BIT 10 (1970), 295–309

STABLE MARRIAGE ASSIGNMENT FOR UNEQUAL SETS

D. G. McVITIE and L. B. WILSON

Abstract.

The theory and algorithms developed by the authors in a previous paper for the stable marriage assignment for equal sets are extended to the case of unequal sets. The main theorem in this paper shows that a person unmarried in one stable marriage solution will be unmarried in any other stable marriage solution.

1. Introduction.

The Stable Marriage concept of assigning the members of two disjoint sets to each other is defined as follows.

> If one set is considered to be men and the other set women and if we assign the men to the women in such a way that there exists no case where a man and a woman are not assigned to each other but who would *both* prefer each other to their present partners, then the assignment is said to form a set of stable marriages.

Gale and Shapley [1] introduced the idea of a stable marriage assignment and applied it to College entries in the U.S.A. The present authors [2] have extended the work of Gale and Shapley to the problem of finding all the possible sets of stable marriages, and they have also given detailed algorithms for solving these problems.

In our previous paper we confined our attention to equal numbers of men and women. In this paper we have removed this restriction and examined the stable marriage assignment problems when the sets of men and women are unequal. The algorithms developed previously have been extended to cover the case of unequal sets. The most important result proved in this paper is the theorem that shows that a person unmarried in any particular stable marriage solution will be unmarried in all the other stable marriage solutions. This theorem is useful in developing the algorithm for all the stable marriage solutions for unequal sets

Received February 23, 1970.

from that for equal sets. It will also influence the method of applying the stable marriage assignment methods to the problem of student entry to university, which it is hoped to discuss in a later paper.

2. Basic Concepts.

2.1. *Definition of a Stable Marriage.*

Consider two distinct sets A and B. An assignment of the members of A to the members of B is said to be a stable marriage if and only if, there exist no elements a and b (belonging to A and B respectively) which are not assigned to each other but who would both prefer each other to their present partners.

The sets must be distinct but the number of elements of each set need not be the same. The members of each set will list the members of the other set in order of preference. However, these choice lists need not be complete; for example if the candidates for University entrance are the members of one set, they will not be expected to list all the possible University courses in their choice list. It is useful in the theoretical work to consider the members of one set to be men and the members of the other to be women.

2.2. *Optimality.*

For any given sets A and B there are in general several stable marriage solutions. Three of these will be defined as follows:

Male optimal stable solution. This is the stable solution when every man is at least as well off under it as under any other stable solution.

Female optimal stable solution. This is similar except that the women get their best possible choices.

Minimum choice stable solution. In this stable solution the sum of the choice numbers of the men and the women is a minimum. This solution may not be unique but it provides a sort of unselfish optimal solution giving credit to low choice numbers in both sets.

The minimum choice solution may coincide with either the male optimal or female optimal solution. If the male and female optimal solutions are the same then there is only one solution.

3. Unequal numbers of men and women.

In the previous paper by McVitie and Wilson [2] the numbers of men and women were equal. This restriction will be removed and the effect on the algorithms discussed.

We will consider a marriage problem in which there are n men and k women. Let each man and woman list all the members of the opposite sex in order of preference as before. A stable set of marriages is required with as many men and women married as possible. Hence there will be $|n-k|$ men or women single after the stable marriages have been formed.

We wish to consider the following three situations.

(i) A particular stable solution is required, this usually being the male optimal solution.
(ii) Any single stable solution is required with no restrictions on which solution is obtained.
(iii) All the stable solutions are required.

3.1. *A particular single stable solution.*

We will consider in this section that the particular solution required is the male optimal solution. Obviously the female optimal solution can be obtained by analogous methods. The algorithms given in the previous paper [2] for equal sets of men and women can be applied for unequal sets if the stopping condition is altered. In the equal set algorithms the men proposed to the women and the male optimal solution was obtained. The stopping conditions can be modified so that if there are fewer men than women ($n < k$) the algorithm is stopped when n of the women have been proposed to, and if there are more men than women ($n > k$) the algorithm is stopped when each man is either being kept in suspense by some woman or has been refused by all the women.

