with different size sets but it needs some modification to the conditions for halting, thus:

Let there be y men and z women, $y < z$.

Case 1. In the man-oriented process the smaller set are proposing, and engagements are made according to the preferment of the men. The algorithm must stop as soon as all y men are engaged, and we can simply substitute y for n in the standard algorithm, line 2.

Case 2. If the larger set (women) are proposing, we can only halt when $z - y$ of the women have each proposed to all y men, and been rejected by all of them. The author gives her changes to the standard algorithm, with the larger set proposing (woman-oriented). Variables are as before, with the addition of:

y = number of men;

z = number of women; $y < z$;

u = number of exhausted women's lists, = number of unmatched women;

extra code has been inserted, and highlighted, thus:

Algorithm to derive a stable marriage from unequal sized sets, larger set proposing

```
1    k ← 0 ; u ← 0;
2    while k < z and u < (z – y) do
3        begin x ← ( k + 1 )-st woman ;
4        while x ≠ Ω and x's list is not empty do
5            begin X ← best choice remaining on x's list ;
6            if X prefers x to his fiancée then
7                begin engage xand X ;
8                x ← preceding fiancée of X
9                end ;
10           if x ≠ Ω then withdraw X from x's list
11           end ;
11a      if x's list is empty then u = u + 1
12       k ← k + 1
13       end ;
14   celebrate y weddings
```

In fact, the extra code on lines **2** and **11a** is redundant, because the control on the number of unmatched women is not required. When $k = y$, we are in the position that y of the women are engaged to y of the men, and u = 0. After that, for each iteration of the main loop (lines **2** through **13**) one woman's list becomes exhausted without engagement. So by the time $k = z$ it will also be true that $u = (z - y)$, and u can never be greater than $(z - y)$. The only modification required is the extra condition for a non-empty list on line **4**.

4 Chapter 4 - Solution Sets

4.1 Median Solutions

In earlier sections we have shown that every problem instance for a stable matching has at least one solution, namely the man-optimal (or woman-optimal) solution. Further, we know how to derive the proposer-optimal solution by applying the algorithm to the preference lists. If the man-optimal solution is different from the woman-optimal solution, then a second application of the algorithm, with the women proposing, will produce the other extremal solution.

We have also shown that, depending on the instance, there may be many other stable matchings. However, there appears to be no easy method for deriving the solutions that may exist between the two optima.

The 'brute force' approach would be to consider all possible matchings between the two sets and to test each matching in turn for stability. To consider the work involved in such an approach we first note that the number of matchings between two sets of size n is $n!$. [The first man can be paired with any of n women, the second man with any of $n-1$ women, and so on, until the n-th man has no choice.]

We restate the stability checking algorithm from section 2.2.3 , thus:

```
for m = 1 to n do
        for each w that m prefers to p_M(m) do
                if w prefers m to p_M(w) then
                begin
                        report matching unstable
                        halt
                end
report matching stable
```

The workload of this algorithm, including the preparation of the ranking arrays, is $\boldsymbol{O}(n^2)$ but when applied to all possible matchings the total workload for finding all solutions is $\boldsymbol{O}(n!n^2)$. Even for small n this is prohibitive.

A desirable approach therefore is to first eliminate all matchings that cannot possibly be stable, and apply the checking algorithm to only the remaining potential. To this end we return to the Gale-Shapley lists from section 2.3.6.

4.2 The Gale-Shapley Lists

4.2.1 Deriving the GS-Lists

In section 2.3.6 we showed how two applications of the extended algorithm (man proposing and woman proposing) produced the reduced preference lists, known as the GS-lists.

For ease of reference we show again the GS-lists produced for that example.

Men	GS-List
A	*e h c*
B	*c h f*
C	*h e a f b*
D	*f h*
E	*g b a*
F	*a e*
G	*b e g*
H	*d*

Women	GS-List
a	*E C F*
b	*C E G*
c	*A B*
d	*H*
e	*F G C A*
f	*B C D*
g	*G E*
h	*D A B C*

4.2.2 Exploiting the GS-Lists

The author provides her own detailed evaluation of this example.

We first define a "stable pair" as a pair *mw* that occurs in some stable matching, and we say that *w* is a "stable partner" of *m*, and vice versa. If *m* and *w* are partners in *all* stable matchings then we call *mw* a "fixed pair".

For stability checking we would want to construct only those matchings that exist in the GS-lists. By looking at the above GS-lists, we see that *Hd* is a fixed pair. So the total number of matchings from complete preference lists to be considered for stability checking is straightaway divided by *n*. In our example, the number of permuted matchings, based on the length of the (men's) GS-lists, is no more than 3x3x5x2x3x2x3x1 = 1620, whereas the total matchings from the original complete lists is 8! = 40320.

The upper bound of 1620 includes woman *w* being paired with more than one man. So we can reduce this further because there is an entailment of partnering. Man *F* can only partner *a* or *e*. Therefore if *A* or *C* or *G* are paired with *e* then *F* must be paired with *a*. We can develop strings of entailment, thus:

***Fe*...... (start by choosing one of *F*'s two possible partners)

$\Downarrow$

Ea - - - - - - - - or - - - - - - - - - ***Ca*...** (*F* is gone, so *a* can only partner *E* or *C*)

$\Downarrow$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\Downarrow$

Gg (*g* can only partner *G)* $\qquad$ ***Eb*** - - - - - - - or - - - - - - - - ***Eg***

$\Downarrow$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\Downarrow$ $\qquad\qquad\qquad\qquad\qquad$ $\Downarrow$

Cb (*b* can only partner *C)* $\qquad$ ***Gg*** $\qquad\qquad\qquad\qquad\qquad$ ***Gb***

In this way the possible matchings that need to be checked for stability can be considerably reduced, depending on the amount of reduction in the GS-lists.

