Discrete Applied Mathematics 48 (1994) 261–272
North-Holland

Stable marriage and indifference

Robert W. Irving

Computing Science Department, University of Glasgow, Glasgow G12 8QQ, UK

Received 30 July 1989
Revised 5 February 1990

Abstract

It is well known that every instance of the classical stable marriage problem admits at least one stable matching, and that such a matching can be found in $O(n^2)$ time by application of the Gale/Shapley algorithm. In the classical version of the problem, each person must rank the members of the opposite sex in strict order of preference.

In practical applications, a person may not wish (or be able) to choose between alternatives, thus allowing ties in the preference lists (or, more generally, allowing each preference list to be a partial order). With the introduction of such *indifference*, the notion of stability may be generalised in three obvious ways. For the weakest extension of stability, the same existence result holds, and essentially the same algorithm may be applied. In the other two cases, however, there is no guarantee that stable matchings exist. Nonetheless, in this paper, we describe polynomial-time algorithms that will establish, in either of these two cases, whether a matching of the appropriate kind exists, and if so will find such a matching.

Keywords. Stable marriage, matching, polynomial-time algorithms.

1. Introduction

An instance of the classical *stable marriage problem* involves n men and n women, each of whom ranks all the members of the opposite sex in strict order of preference. A complete matching of the men and women is *unstable* if there is a man m and a woman w who are not partners but each of whom prefers the other to his/her partner in the matching. Such a pair is said to *block*, or to be a *blocking pair* for, the matching. Naturally, a matching for which there is no blocking pair is said to be *stable*. We use the term in other obvious ways—a man-woman pair (m, w) is a *stable pair* if there is some stable matching in which m and w are partners, in which case each is a *stable partner* of the other.

A generalisation of the stable marriage problem, usually referred to as the *stable roommates problem*, involves a single set of $2n$ people, each of whom ranks all of the

SSDI 0166-218X(92)00179-Z

others in strict order of preference. The definition of a blocking pair and of a stable matching is analogous to that for the stable marriage case.

The classical results in stable marriage are the Gale/Shapley theorem and algorithm [1]—the theorem asserts that at least one stable matching exists for every stable marriage instance, and the algorithm finds such a matching in $O(n^2)$ time. For the stable roommates problem, there is no guarantee that a stable matching exists for any particular instance, but in this case, there is also an $O(n^2)$ algorithm that will determine such a matching if one exists [3].

Much progress has recently been made in understanding the mathematical structure underlying stable matching problems of this kind, and this understanding has led to new efficient algorithms for a range of associated problems. A comprehensive discussion of this work appears in [2].

Instances of the stable marriage problem arise in a number of practical contexts where algorithms such as that of Gale and Shapley are directly applicable. Many of the practical applications in this area are covered in [8]. However, in practical instances, it is perhaps unrealistic to expect each participant to provide a *strict* preference ordering of all of his/her prospective partners. If indifference is permitted, so that each person's preference list becomes a partial order, the character of the problems changes somewhat. We concentrate our attention on the particular case in which each person's preference list is allowed to contain ties, but it is not hard to show that all of the results can be extended to the general case of partially ordered preference lists.

We consider three possibilities for the appropriate definition of stability in this setting. A matching will be called *weakly stable* unless there is a couple each of whom strictly prefers the other to his/her partner in the matching. It is not hard to see that, in the stable marriage case, if ties are broken arbitrarily, any matching that is stable in the resulting (strict) instance is weakly stable in the original instance. So the Gale/Shapley algorithm (or other algorithms that find a stable matching in the classical case) may be applied in this context to find a weakly stable matching. On the other hand, it has recently been proved [7], perhaps rather surprisingly, that the stable roommates problem with ties in NP-complete under this notion of weak stability.

Two alternative, stronger notions of stability also suggest themselves. A matching is *strongly stable* if there is no couple x, y such that x strictly prefers y to his/her partner, and y either strictly prefers x to his/her partner or is indifferent between them. A matching is *super-stable* if there is no couple each of whom either strictly prefers the other to his/her partner or is indifferent between them.