The Recursive algorithm given for equal sets in [2] has been modified to deal with unequal sets. This modified algorithm is given in Appendix A. The main point to note is that in the sub-procedure *PROPOSAL* we must cater for the men running through all their choices.

3.2. *Any single stable solution.*

If we look at the algorithm discussed in the previous section for finding the male optimal solution we see it works well when there are less men than women. However, when the men are more numerous, several men will have to exhaust their choice lists since they will not be chosen by any woman and will be the men who end up unmarried.

Clearly a more efficient algorithm for finding a single stable marriage would operate such that the less numerous set proposed. Thus, if there are less men than women, the men propose and the male optimal solu-

tion is obtained. However, if there are more men than women, the women propose and the female optimal solution is obtained.

The algorithm (Appendix A) which gives the male optimal solution could be used for this case by making sure that when called with more men than women ($n > k$) the actual parameters corresponding to malechoice, femalechoice, n and k were in fact femalechoice, malechoice, k and n respectively. However, a separate algorithm, given in Appendix B, has been written for finding any single stable solution. This procedure determines which is the smaller set and the smaller set then proposes. Since the smaller set always proposes there is no longer any need to check for proposers exhausting their choice lists.

3.3. *All the stable solutions.*

The following very useful and interesting theorem enables the multiple solution algorithm described by the authors for equal sets [2] to be applied to the stable marriage problem for unequal sets.

THEOREM. *In a marriage problem of n men and k women if any person is unmarried in one stable marriage solution he or she will be unmarried in all the stable solutions.*

PROOF. Let the larger set propose. Then $|k-n|$ members of this set will be single at the first stable solution obtained by a Gale and Shapley type method, and these people will have proposed to all their choices. Now by a theorem proved by Gale and Shapley [1] (see also theorem 1 in reference [2]) this stable solution is optimal for the larger set. Therefore, those single in the first stable solution can do no better than remain single in any other stable solution.

Using the above theorem the algorithm for finding the stable marriage solutions for n men and k women can be formulated. The algorithm, which is given in Appendix C, is similar to that for finding all the stable marriage solutions for equal sets given by the authors [2]. The theorem given above is followed in the algorithm by letting the larger set propose. Thus when the first stable marriage solution is found it is the optimum for the larger set. As the unmarried members of the larger set have made all their proposals (and had them all rejected) they play no further part in the algorithm. The rest of the algorithm after the first stable marriage solution is essentially the same as for the equal set case. Thus the subprocedures "breakmarriage", "proposal" and "refusal" are apart from a few slight changes the same as those given in [2] for equally sized sets.

The minimum choice stable solution can be deduced from the procedure by printing out the sum of the choice numbers for each stable solution.

4. Results and Discussion.

The algorithms (Appendices A, B and C) were tested in Whetstone Algol on the KDF9 computer for several different sized sets of men and women. In our earlier paper [2] the results were given for a case of 8 men and 8 women. The algorithm for finding all the stable marriage solutions for equal sets gave 9 stable solutions in this case. In order to test the unequal set case the original problem was altered in various ways. First, two extra men were added to the problem to get the choice matrices shown in Table 1. The extra men 9 and 10 were added to the female choice matrix so that they were always a worse choice than the woman's choice in the male optimal solution. The male optimal are indicated by the asterisks in Table 1. The choice lists of the men 9 and 10 were arbitrary. The results for these choice matrices are given in Table 2 and as expected are the same nine solutions $S_1S_2 \ldots S_9$ as obtained previously for the 8×8 case with men 9 and 10 unmarried.

Table 1. *Number of men* = 10. *Number of women* = 8.