It is important to remember that the GS-lists contain all the stable pairs, but they may also contain some pairs that do not occur in any stable matching. In our example *Bh* is not a stable pair. We can see this directly from the lists. *Bh* entails *Ac* (*c* only has a choice of *A* or *B*), and the pairs *Bh* and *Ac* cannot be in the same stable matching because they are blocked by *Ah* (*A* prefers *h* to *c*, and *h* prefers *A* to *B*).

4.3 The Lattice Structure

4.3.1 Comparing Stable Matchings

We have seen that for an instance of preference lists there may be many possible stable matchings. We have also seen that there is always a man-optimal solution and a (not necessarily distinct) woman-optimal solution.

By extending our notion of preference to an entire matching, rather than just to the partner assigned in the matching, we are able to compare the different stable matchings that exist for a given instance. The comparison requires a natural relation, which is expressed as a *dominance* relation.

We define the man-oriented dominance relation as follows:

Let M þ$_m$ M' (also written M' æ$_m$ M) if every man has at least as good a partner in M' as he has in M. If M þ$_m$ M' and $M \neq M'$, then we write Mp$_m$ M'.

We denote the analogous woman-oriented dominance relation by M þ$_w$ M', and it means that every woman has at least as good a partner in M' as she has in M. Again, M p$_w$ M' means that M þ$_w$ M' and $M \neq M'$.

We will show in section 4.3.4 that the relation þ$_w$ coincides with æ$_m$, that is, M þ$_w$ M' if and only if M' þ$_m$ M.

We first show that the solution set of stable matchings forms a distributive lattice under the relation þ$_m$, with the man-optimal and woman-optimal solutions being the maximum and minimum elements.

4.3.2 Lattices – terms and definitions

The lattice structure provides a framework for further insights into the stable matching problem, such as determining the number of solutions, and determining the intermediate solutions between the two extremes. It is therefore appropriate at this point to state some of the defining characteristics of lattices. We start with a "partially ordered set" or *poset*.

A *poset* is a set P together with a binary relation $\leq$ such that the following conditions are satisfied for all $x, y, z \in P$.

1. $x \leq x$ (Reflexivity)

2. If $x \le y$ and $y \le x$, then $x = y$ (Antisymmetry)

3. If $x \le y$ and $y \le z$, then $x \le z$ (Transitivity)

The *greatest lower bound* (also called *GLB* or *meet*) of elements *x* and *y* in a poset is a common lower bound z such that $z \le x$ and $z \le y$, and every other common lower bound z' satisfies the inequality $z' \le z$. Such an element, if it exists, is denoted by $x \wedge y$.

A *minimal* element of a poset is an element z such that there is no element z' where $z' < z$. A poset may have several, incomparable, minimal elements, but if there is only one minimal element then that is the *minimum* element.

The *least upper bound* (also called *LUB* or *join*) of elements *x* and *y* in a poset is a common upper bound z such that $x \le z$ and $y \le z$, and every other common upper bound z' satisfies the inequality $z \le z'$. Such an element, if it exists, is denoted by $x \vee y$.

A *maximal* element of a poset is an element z such that there is no element z' where $z' > z$. A poset may have several, incomparable, maximal elements, but if there is only one maximal element then that is the *maximum* element.

We can now state that a *lattice* is defined as a *poset* in which every pair of elements has both a *greatest lower bound* and a *least upper bound*. The *GLB* or *meet* of *x* and *y* is denoted by $x \wedge y$, and the *LUB* or *join* of *x* and *y* is denoted by $x \vee y$.

Furthermore, a *distributive lattice* is a lattice in which the laws of distributivity hold; thus, for any elements x, y, and z

(1) $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

and (2) $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.

4.3.2.1 Laws of Distributivity

It is of interest that in a lattice as defined above only one of these identities is necessary to prove distributivity since either identity implies the other. The proof is adapted from Grätzer, 1971.

Theorem 4.3.2.1 In any lattice L, (1) holds for all $x, y, z \in L$ if and only if (2) holds for all $x, y, z \in L$.

Proof

Let L be a lattice and let a,b,c ˜ L.
Let identity (1) hold, and substitute $x = a \vee b, y = a, z = c$, which gives

$(a \vee b) \wedge (a \vee c) = ((a \vee b) \wedge a) \vee ((a \vee b) \wedge c)$

$= a \vee ((a \vee b) \wedge c)$ {because $a = (a \vee b) \wedge a$)}

$= a \vee (a \wedge c) \vee (b \wedge c)$ {using (1) with $x = c, y = a, z = b$}

$= a \vee (b \wedge c)$, {because $a = a \vee (a \wedge c)$}

which verifies identity (2).

Similarly, if we take identity (2) with $x = a \wedge b, y = a, z = c$ we get

$(a \wedge b) \vee (a \wedge c) = ((a \wedge b) \vee a) \wedge ((a \wedge b) \vee c)$

$= a \wedge ((a \wedge b) \vee c)$

$= a \wedge (a \vee c) \wedge (b \vee c)$

$= a \wedge (b \vee c)$, which verifies identity (1).

4.3.3 Lattices for a Solution Set of Stable Matchings

Theorem 4.3.3

For a given instance of preference lists the solution set of stable matchings $\{M, M', M'', ...\}$, with the dominance relation þ$_m$, is a lattice.

Proof

Firstly, it is a poset under the man-oriented dominance relation þ$_m$ because of:
1) Reflexivity - M þ$_m$ M
2) Antisymmetry - if M þ$_m$ M' and M' þ$_m$ M then $M = M'$
3) Transitivity - if M þ$_m$ M' and M' þ$_m$ M'' then M þ$_m$ M''

Secondly, it is a lattice because, for any two matchings M and M' the meet (greatest lower bound) exists in the poset (proof follows, 4.3.3.2), and the join (least upper bound) exists in the poset (proof follows, 4.3.3.3). The proofs are adapted from Gusfield and Irving, 1989, and theorem 4.3.3.1 is a preparation for the other two.