It is easy to find instances of both stable marriage with ties and stable roommates with ties for which no strongly stable matching and/or no super-stable matching exists. For example, in a situation of complete indifference, in which no person has any strict preferences, there obviously cannot be a super-stable matching. In the instance of size 2 described by the preference lists in Fig. 1, in which all preferences are strict except those of man 2 (indicated by the parantheses in the preference list of man 2), there is no strongly stable matching. For the matching $\{(1, 1), (2, 2)\}$ is blocked by the pair $(2, 1)$, while the matching $\{(1, 2), (2, 1)\}$ is blocked by the pair $(2, 2)$.

1 1 2	1 2 1
2 (1 2)	2 2 1
Men's preferences	Women's preferences

Fig. 1. An instance with no strongly stable matching.

It is our purpose in this paper to describe polynomial-time algorithms, for the stable marriage problem with ties, to determine whether a strongly stable or super-stable matching exists, and if so to find such a matching. The corresponding problems for stable roommates, however, remain open.

2. The (extended) Gale/Shapley algorithm

As a prelude to the algorithms for super-stable and strongly stable matchings, we describe an extended version of the classical Gale/Shapley algorithm that forms the basis of each of the new algorithms. The proof of the correctness of this algorithm may be found in [2]. Alternatively, since each of Algorithms SUPER and STRONG of Sections 3 and 4 reduces to it in the special case of strict preferences, the proofs of correctness of these algorithms apply, in simplified form, to the extended Gale/Shapley algorithm.

Informally, the algorithm may be expressed in terms of a sequence of "proposals" from the men to the women (or from the women to the men if the roles of men and women are reversed throughout). At any point during the algorithm's execution, each person is either *engaged* or *free*; each man may alternate between being engaged and being free, but once a woman is engaged, she is never again free, although the identity of her fiancé may change. A man who becomes engaged more than once obtains fiancées who are successively less desirable to him, while each successive engagement brings a woman a more favoured partner.

When a woman receives a proposal she will immediately (but provisionally) accept it, becoming engaged to the proposer, simultaneously breaking her current engagement, if any, and setting her previous fiancé free. Each proposal in the sequence is made by a free man m to the first woman w on his current preference list, and results in the deletion from w's list of all of the successors of m, and of w from the list of each of these men. We shall refer to the *deletion of the pair* (m, w). It is in respect of this reduction of the preference lists that the algorithm described is an extension of the classical Gale/Shapley algorithm.

The proposal sequence continues until everyone is engaged, a state of affairs that is bound to arise before any man's list becomes empty. This is because, if some man m's list becomes empty, then every woman must be engaged to a man whom she prefers to m, giving a contradiction since no man is multiply engaged and the numbers of men and women are equal. On termination, the engaged pairs constitute a stable matching. Notice that the algorithm involves an element of nondeterminism, since the order in

```
assign each person to be free;
while some man m is free do
begin
  w := first woman on m's list;
  m proposes, and becomes engaged, to w;
  if some man p is engaged to w then
     assign p to be free;
  for each successor m' of m on w's list do
    delete the pair (m', w)
end;
output the engaged pairs, which form a stable matching
```

Fig. 2. The extended Gale/Shapley algorithm, Algorithm EGS.

which the free men propose is not specified. However, as is implicit in the statement of Theorem 2.1 below, this nondeterminism is immaterial to the outcome of the algorithm.

Figure 2 contains a description of the algorithm, which we shall refer to as Algorithm EGS.

The following theorem summarises the effect of this extended Gale/Shapley algorithm.

Theorem 2.1. *For every instance of the stable marriage problem with strict preferences, every execution of Algorithm EGS finds the unique stable matching in which every man has his best possible, and every woman her worst possible, stable partner (the so-called man-optimal / woman- "pessimal" stable matching).*

Analysis of the extended Gale/Shapley algorithm. It is not hard to devise a data structure that provides a faithful representation of the current preference lists throughout the execution of the algorithm, and such that the operations of locating one person in the list of another, deleting a pair, locating the first or last entry in a list, and locating the successor of an entry in a list can all be achieved in constant time.

With such a data structure, the total number of operations carried out during the execution of the extended Gale/Shapley algorithm is bounded by a constant times the number of pairs deleted plus the number of engaged pairs, and since the total number of man-woman pairs is n^2, the algorithm has an $O(n^2)$ worst-case time bound.

An $\Omega(n^2)$ lower bound has recently been established for the problem of finding a stable matching in the stable marriage problem [5], so that the (extended) Gale/Shapley algorithm is asymptotically optimal in the worst case.