Men choose women in order

Man	1 chooses	5	7	1	2	6	8	4	3
Man	2 chooses	2	3	7	5	4	1	8	6
Man	3 chooses	8	5	1	4	6	2	3	7
Man	4 chooses	3	2	7	4	1	6	8	5
Man	5 chooses	7	2	5	1	3	6	8	4
Man	6 chooses	1	6	7	5	8	4	2	3
Man	7 chooses	2	5	7	6	3	4	8	1
Man	8 chooses	3	8	4	5	7	2	6	1
Man	9 chooses	1	6	7	4	2	5	8	3
Man	10 chooses	7	4	5	8	2	1	3	6

Women choose men in order

Woman	1 chooses	5	3	7	6*	9	10	1	2	8	4
Woman	2 chooses	8	6	3	5	7*	2	1	10	9	4
Woman	3 chooses	1	5	6	2*	10	4	9	8	7	3
Woman	4 chooses	8*	7	3	9	2	4	1	5	6	10
Woman	5 chooses	6	4	7	3	8	1*	10	9	2	5
Woman	6 chooses	2	8	5	4*	6	3	9	7	1	10
Woman	7 chooses	7	5*	10	9	2	1	8	6	4	3
Woman	8 chooses	7	4	1	5	2	3*	9	10	6	8

Table 2. *Stable solutions for the case of* 10 *men and* 8 *women.*

Man		S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
1	marries	5	8	3	3	3	3	8	8	8
2	marries	3	3	6	6	6	6	3	3	3
3	marries	8	5	5	1	2	1	1	2	1
4	marries	6	6	8	8	8	8	6	6	6
5	marries	7	7	7	7	1	2	7	1	2
6	marries	1	1	1	5	5	5	5	5	5
7	marries	2	2	2	2	7	7	2	7	7
8	marries	4	4	4	4	4	4	4	4	4
Number of proposals		32	38	47	51	59	54	42	50	45
Choice Count		64	65	67	66	70	67	64	68	65

Table 3. *Number of men* = 8. *Number of women* = 11.

Men choose women in order

Man	1 chooses	5	7	1	2	6	8	4	3*	9	10	11
Man	2 chooses	2	3	7	5	4	1	8	6*	10	11	9
Man	3 chooses	8	5	1	4	6	2*	3	7	11	9	10
Man	4 chooses	3	2	7	4	1	6	8*	9	10	5	11
Man	5 chooses	7	2	5	1*	11	9	3	6	10	8	4
Man	6 chooses	1	6	7	5*	9	11	10	8	4	2	3
Man	7 chooses	2	5	7*	11	10	6	9	3	4	8	1
Man	8 chooses	3	8	4*	5	9	7	10	2	6	11	1

Women choose men in order

Woman	1 chooses	5	3	7	6	1	2	8	4
Woman	2 chooses	8	6	3	5	7	2	1	4
Woman	3 chooses	1	5	6	2	4	8	7	3
Woman	5 chooses	8	7	3	2	4	1	5	6
Woman	5 chooses	6	4	7	3	8	1	2	5
Woman	6 chooses	2	8	5	4	6	3	7	1
Woman	7 chooses	7	5	2	1	8	6	4	3
Woman	8 chooses	7	4	1	5	2	3	6	8
Woman	9 chooses	2	3	8	6	4	1	5	7
Woman	10 chooses	8	6	3	2	1	4	7	5
Woman	11 chooses	6	3	5	2	7	1	8	4

The second alteration to the original problem was to add three extra women and obtain the choice matrices shown in Table 3. The extra three women 9, 10 and 11 were added to the malechoice matrix in such a way that they were always a worse choice than the man's choice in the female optimal stable solution. Also the choice lists of the extra women 9, 10 and 11 were arbitrary. As before the nine stable solutions were obtained

and the extra women 9, 10 and 11 were unmarried. However, since the larger set always propose, in this case the women did the proposing and so the solutions were obtained in a different order as shown in Table 4. In comparison with Table 2 the order is S_5, S_6, S_3, S_2, S_1, S_4, S_7, S_9, S_8 with the female optimal solution first.