4.3.3.1 If One goes up the Other goes down

Theorem: Let M and M' be different stable matchings from a given instance of preference lists, where *m* and *w* are partners in M but not partners in M'. Then either *m* or *w* prefers M and the other prefers M'.

Proof

We denote by Ξ and Ψ (respectively Ξ' and Ψ') the sets of men and women who prefer M to M' (respectively M' to M). For any couple (m,w) in M, if $m \in \Xi$ then $w \in \Psi'$, otherwise (m,w) would block M'. Similarly, for any couple (m,w) in M', if $m \in \Xi'$ then $w \in \Psi$, otherwise (m,w) would block M. Thus, $|\Xi| \leq |\Psi'|$ and also $|\Xi'| \leq |\Psi|$.

But the number of men who have different partners in M and M' equals the number of women who have different partners, so $|\Xi| + |\Xi'| = |\Psi| + |\Psi'|$. It then follows that $|\Xi| = |\Psi'|$ and $|\Xi'| = |\Psi|$.

Thus every man who prefers M has a partner in M who prefers M', and every man who prefers M' has a partner in M' who prefers M.

4.3.3.2 Proof that the join of M and M' exists in the solution set

Let M and M' be two distinct stable matchings from a given instance of preference lists, and let M'' be the set of man-woman pairs that is formed by each man being paired with the woman he prefers from M and M'.

In order to prove that M'' is a stable matching we must first show that M'' is indeed a matching; ie. we must be sure that no woman appears more than once in M''. Suppose not; then M'' has two couples, (m,w) from M and (m',w) from M', where *m* prefers *w* to his partner in M' and m' prefers *w* to his partner in M.

The above theorem (4.3.3.1) gives us that *m* prefers M so then *w* prefers M', and m' prefers M' so then *w* must prefer M, which is a contradiction, so M'' is indeed a matching.

To show that the M'' matching is stable we suppose to the contrary that it contains the pairs (m,w') and (m',w) but is blocked by the non-matched pair (m,w). Then m prefers w to w', which was his preferred partner from either M or M', and m therefore prefers w to both his M partner and his M' partner. Also w must prefer m to m', but (m',w) is a pair in either M or M', so (m,w) would then have blocked either M or M', which we know are stable, so there is a contradiction.

Therefore $M \vee M'$ exists in the solution set.

4.3.3.3 Proof that the meet of M and M' exists in the solution set

This is the dual of proof 4.3.3.2.

Let M and M' be two distinct stable matchings from a given instance of preference lists, and let M'' be the set of man-woman pairs that is formed by each man being paired with the woman he *least* prefers from M and M'.

In order to prove that M'' is a stable matching we must first show that M'' is indeed a matching; ie. we must be sure that no woman appears more than once in M''. Suppose not; then M'' has two couples, (m,w) from M and ($m¢$,w) from M', where m prefers his partner in M' to w and m' prefers his partner in M to w.

The above theorem (4.3.3.1) gives us that m prefers M' so then w prefers M, but m' prefers M so then w must prefers M', which is a contradiction, so M'' is indeed a matching.

To show that the M'' matching is stable we suppose to the contrary that it contains the pairs (m,w') and (m',w) but is blocked by the non-matched pair (m,w). Since w is the *less* preferred partner of m' in either M or M', theorem 4.2.3.1 gives us that m' is the *more* preferred partner of w in either M or M', and the blocking pair (m,w) says that w would prefer m to *both* $p_M(w)$ or $p_{M¢}(w)$. Also, the blocking pair says that m prefers w to w', which was his partner in either M or M', so (m,w) would then have blocked either M or M', which we know are stable, so there is a contradiction.

Therefore $M \wedge M'$ exists in the solution set.

4.3.4 Woman dominance is the Inverse of Man dominance

We noted in section 4.3.1 that the relation $þ_w$ coincides with $æ_m$. We can now give the theorem and proof.

Theorem 4.3.4

For any two stable matchings, $M¢ þ_m M$ if and only if $M þ_w M¢$.

Author's Own Proof

Suppose $M¢ þ_m M$. Let w be an arbitrary woman, and let (m,w) be a pair in M, and $(m¢, w), (m, w¢)$ are pairs in $M¢$. The fact that $M¢ þ_m M$ gives us that m prefers w to $w¢$. Suppose that w prefers m to $m¢$. Then (m,w) would block $M¢$. So it must be that w prefers $m¢$ to m, and $M þ_w M¢$.

Conversely, suppose $M þ_w M¢$. Let $m¢$ be an abitrary man, and let $(m¢, w¢)$ be a pair in $M¢$, and $(m¢, w), (m, w¢)$ are pairs in M. The fact that $M þ_w M¢$ gives us that $w¢$ prefers $m¢$ to m. Suppose that $m¢$ prefers $w¢$ to w. Then $(m¢, w¢)$ would block M. So it must be that $m¢$ prefers w to $w¢$, and $M¢ þ_m M$.

Summary of lattice properties for a solution set of stable matchings

There are *two posets* for the solution set of stable matchings from any problem instance, namely (M, $þ_m$) and (M, $þ_w$), and the elements in both posets are the stable matchings $M, M', M'', etc.$

Each poset defines a lattice since every pair of stable matchings has a least upper bound and a greatest lower bound.

Under the man-oriented dominance relation $þ_m$, there is always a maximum element which is the common *LUB* of *all* elements. This is the man-optimal solution, M_O , which is also the woman-pessimal solution by the inverse relation $þ_w$.

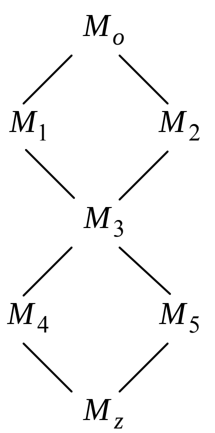
Also for (M, þ$_m$) there is always a minimum element which is the common *GLB* of *all* elements, and it is the woman-optimal solution, M_Z, which is also the man-pessimal solution by the inverse relation þ$_w$.