3. Super-stable matchings

We extend the terminology of the classical stable marriage problem to the case of super-stable matchings. For example, a man-woman pair (m, w) is a *super-stable pair* if

there is some super-stable matching in which they are partners; each is said to be a *super-stable partner* of the other. Further, when we use the terms *block* or *blocking pair* in this section, it is in the sense of super-stability.

It turns out that the extended Gale/Shapley algorithm of Section 2 can be amended in a quite straightforward way to find a super-stable matching, or to establish that no super-stable matching exists. The proposal sequence proceeds in the usual way, except that a free man who has two or more women tied at the head of his current list proposes to all of them simultaneously. When a woman receives a proposal, all men strictly inferior to the proposer are deleted from her list, and she from theirs, but she may hold more than one proposal if the men in question are tied in her list. The proposal sequence may terminate with one or more of the men's lists empty, in which case no super-stable matching exists. Otherwise, each man will be engaged to one or more women, and the fiancé(e)s of any multiply engaged person must be tied in his/her list.

As we will show, no woman who is multiply engaged at this point can have a super-stable partner from among any of her fiancés, nor from among any men who may be tied with them in her list. So all such pairs may be deleted and the proposal sequence re-activated. The whole process is repeated until it produces a one-one engagement mapping, which will be a super-stable matching, or until some man's list becomes empty, indicating that no super-stable matching is possible.

The algorithm appears as Algorithm SUPER in Fig. 3. Note that, by the *head* of a man's (current) list, we mean the set of one or more women, tied in his list, whom he

```
assign each person to be free;
repeat
  while some man m is free do
    for each woman w at the head of m's list do
    begin
      m proposes, and becomes engaged, to w;
      for each strict successor m' of m on w's list do
      begin
        if m' is engaged to w then
          break the engagement;
        delete the pair (m', w)
      end
    end;
  for each woman w who is multiply engaged do
  begin
    break all engagements involving w;
    for each man m at the tail of w's list do
      delete the pair (m, w)
  end;
until (some man's list is empty) or (everyone is engaged);
if everyone is engaged then
  the engagement relation is a super-stable matching
else
  no super-stable matching exists
```

Fig. 3. Algorithm SUPER.

strictly prefers to all of the others in the list. Similarly, by the *tail* of a woman's (current) list, we mean the set of one or more men, tied in her list, to whom she strictly prefers all of the others in the list.

Correctness of Algorithm SUPER. We now establish the correctness of Algorithm SUPER with the aid of a number of lemmas.

Lemma 3.1. *If the pair* (m, w) *is deleted during an execution of Algorithm SUPER, then that pair cannot block any matching consisting of pairs that are never deleted.*

Proof. This is an immediate consequence of the fact that, in w's list, m is a strict successor of any undeleted entries. □

Lemma 3.2. *A matching generated by Algorithm SUPER is super-stable.*

Proof. Suppose that some execution of Algorithm SUPER generates matching M, and suppose that matching M is blocked by the pair (m, w). By Lemma 3.1, the pair (m, w) cannot have been deleted. So, if (m, w) blocks M, w must be at the head of m's list, and so must be engaged to m. But the engagement mapping is one-one, so that m and w are partners in M, giving a contradiction. □

To complete the proof of the correctness of the algorithm, it will suffice to show that no super-stable pair is ever deleted during its execution. For then the algorithm can never fail to find a super-stable matching if one exists.

Lemma 3.3. *No super-stable pair is ever deleted during an execution of Algorithm SUPER.*

Proof. Suppose, for a contradiction, that (m, w) is the first super-stable pair deleted during some execution of the algorithm. Let M be a super-stable matching in which m and w are partners.

Case 1. Suppose that (m, w) is deleted as a result of some other man, say m', becoming engaged to w. Then w strictly prefers m' to m. Also, there is no super-stable matching in which m' has a partner, say w', whom he strictly prefers to w; for then the super-stable pair (m', w') would have been deleted before (m, w). So the supposed super-stable matching M is blocked by (m', w), giving a contradiction.