Table 4. *Stable solutions for the case of* 8 *men and* 11 *women.*

Man									
1	3	3	3	8	5	3	8	8	8
2	6	6	6	3	3	6	3	3	3
3	2	1	5	5	8	1	1	1	2
4	8	8	8	6	6	8	6	6	6
5	1	2	7	7	7	7	7	2	1
6	5	5	1	1	1	5	5	5	5
7	7	7	2	2	2	2	2	7	7
8	4	4	4	4	4	4	4	4	4
9, 10	Single								
Number of proposals	35	37	44	51	56	39	46	44	42
Choice Count	78	75	75	73	72	74	72	73	76

The female choice matrix given in Table 1 was now altered by changing the choice list of woman 3 to be

1 5 10 6 2 4 9 8 7 3

Thus we advance man 10 in her choice list so that he is now between man 1 (her choice in the female optimal solution, S_5) and man 2 (her choice in the male optimal solution, S_1). The solutions obtained were S_3, S_4, S_5, S_6. The other five solutions obtained previously are now unstable. We can see this by looking at the previous male optimal solution S_1. In this solution woman 3 is married to man 2, but she would prefer man 10 who is unmarried, and man 10 would prefer woman 3 to being single. Therefore S_1 is now unstable.

This example introduces an interesting point that even though the extra men 9 and 10 do not get married in any stable marriage solution their presence can effect the allowable stable solutions. Originally we attempted to make our algorithm more efficient by letting the smaller set propose, finding the optimal stable solution for the smaller set, then discarding all the unmarried members of the larger set and treating the problem as one of equal size sets. This example shows why such an

algorithm is not correct and would give the invalid solutions S_1, S_2, S_7, S_8 and S_9.

The algorithms *MOSM* (Appendix A) and *SM* (Appendix B) were tested with the data of Tables 1 and 3. The algorithm *MOSM* gave the solution S_1 in each case. The algorithm *SM* gave the solution S_5 for the data of Table 2 and Solution S_1 for the data of Table 3.

It is hoped to discuss in a later paper the application of the stable marriage assignment theory to the entry of students through the University Central Council for Admissions (UCCA) system. However, two interesting points emerge from the present paper. First, the basic theorem shows that no matter what stable marriage assignment is used those who are not acceptable in one stable marriage assignment will not be in any other. Hence, no matter what strategy is used, student optimal, university optimal or something in between, the same students will get places. However, different strategies may assign them to different universities. The second point is that although some applicants are not accepted their positions in the university choice lists can affect the final assignment. They have a sort of nuisance value which can mean that stable solutions obtained when such applicants are completely ignored are no longer stable.

Acknowledgement.

Some of this work was done as a dissertation prepared as part of an M. Sc. course by one of the authors (D. G. M.). Financial support for the course was obtained from the Science Research Council.

Appendix A.

Algorithm for male optimal stable marriage assignment for unequal sets.