We show below (ref. 4.3.7) that the solution poset (M, þ$_m$) , or (M, þ$_w$), for a stable matching instance with strict preference forms a *distributive* lattice. Where the men's/women's preferences are weakened by indifference the lattice may be nondistributive.

4.3.5 The Hasse Diagram of the Solution Set

A Hasse diagram gives a useful representation of a lattice structure for stable matchings, showing clearly the comparability of elements/solutions and depicting the dominance relation between different solutions. Each node in the diagram represents a stable matching and the lines show a relation between two comparable nodes, directed downward.

Example: where M_0 is the man-optimal solution, and M_z is the woman-optimal solution.



M_1 and M_2 are not comparable
(at least one man has a better partner in M_1 and at least one man has a better partner in M_2)

M_4 and M_5 are not comparable

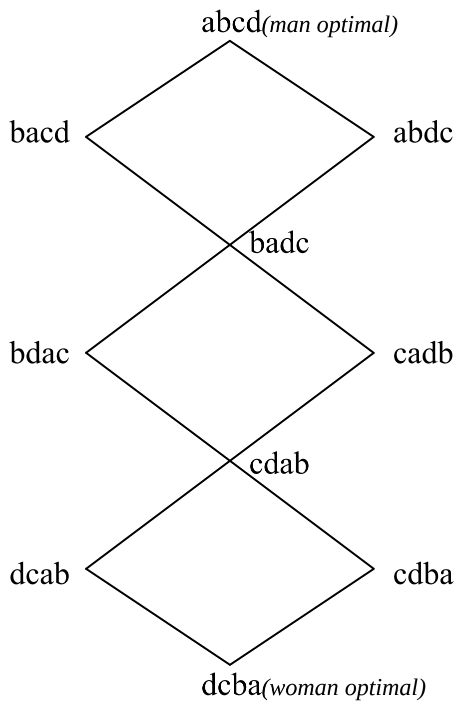
All other element pairs are comparable.
eg:Every man in M_3 has at least as good a partner as he has in M_5

We give an expanded example of a Hasse diagram representing the lattice structure for the solution set of stable matchings for the following preference lists.

Men	**Order of Preference**			
A	*a*	*b*	*c*	*d*
B	*b*	*a*	*d*	*c*
C	*c*	*d*	*a*	*b*
D	*d*	*c*	*b*	*a*

Women	**Order of Preference**			
a	*D*	*C*	*B*	*A*
b	*C*	*D*	*A*	*B*
c	*B*	*A*	*D*	*C*
d	*A*	*B*	*C*	*D*

The label $w_A w_B w_C w_D$ on a node indicates the stable matching in which men A,B,C and D are matched with women w_A, w_B, w_C or w_D respectively.



Notes:

1. For example 'cadb' æ$_m$ 'dcab' by transitivity.

2. Also by example, 'bdac' and 'cadb' are incomparable; men A and D prefer the matching 'bdac' to 'cadb', whereas men B and C prefer the matching 'cadb' to 'bdac'.

3. all possible stable matchings are represented in this Hasse diagram

4.3.6 The Lattice of Stable Matchings is Distributive

We have shown that the solution set of stable matchings always forms a lattice, with a maximum element (man-optimal) and a minimum element (woman-optimal). We now show that the lattice is always *distributive* (providing the solution set derives from complete preference lists, with no indifference).

We need only prove one of the required identities stated in 4.3.2 since we proved in 4.3.2.1 that one identity implies the other.

4.3.7 Proof of Distributivity

Given a lattice ***L*** , we define a **sub-lattice** of ***L*** as a nonempty subset ***S*** such that $a \vee b \in S$ and $a \wedge b \in S$ whenever $a \in S$ and $b \in S$.

From general lattice theory we are given that a lattice *L* is nondistributive if and only if it contains a sublattice that is isomorphic to one of the two lattices shown below. (Grätzer, 1971, Chapter 2)

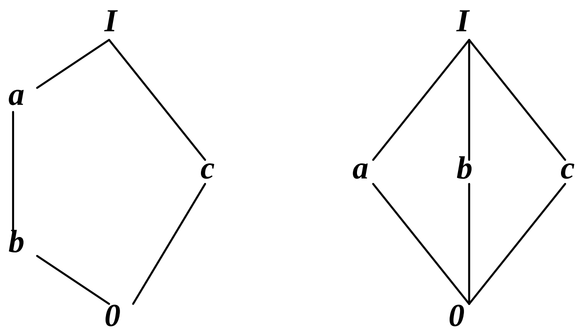


Fig. 1 Fig. 2

For both of these lattices, $a \wedge (b \vee c) = a \wedge I = a$

but, in Fig. 1, $(a \wedge b) \vee (a \wedge c) = b \vee 0 = b$

and, in Fig. 2, $(a \wedge b) \vee (a \wedge c) = 0 \vee 0 = 0$

So, in both lattices, the first identity of the law of distribution does not hold.

However, to prove that a solution set of stable matchings *cannot* contain such a sublattice is probably not a useful proof technique. Gusfield & Irving (1989, page 21) give a generalised proof that the stable matching lattice is distributive but the author gives a more transparent proof by taking the direct approach of considering all possible cases for any three solutions X, Y and Z of the complete solution lattice that results from a given stable matching instance.

Theorem 4.3.7 – The Lattice of Stable Matchings is Distributive

Proof

Let X, Y and Z be stable matchings of a fixed problem instance, and let $U = X \wedge (Y \vee Z)$ and $V = (X \wedge Y) \vee (X \wedge Z)$. We need to prove that $U = V$. It is sufficient to consider any man m and prove that $p_U(m) = p_V(m)$, for if every man has the same partner in U as he has in V then $U = V$.

Let m's partner in X, Y or Z be x, y, z respectively.
There are 3! = 6 possibilities for the order of x, y, z in m's preference lists:

1. $m: \; x \; y \; z$
2. $m: \; x \; z \; y$
3. $m: \; y \; x \; z$
4. $m: \; y \; z \; x$
5. $m: \; z \; x \; y$
6. $m: \; z \; y \; x$

and we consider each case in turn, noting that other women may be in the lists and the juxtaposition of x, y, z is not important, only the sequence in which they appear.

By using 4.3.3.2 and 4.3.3.3 above we are able to deduce m's partner in the joins (preferred partner) and meets (less preferred partner) thus:

Case 1: $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = y$

$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = y$

Case 2: $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = z$

$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = z$

Case 3: $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = x$

$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = x$

Case 4: $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = x$

$$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = x$$

Case 5: $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = x$

$$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = x$$

Case 6: $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = x$

$$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = x$$

So, in all cases, $p_U(m) = p_V(m)$, so $U = V$, and the theorem is proved.

We note that if indifference is introduced into the preference lists, or they are incomplete, then we cannot prove that the solution lattice is distributive. By counterexample, in Case 1, if m is indifferent between y and z

then $p_U(m) = p_X(m) \wedge (p_Y(m) \vee p_Z(m)) = y$ **or** z and

$$p_V(m) = (p_X(m) \wedge p_Y(m)) \vee (p_X(m) \wedge p_Z(m)) = y \textbf{ or } z,$$

so the identity does not hold.

5 Chapter 5 - Variations of Stable Marriage

5.1 Many-To-One Matching

We return to the college admissions (hospital-residents) problem that was stated in section 2.3.1 and explore the implications of deriving a stable matching in the one (college) to many (students) situation .

The college admissions problem, whereby each college seeks to gain acceptance from *multiple* preferred students to fill a pre-set quota of places available, can be seen as analogous to a stable marriage problem wherein one of the sets is polygamous. Indeed, Gale and Shapley (1962) present the stable marriage theorem as an instance of the college admissions problem wherein each college has a quota of unity.

It is interesting that the hospital-residents algorithm used by the US hospitals in the National Residents Matching Program (NRMP) favours the hospitals (hospitals proposing), whereas the college admissions algorithm put forward by Gale and Shapley (1962) favours the students (students proposing).

We could justify the hospital/college optimality by saying that hospitals/colleges are more likely to rank students on strict academic criteria, whereas students may rank colleges on criteria such as location, prestige, ancestry, etc., which are of less importance surely? Gale and Shapley justify the student optimality approach by saying that "colleges exist for the students"; but there is another aspect to this: students who are being proposed to are intelligent enough to understand the matching algorithm and are capable of falsifying their preferences in order to achieve a better result for themselves. Once the preferences are misrepresented the real stability of the solution must be compromised. If the roles are reversed, and the students do the proposing, it is highly unlikely that the colleges would falsify their preferences, so a more honest result ensues.

5.1.1 Hospital-Residents – hospital proposing

The major difference between the iterative procedure for stable marriage and the one for (polygamous) hospital-residents is that the hospital-residents procedure does not terminate when the last resident (student/woman) is asked for the first time. To see this, consider 2 hospitals, each with a quota of 6 places, and a total of 6 residents. Clearly, hospital #1 will ask all 6 residents, and hospital #2 must also ask all 6 residents.

In real life it is almost certain that the total number of places available is either greater or less than the total number of applicants; also, a member of either set may declare one or more members of the other set to be unacceptable. In practice this means that the matching outcome will only be a partial mapping, and quotas may be unfilled, even if residents are unassigned.

We can use similar terminology and notation as for the stable marriage problem, so that if resident *r* is assigned to hospital *H* in some stable matching we can say that (*H*, *r*) is a stable pair, *M* is a stable matching, $p_M(r)$ is the hospital assigned to *r* in *M*, and $p_M(H)$ is *one of the set* of residents assigned to *H* in *M*.

In the hospital-residents context, a matching is *unstable* if there is a resident *r* and hospital *H* such that all of the following hold:

i. *H* is acceptable to *r* and *r* to *H*;
ii. either *r* is unmatched, or *r* prefers *H* to his assigned hospital;
iii. either *H* does not have its quota filled, or *H* prefers *r* to at least one of its assigned residents.

A hospital-oriented algorithm that reduces preference lists can then be shown:

Hospital-Oriented Algorithm (One-To-Many)

```
initialise every resident to be unassigned
initialise every hospital's assignments to be 0
while (some hospital H has assignments < quota) and
        (H's list is not empty) do
         begin
          r ← first resident remaining on H's list
          if r is already assigned to H′ do
          begin remove r from H′'s set of provisional assignments
                    decrement H′'s assignment count
          end
          add r to H's set of provisional assignments
          increment H's assignment count
          for each successor H′ of H on r's list do
                    remove H′ and r from each other's lists
          end
```

Alternatively, a One-To-Many hospital-resident problem can be adapted to the One-To-One stable marriage problem so that the standard algorithm can be used, thus:

1. each hospital H_A with q places is replaced by q separate but identical hospitals, $H_{A_1},...,H_{A_q}$, each with just one place and identical preferences
2. each occurrence of H_A in a resident's preference list is replaced by the sequence $H_{A_1},...,H_{A_q}$

To see how these two approaches work in practice the author provides her own example of 3 hospitals and 5 residents, each hospital having a quota for 3 residents.

Example 5.1.1(a) The original preference lists are:

Hospital	Order of Preference					Resident	Order of Preference		
H_A	r_2	r_3	r_4	r_5	r_1	r_1	H_A	H_B	H_C
H_B	r_4	r_3	r_1	r_5	r_2	r_2	H_B	H_C	H_A
H_C	r_3	r_2	r_1	r_4	r_5	r_3	H_A	H_C	H_B
						r_4	H_B	H_A	H_C
						r_5	H_A	H_B	H_C

The use of these lists in a modified procedure, that reduces preference lists as it proceeds, would give rise to the following proposals, the ones highlighted thus being rejected.