The procedure *MOSM* finds the male optimal stable marriage solution for assigning the members of two distinct sets to each other. One set is considered to be all men and is of size n, the other set to be all women and of size k. The men always propose. The theory used in this algorithm is explained in Section 3.1. It is similar to and based on the algorithm *MW* for equal sets given by McVitie and Wilson [2].

```
procedure MOSM(malechoice, femalechoice, marriage, count, n, k);
value n, k; integer count, n, k;
```

```
integer array malechoice, femalechoice, marriage;
comment this procedure finds the male optimal stable marriage (MOSM)
  solution. It leaves the result in the integer array marriage. Thus marriage
  [i] is the man whom the i-th woman marries. There are n men and k women
  thus the integer arrays have the following sizes, malechoice[1:n, 1:k],
  femalechoice[1:k, 1:n], marriage[1:k]. If k>n (more women than men)
  then some women will be unmarried and those k-n women will have the
  appropriate elements of marriage zero, e.g. if marriage[i]=0 at the end
  then woman i is unmarried.

    malechoice and femalechoice are the choice matrices for the men and
  women respectively, i.e. malechoice[i,j] is the j-th choice of the i-th man.
  count is the number of proposals made before the stable marriage is found.
  The femalechoice array is changed to the integer array fc, where fc[i,j] is
  the choice number (first, second, ...) of the j-th man to woman i. This
  new arrangement is more convenient when the women compare proposals
  in the procedure REFUSAL. All the women keep a dummy man 0 in
  suspense initially, this dummy man is given a choice number n+1 so
  he will be given up as soon as any other offer is made;
begin integer i, j;
  integer array fc[1:k, 0:n], malecounter[0:n];
  procedure PROPOSAL(i); value i; integer i;
  comment this procedure makes the next proposal for man i, and calls
    the procedure REFUSAL to see what effect this proposal will have.
    The procedure does nothing if man i is the dummy man 0 or if man i
    has exhausted all his choice list;
  if i ≠ 0 ∧ malecounter[i] < k+1 then
  begin integer j; count := count+1;
    j := malecounter[i]; malecounter[i] := j+1;
    REFUSAL(i, malechoice[i,j])
  end;
  procedure REFUSAL(i,j); value i, j; integer i, j;
  comment this procedure decides whether woman j should keep the man
    she is holding in suspense in marriage[j] or man i who has just pro-
    posed to her. Whichever she rejects goes back to the procedure PRO-
    POSAL to make his next proposal;
  if fc[j, marriage[j]] > fc[j,i] then
  begin integer l;
    l := marriage[j]; marriage[j] := i;
    PROPOSAL(l)
  end else PROPOSAL(i);
  for i := 1 step 1 until k do
```

```
  begin for j := 1 step 1 until n do
    fc[i, femalechoice[i,j]] := j;
    marriage[i] := 0; fc[i, 0] := n + 1
  end;
  for i := 1 step 1 until n do malecounter[i] := 1;
  count := 0;
  for i := 1 step 1 until n do PROPOSAL(i);
  comment this for statement operates the algorithm;
end of procedure MOSM;
```

Appendix B.

Algorithm for single stable marriage assignment for unequal sets.

The procedure *SM* finds a single stable marriage solution for assigning the members of two distinct sets to each other. One set is considered to be all men and is of size n, the other set to be all women and of size k. The smaller set proposes. Thus, if the men are less numerous they propose and the male optimal stable solution is obtained.

In the comments of this procedure the following notation,

man/woman

is used to indicate an alternative. The first alternative is selected if there are less men than women, i.e. the men are proposing.

The theory used in this algorithm is explained in Section 3.2.

```
procedure SM(malechoice, femalechoice, marriage, count, n, k);
value n, k; integer count, n, k;
integer array malechoice, femalechoice, marriage;
comment this procedure finds a single stable marriage (SM) solution.
  There are n men and k women, and the smaller set proposes. The optimal
  stable solution for the smaller set is obtained. The result is left in the
  integer array marriage. Thus marriage[i] is the man whom the i-th woman
  marries if n ≦ k, but if there are less women then marriage[i] is the woman
  whom the i-th man marries. If marriage [i] = 0 at the end then that per-
  son is unmarried. There will be abs (n - k) elements of marriage zero.
    malechoice and femalechoice are the choice matrices for the men and
  women respectively, i.e. malechoice[i,j] is the j-th choice of the i-th man.
  count is the number of proposals made before the stable marriage is found.
```

```
  The formal integer arrays should have the following sizes, malechoice
  [1:n, 1:k], femalechoice[1:k, 1:n], marriage[1:max(n,k)];
begin integer i, j, max, min; boolean menlarger;
  menlarger := n > k;
  max := if menlarger then n else k;
  min := if menlarger then k else n;
  begin integer array ch no[1:max, 0:min], counter[0:min];
    procedure PROPOSAL(i, choice); value i; integer i;
    integer array choice;
    comment this procedure makes the next proposal for man/woman i,
      and calls the procedure REFUSAL to see what effect this proposal
      will have. The procedure does nothing if man/woman is the dummy 0;
    if i ≠ 0 then
    begin integer j; count := count+1;
      j := counter[i]; counter[i] := j+1;
      REFUSAL(i, choice[i,j], choice)
    end;
    procedure REFUSAL(i, j, choice); value i, j; integer i, j;
    integer array choice;
    comment this procedure decides which of the two proposals, the one
      being kept in suspense or the one just received, should be retained.
      Whichever is rejected goes back to the procedure PROPOSAL to
      make the next proposal;
    if ch no[j, marriage[j]] > ch no[j, i] then
    begin integer l;
      l := marriage[j]; marriage[j] := i;
      PROPOSAL(l, choice)
    end else PROPOSAL(i, choice);
    for i := 1 step 1 until max do
    begin for j := 1 step 1 until min do
      ch no[i, if menlarger then malechoice[i, j]
        else femalechoice[i, j]] := j;
      marriage[i] := 0; ch no[i, 0] := min+1
    end;
    for i := 1 step 1 until min do counter[i] := 1;
    comment the femalechoice/malechoice array is changed to the integer
      array ch no, where ch no[i,j] is the choice number (first, second, ...)
      of the j-th man/woman to woman/man i. This new arrangement is
      more convenient when the women/men compare proposals in the
      procedure REFUSAL. All the women/men keep a dummy 0 in sus-
```

```
        pense initially, this dummy is given a choice number min(n,k)+1
        so that it will be given up as soon as any other offer is made;
    count := 0;
    for i := 1 step 1 until min do
    if menlarger then PROPOSAL(i, femalechoice)
    else PROPOSAL(i, malechoice);
    comment this for statement operates the algorithm, the smaller set
    proposing;
  end
end of procedure SM;
```