(H_A,r_2), (H_A,r_3), (H_A,r_4), (H_B,r_4), (H_A,r_5), (H_B,r_1), (H_B,r_2), (H_A,r_1)

That is, there are 8 proposals, H_A gets its quota, H_B is underfilled, and H_C gets no residents at all.

Now, if we expand the preference lists as stated above in order to use the standard, non-extended, unequal-sized-sets, stable marriage algorithm we get these preference lists:

Hospital	Order of Preference				
H_{A1}	r_2	r_3	r_4	r_5	r_1
H_{A2}	r_2	r_3	r_4	r_5	r_1
H_{A3}	r_2	r_3	r_4	r_5	r_1
H_{B1}	r_4	r_3	r_1	r_5	r_2
H_{B2}	r_4	r_3	r_1	r_5	r_2
H_{B3}	r_4	r_3	r_1	r_5	r_2
H_{C1}	r_3	r_2	r_1	r_4	r_5
H_{C2}	r_3	r_2	r_1	r_4	r_5
H_{C3}	r_3	r_2	r_1	r_4	r_5

Resident	Order of Preference								
r_1	H_{A1}	H_{A2}	H_{A3}	H_{B1}	H_{B2}	H_{B3}	H_{C1}	H_{C2}	H_{C3}
r_2	H_{B1}	H_{B2}	H_{B3}	H_{C1}	H_{C2}	H_{C3}	H_{A1}	H_{A2}	H_{A3}
r_3	H_{A1}	H_{A2}	H_{A3}	H_{C1}	H_{C2}	H_{C3}	H_{B1}	H_{B2}	H_{B3}
r_4	H_{B1}	H_{B2}	H_{B3}	H_{A1}	H_{A2}	H_{A3}	H_{C1}	H_{C2}	H_{C3}
r_5	H_{A1}	H_{A2}	H_{A3}	H_{B1}	H_{B2}	H_{B3}	H_{C1}	H_{C2}	H_{C3}

And then the proposals would be: (rejections highlighted thus):

(H_{A1},r_2), (H_{A2},r_2), (H_{A2},r_3), (H_{A3},r_2), (H_{A3},r_3), (H_{A3},r_4), (H_{B1},r_4), (H_{A3},r_5), (H_{B2},r_4), (H_{B2},r_3), (H_{B2},r_1), (H_{B3},r_4), (H_{B3},r_3), (H_{B3},r_1), (H_{B3},r_5), (H_{B3},r_2), (H_{A1},r_3), (H_{A2},r_4), (H_{A2},r_5), (H_{A3},r_1), (H_{B2},r_5), (H_{B2},r_2), (H_{B3},r_4), at this point H_{B3} has an empty list, (H_{C1},r_3), (H_{C1},r_2), (H_{C1},r_1), (H_{C1},r_4), (H_{C1},r_5), (H_{C1},r_3), at this point H_{C1} has an empty list, (H_{C2},r_3), (H_{C2},r_2), (H_{C2},r_1), (H_{C2},r_4), (H_{C2},r_5), (H_{C2},r_3), at this point H_{C2} has an empty list, (H_{C3},r_3), (H_{C3},r_2), (H_{C3},r_1), (H_{C3},r_4), (H_{C3},r_5), (H_{C3},r_3), and finally H_{C3} also has an empty list.

That is, there are 41 proposals with 36 rejections.

The resultant matching is exactly the same as that of the modified procedure above. The purpose of showing the One-To-One proposals on the expanded lists is to illustrate clearly the number of redundant proposals that are introduced, because each resident gets multiple proposals from the same hospital. If the extended version of the One-To-One algorithm were applied, that reduces the preference lists as it proceeds, then the number of proposals would be significantly less, and H_C for instance would not make any proposals at all because the list would be empty beforehand. In fact, 28 of the rejections have resulted from repeat proposals by the same (cloned) hospital to the same resident, and these would have been avoided by using reduced preference lists.

5.1.2 College Admissions – student proposing

Turning the algorithm around to allow the resident/student to do the proposing requires a different modification. The author makes the observation that we cannot reduce preference lists, because any hospital/college with an unfilled quota may well want an application from another student that is lower in the lists than the 'best so far'.

Also, because there may be more students than places, the algorithm must not halt until every unassigned student has asked every college on his/her list.

With this in mind, the author presents her modified algorithm, thus:

Student-Oriented Algorithm (Many-To-One)

```
initialise every student to be unassigned
initialise every college's assignments to be 0
while some student s is unassigned and s's list is not exhausted do
        begin
         C ← next college on s's list
         if C already has full quota of assignments do
                  begin
                  if C prefers s to a provisionally assigned student s'
                  make s' unassigned
                  decrement C's assignment count
                  end
         add s to C's set of provisional assignments
         increment C's assignment count
         end
```

Using the non-expanded example for hospitals and residents from 5.1.1(a) above the proposals would be:

(r_1, H_A), (r_2, H_B), (r_3, H_A), (r_4, H_B), (r_5, H_A), which in this example results in the same matching as the hospital oriented algorithm.

We note that only five proposals are necessary, and there are no rejections.

5.1.3 Many-to-Many Matching

The stable matching problem can extend to Many-To-Many matchings, for which there are various examples in society. We briefly mention two of them:

5.1.3.1 Course-Based Students

It is becoming more usual for students to pursue different courses at different colleges concurrently, and accumulate the different course credits towards a final degree. The administration of this requires a central applications body with appropriate algorithms.

5.1.3.2 Contract Workers

The tendency towards contract workers has increased in the last few years, particularly in industries where highly specialised skills may be required for only a short space of time on development projects. Example: the use of underwater welders at crucial stages in a large project. The contractors wish to choose the best projects, and maybe to have several projects running concurrently, that is, project A on Mondays and Tuesdays, project B on Wednesdays, and so on; the clients/project management on the other hand wishes to get the best group of contractors for a particular task within the project.

5.1.3.3 The Many-To-Many Algorithm

Gusfield and Irving (1989) state that "relevant aspects" of the Gale and Shapley algorithm can be combined to produce a version for the many-to-many situation, but no further details are given.

The author claims that the many-to-many problem cannot be reduced to a one-to-one problem in the same way that a one-to-many problem can be reduced. (See section 5.1.1). If we attempt to expand *both* preference lists by cloning the residents as well as the hospitals we would get lists that look as follows (assuming our example 5.1.1(a) but that each resident may register at *two* hospitals):

Hospital	Order of Preference					
H_{A1}	r_{b1}	r_{b2}	r_{c1}	r_{c2}	r_{d1}	. . .
H_{A2}	r_{b1}	r_{b2}	r_{c1}	r_{c2}	r_{d1}	. . .
H_{A3}	r_{b1}	r_{b2}	r_{c1}	r_{c2}	r_{d1}	. . .
H_{B1}	r_{d1}	r_{d2}	r_{c1}	r_{c2}	r_{a1}	. . .
H_{B2}	r_{d1}	r_{d2}	r_{c1}	r_{c2}	r_{a1}	. . .
H_{B3}	r_{d1}	r_{d2}	r_{c1}	r_{c2}	r_{a1}	. . .
H_{C1}	r_{c1}	r_{c2}	r_{b1}	r_{b2}	r_{a1}	. . .
H_{C2}	r_{c1}	r_{c2}	r_{b1}	r_{b2}	r_{a1}	. . .
H_{C3}	r_{c1}	r_{c2}	r_{b1}	r_{b2}	r_{a1}	. . .

Resident	Order of Preference					
r_{a1}	H_{A1}	H_{A2}	H_{A3}	H_{B1}	H_{B2}	. .
r_{a2}	H_{A1}	H_{A2}	H_{A3}	H_{B1}	H_{B2}	. .
r_{b1}	H_{B1}	H_{B2}	H_{B3}	H_{C1}	H_{C2}	. .
r_{b2}	H_{B1}	H_{B2}	H_{B3}	H_{C1}	H_{C2}	. .
r_{c1}	H_{A1}	H_{A2}	H_{A3}	H_{C1}	H_{C2}	. .
r_{c2}	H_{A1}	H_{A2}	H_{A3}	H_{C1}	H_{C2}	. .
r_{d1}	H_{B1}	H_{B2}	H_{B3}	H_{A1}	H_{A2}	. .
r_{d2}	H_{B1}	H_{B2}	H_{B3}	H_{A1}	H_{A2}	. .
r_{e1}	H_{A1}	H_{A2}	H_{A3}	H_{B1}	H_{B2}	. .
r_{e2}	H_{A1}	H_{A2}	H_{A3}	H_{B1}	H_{B2}	. .

Initially all is in order, and H_{A1} proposes to r_{b1} as usual. However, later on, when H_{A3} proposes to r_{b2}, it is not possible for r_{b2} to recognise that he/she is actually the same resident as r_{b1} and that 'they' actually are already assigned to hospital H_A. We have engineered impossible lists whereby hospital H_A has made two proposals to the same student r_b.

The author proposes two methods for deriving a stable matching on a many-to-many instance:

Method 1

Apply the many-to-one (college admissions) algorithm in successive iterations, adjusting the preference lists between iterations to remove stable pairs already achieved. Note that all iterations must be either hospital/college oriented or resident/student oriented; switching from one to the other could yield an unstable matching.

Method 2

Use a single iteration of a modified algorithm, whereby after a pairing is provisionally assigned, the element pair are removed from each other's preference lists *and* also from the lists of hospital/college clones and resident/student clones. Note that we cannot remove *lower* ranking pairs, as is done in the normal extended algorithm, for reasons stated above in section 5.1.2.

5.2 Indifference

5.2.1 Tied Preferences

When participants in a stable matching problem are permitted to express indifference in their preference lists they have lists that are only partially ordered because there are tied entries.

The implications are that we need to introduce three new concepts of stability, thus:

1. A matching M is *super-stable* if there is no blocking pair (m,w) such that m is indifferent between w and $p_M(m)$ and w is indifferent between m and $p_M(w)$. It is entirely possible that no super-stable matchings exist. In the trivial example of complete indifference by all parties there can be no super-stable matching.
2. A matching is *strongly-stable* if there is no blocking pair (m,w) such that either m strictly prefers w to $p_M(m)$ and w is indifferent between m and $p_M(w)$, or m is indifferent between w and $p_M(m)$ and w strictly prefers m to $p_M(w)$. Again, there can be instances for which no strongly-stable matching exists.
3. A matching is *strictly-stable* (the author's term) if there is no blocking pair (m,w) such that m strictly prefers w to $p_M(m)$ and w strictly prefers m to $p_M(w)$. This is the standard notion of stability that we have explored up to now with complete preferences. In an instance with indifference, if the ties in the preference lists are broken arbitrarily, thereby creating complete lists of strict preference, then the standard algorithm can be applied to derive a strictly-stable matching; of course, it may be that a different tie-breaking would lead to a different strictly-stable matching.

5.3 Stable Roommates

The stable roommates problem is a generalization of the stable marriage problem to a matching within a single set of size n , n an even number. Each person p in the set ranks *all* of the others in order of preference. A matching is a partition of the set into disjoint pairs (of partners, or "roommates"), and a matching M is stable if there is no blocking pair (p_1, p_2) such that p_1 and p_2 mutually prefer each other to their partners in M.

5.3.1 The Existence Problem

The striking difference between the stable marriage problem and the stable roommates problem is that there are instances of the stable roommates problem for which there is no stable matching.
For example, the stable roommates instance for $n = 4$, with the following preference lists, does not have a stable matching.

Person	Preference List		
p_1	p_3	p_2	p_4
p_2	p_1	p_3	p_4
p_3	p_2	p_1	p_4
p_4	p_1	p_2	p_3

There are three possible matchings:
(i) $\{(p_1,p_2),(p_3,p_4)\}$, (ii) $\{(p_1,p_3),(p_2,p_4)\}$ and (iii) $\{(p_1,p_4),(p_2,p_3)\}$.
However, (p_1,p_3) is a blocking pair for (i), (p_2,p_3) is a blocking pair for (ii), and (p_1,p_2) is a blocking pair for (iii).