Appendix C.

Algorithm for all the stable marriage assignments for unequal sets.

The procedure *ALL STABLE MARRIAGES RECT* finds all the possible stable marriage solutions for assigning the members of two distinct sets to each other. The sets are considered to be of sizes n and k with $n \geqq k$. The larger set propose. This means when using the procedure we must call the parameters in the correct order. For example, suppose the actual parameters are,

m — the number of men
w — the number of women
malechoice — the choice matrix of the men
femalechoice — the choice matrix of the women
P — procedure name corresponding to the formal parameter *STABLE MARRIAGE*

If there are more men than women we call the procedure by the statement *ALL STABLE MARRIAGES RECT* (*malechoice*, *femalechoice*, *m*, *w*, *P*); however, if there are more women than men the procedure call should be, *ALL STABLE MARRIAGES RECT* (*femalechoice*, *malechoice*, *w*, *m*, *P*);

The actual procedure (*P* in the above example) which corresponds to the formal parameter *STABLE MARRIAGE* has to be written by the user. This procedure will be entered each time a new stable marriage solution has been formed, and the new answer will be in the parameter "marriage" whilst the parameter "count" will contain the number of proposals made. What the user writes in the actual procedure will de-

pend on how he intends to use the stable marriage solutions when they are found.

In the **comments** in the procedure the following notation,

man/woman

has been adopted to indicate an alternative. The first alternative is selected if there are more men than women, i.e. the men are proposing.