5.3.2 The Complexity of the Stable Roommates Problem

The other difference from the stable marriage problem is in the workload of the algorithms. We recall that a stable marriage problem of size n is solvable in $\boldsymbol{O}(n^2)$. If a stable roommates matching exists then it can be found in polynomial time (Sethuraman and Teo, 2000); but if the lists contain ties of indifference then it becomes NP-complete, not computable within practical time constraints.

6 Conclusion

As a final section, we conclude with some comments on current work and applications.

6.1.1 The National Residents Matching Program

From its inception in 1952 up until 1997, the NRMP in the USA used the hospital-optimal matching mechanism to allocate residents to hospitals. Increasing pressure from student bodies who realised that they were disadvantaged by the algorithm led to the redesign of the matching mechanism in 1998, to derive allocations that are not optimal for either party, but considered fairer to both.

Similar schemes operate in Canada (the Canadian Resident Matching Service (CaRMS)), Scotland (the Scottish Foundation Allocation Scheme (SFAS)), Norway (the assignment of secondary school students to universities), and Singapore(primary school students to secondary schools).

The hospital-residents problem in the UK was more complicated because graduating medical students sought two posts, a medical post and a surgical post, posing a many-to-many matching problem. Schemes employed by various UK regions prior to 1999 suffered from the fact that either they did not produce stable matchings (such as the disbanded Newcastle and Birmingham schemes) or they took many days to compute a stable matching (such as the former Edinburgh scheme). The need for an efficient algorithm for use in a UK context was an open problem.

In 1998, Robert Irving described how techniques involving network flow and negative weight cycles can be used in order to solve this problem (Irving, 1998). Using his algorithm, the SPA matching scheme was designed and implemented at the University of Glasgow, and ran for the first time in the academic year 1999/2000.

6.1.2 Fairness

Unfairness, for one side or the other, is embedded in most known two-sided matching mechanisms. Recent work has focussed on the derivation of *fair* stable matchings, including the egalitarian solution (minimize the sum of the ranks of the participants) and the minimum-regret solution (maximize the welfare of the participant worst-off in the matching). These focus on socially optimal solutions, and ignore the issue of fairness for the individual.

A different approach is that of stable matching mechanisms that find a probability distribution over the set of stable matchings. Whenever an agent has the same probability to move at a certain point in the procedure that determines the final probability distribution, the random stable matching mechanism is considered to be “procedurally fair". (Klaus and Klijn, 2003)

6.1.3 Other Applications

The applications are many and diverse. Alvin Roth has made large contributions to the improvement of the college admissions process, and has also applied stable matching theory to the labour market. Yi-Cheng Zhang of Fribourg University in Switzerland applies it to economics, examining the marriage of supply with demand. In the case of roaming robots working on a space station, it can help determine the best way to distribute battery recharging stations. Theo Nieuwenhuizen of the University of Amsterdam has used the stable marriage problem in his studies of solid-state physics, measuring the energy costs of pairing particles.

Bibliography

Aigner, M. 1979, “Combinatorial Theory”; A Series of Comprehensive Studies in Mathematics, Springer-Verlag New York Inc.

Gale, D. and Shapley, L.S. 1962, “College Admissions and the Stability of Marriage”; The American Mathematical Monthly, Volume 69, No.1, pp 9-15

Garey, M.R. and Johnson, D.S. 1979, "Computers and Intractability: A Guide to the Theory of NP-Completeness"; W.H.Freeman and Company

Grätzer, G. 1971, “Lattice Theory”; W.H.Freeman and Company, San Francisco

Gusfield, D. and Irving, R.W. 1989, “The Stable Marriage Problem: Structure and Algorithms”; The MIT Press

Irving, R.W. 1989, "Matching medical students to pairs of hospitals: a new variation on a well-known theme"; Proceedings of ESA'98: the Sixth Annual European Symposium on Algorithms, vol. 1461 of Lecture Notes in Computer Science, pp. 381-392.

Klaus, B. and Klijn, F. 2003, “Procedurally Fair and Stable Matching”; Universitat Autònoma de Barcelona

Klaus, B. and Klijn, F. 2004, “Median Stable Matching for College Admissions”; Working Paper, Universitat Autònoma de Barcelona

Knuth, D.E. 1976, “Stable Marriage and Its Relation to Other Combinatorial Problems”; American Mathematical Society (English edition 1997)

Martínez, R., Massó, J., Neme, A. and Oviedo J. 2003, “An Algorithm To Compute The Full Set Of Many-To-Many Stable Matchings”; Working Paper, Universidad Nacional de San Luis and Universitat Autònoma de Barcelona

McVitie, D.G. and Wilson, L.B. 1970, “Stable Marriage Assignments for Unequal Sets”; BIT 10, 295-309.

Pittel, B. 1989, “The Average Number of Stable Matchings”; Society for Industrial and Applied Mathematics, SIDMA Volume 2 Issue 4, pp 530-549

Roth, A.E. 1982, “The Economics of Matching: Stability and Incentives”; Mathematics of Operations Research 7, 617-628.

Roth, A.E. and Sotomayor, M.A.O. 1990, “Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis”; Econometric Society Monograph Series. New York: Cambridge University Press.

Sethuraman, J., Teo, C-P. 2000 revised 2001, “A Polynomial-time Algorithm for the Bistable Roommates Problem”; Columbia University, NY and National University of Singapore

Sethuraman, J., Teo, C-P. and Qian, L. 2004 revised 2005, “Many-To-One Stable Matching: Geometry and Fairness”; Columbia University, NY

Weisstein, E.W. , “Stable Marriage Problem”, Mathworld – a Wolfram Web Resource, http://mathworld.wolfram.com/StableMarriageProblem.html