The theory used in this algorithm is explained in Section 3.3. A previous algorithm *ALL STABLE MARRIAGES* was given by McVitie and Wilson [2] for the case of equal sets and many of the ideas in it carry over to the unequal set case.

```
procedure ALL STABLE MARRIAGES RECT(choicenk, choicekn, n, k,
                                    STABLE MARRIAGE);
  value n, k, choicekn; integer n, k;
integer array choicenk, choicekn; procedure STABLE MARRIAGE;
comment this procedure finds all the stable marriage solutions. It assumes
  that n≧k and choicenk is the choice matrix of the larger set n, and
  choicekn is the choice matrix of the smaller set k. The larger set n pro-
  poses. STABLE MARRIAGE(marriage, n, k, count) is a procedure
  with 4 parameters which has to be written by the user and which is en-
  tered when a new stable marriage solution is formed after count proposals.
  The integer array marriage contains the solution. Thus marriage[i] is
  the member of the larger set whom the i-th member of the smaller set
  marries;
begin procedure breakmarriage(malecounter, marriage, i, count);
  value malecounter, marriage, i, count;
  integer i, count; integer array malecounter, marriage;
  comment this procedure breaks the marriage of person i;
  begin integer j;
    marriage[choicenk[i, malecounter[i]-1]] := -i;
    proposal(i, malecounter, marriage, count);
    if ¬ success then goto EXIT;
    STABLE MARRIAGE(marriage, n, k, count);
    for j := i step 1 until n-1 do
    breakmarriage(malecounter, marriage, j, count);
    comment the lower limit i in the above for statement is the application
      of the rule which states that after a successful breakmarriage opera-
      tion on man i we restrict further breakmarriages to men≧i;
    for j := i+1 step 1 until n-1 do
    unchanged[j] := true;
```

```
    EXIT: unchanged[i] := false;
end of breakmarriage;
procedure proposal(i, malec, marriage, c); value i;
integer i, c; integer array malec, marriage;
comment in this procedure man/woman i proposes to the next woman/man
    in his/her choice list, and calls the procedure refusal for this woman/
    man. If i is negative on entry then a successful breakmarriage opera-
    tion has been completed and a new stable marriage found. If the boolean
    success is made false during a breakmarriage operation then it means
    that this breakmarriage has failed;
if i < 0 then success := true
else if i = 0 ∨ malec[i] = k+1 ∨ ¬unchanged[i]
then success := false
else begin c := c+1; j := malec[i]; malec[i] := j+1;
        refusal (i, choicenk[i, j], malec, marriage, c)
    end of proposal;
procedure refusal(i, j, malec, marriage, c); value i, j;
integer i, j, c; integer array malec, marriage;
comment this procedure decides which of the two proposals, the one being
    kept in suspense or the one just received, should be retained. Whichever
    is rejected goes back to the procedure proposal to make the next proposal;
if choice[j, abs(marriage[j])] > choice[j, i] then
begin integer s; s := marriage[j];
    marriage[j] := i; proposal(s, malec, marriage, c);
end else proposal(i, malec, marriage, c);
integer array marriage[1:k], counter[0:n],
    choice[1:k, 0:n]; boolean array unchanged[0:n];
integer i, j, count, t; boolean success;
comment the boolean array unchanged is used to ensure that during a
    breakmarriage operation started on man i only men ≧ i propose. The
    boolean success is set true if breakmarriage to man i leads to a new
    stable marriage solution, otherwise it is set false. The integer array
    counter keeps a record of the proposals made by the larger set;
for i := 1 step 1 until k do
begin for j := 1 step 1 until n do
    choice[i, choicekn[i, j]] := j;
    choice[i, 0] := n+1; marriage[i] := 0
end;
comment the choice matrix of the smaller set choicekn is rearranged in
    the array choice so that comparisons of proposals in the procedure
    refusal can be conveniently made. A dummy person 0 is assigned to
```

```
    the array marriage initially and is given the choice number n+1 so
    that when another offer is made the dummy person will be rejected.;
  for i := 1 step 1 until n do
  begin counter[i] := 1; unchanged[i] := true
  end;
  count := 0;
  for i := 1 step 1 until n do
  proposal(i, counter, marriage, count);
  comment the larger set have proposed and their optimal stable solution
    found;
                STABLE MARRIAGE(marriage, n, k, count);
  comment count containing the number of proposals made to date;
  for i := 1 step 1 until n-1 do
  breakmarriage(counter, marriage, i, count)
end of ALL STABLE MARRIAGES RECT;
```

REFERENCES

1. D. Gale and L. S. Shapley, *College Admissions and the Stability of Marriage*, Amer. Math. Monthly 69 (1962), 9–15.
2. D. G. McVitie and L. B. Wilson, *The Stable Marriage Problem*, C.A.C.M. (to be published).

I. C. L.
KIDSGROVE
STOKE-ON-TRENT, ENGLAND

AND

UNIVERSITY COMPUTING LABORATORY
UNIVERSITY OF NEWCASTLE UPON TYNE
ENGLAND