Case 2. Suppose that (m, w) is deleted as a result of w being multiply engaged. If m' is any man who was engaged to w at that point, then m' cannot have a super-stable partner w' whom he strictly prefers to w, for the same reason as before. So once again the supposed super-stable matching M is blocked by the pair (m', w), giving a contradiction. □

If Algorithm SUPER finds a super-stable matching, then, because of Lemma 3.3, no man can have a strictly better partner, or even a different partner of equal ranking, in any other super-stable matching. So, if super-stable matchings exist for a given problem instance, the algorithm always finds the one that is man-optimal in this precise sense. It follows that the nondeterminism arising from failure to specify the order in which the free men propose is immaterial to the outcome. Further, it is easy to see that man-optimal is woman-pessimal, in the sense that no woman can have a worse partner, or even a different partner of equal ranking, in any super-stable matching.

We summarise our findings as a theorem.

Theorem 3.4. *For every instance of stable marriage with ties, Algorithm SUPER determines whether or not a super-stable matching exists. If such a matching does exist, all possible executions of the algorithm find the one in which every man has his strictly best, and every woman her strictly worst, super-stable partner.*

Analysis of Algorithm SUPER. As in the case of the extended Gale/Shapley algorithm, the total number of operations carried out during the execution of Algorithm SUPER is essentially bounded by a constant times the number of pairs deleted, and so the algorithm has an $O(n^2)$ worst-case time bound. Furthermore, since the problem reduces to classical stable marriage in the case of strict preferences, the $\Omega(n^2)$ lower bound of [5] applies.

4. Strongly stable matchings

In this section we use the terms *strongly stable pair* and *strongly stable partner* with the obvious meaning, and the notion of a *blocking pair* is in the sense of strong stability.

The problem of finding a strongly stable matching, if one exists, can be solved by a somewhat more elaborate extension of the Gale/Shapley algorithm. Each proposal sequence proceeds, as in the super-stable case, until some man's list becomes empty, in which case there can be no strongly stable matching, or until all the men are engaged. In the latter case, if the bipartite graph representing the engaged pairs contains a perfect matching—i.e., if all the men and women can be paired off on a one-one basis with a fiancé(e)—then the resulting matching is strongly stable. To explain how the algorithm proceeds if there is no such perfect matching, we need some terminology and some basic results dealing with matchings in bipartite graphs.

It is well known that, if a bipartite graph $G = (V, E)$, with $V = X \cup Y$ and all the edges of E joining a vertex of X to a vertex of Y, has no perfect matching, then there must exist a *deficient* subset of X (and of Y), i.e., a set of say r vertices in X which are collectively adjacent to fewer than r vertices in Y. To be more precise, the *deficiency* of a subset Z of X is defined by $\delta(Z) = |Z| - |\mathcal{N}(Z)|$, where $\mathcal{N}(Z)$, the *neighbourhood* of

Z, is the subset of Y consisting of vertices that are adjacent to at least one vertex in Z. The deficiency $\delta(G)$ of G is the maximum deficiency taken over all subsets of X. The following is a classical theorem:

Theorem 4.1. *The maximum size of matching in a bipartite graph G is given by $|X| - \delta(G)$.*

A *maximally deficient* subset of X is a subset Z such that $\delta(Z) = \delta(G)$, and we shall call a subset of X *critical* if it is maximally deficient *and* contains no maximally deficient proper subset. It is not hard to show that, if Z_1 and Z_2 are maximally deficient, then so also is $Z_1 \cap Z_2$, so that there is a unique critical subset of X.

For a fuller discussion of these issues, see, for example, [4].

Returning to our algorithm for a strongly stable matching, it turns out that if the bipartite graph that embodies the engagements at the end of a proposal sequence contains no perfect matching, then no woman who is engaged to a man in the critical set can have any of her fiancés, nor any man tied with them in her list, as a strongly stable partner. Hence all pairs of this kind may be deleted from the lists and the proposal sequence re-activated.

The whole process continues until some man's list becomes empty, in which case no strongly stable matching exists, or the bipartite graph of the engagement relation contains a perfect matching, in which case this matching is strongly stable. The algorithm appears as Algorithm STRONG in Fig. 4.

Correctness of Algorithm STRONG. We now set about establishing the correctness of Algorithm STRONG. The first part of the argument is similar to that employed in Section 3 for Algorithm SUPER.

Lemma 4.2. *If the pair (m, w) is deleted during an execution of Algorithm STRONG, then that pair cannot block any matching consisting of pairs that are never deleted.*

Proof. This is an immediate consequence of the fact that, in w's list, m is a strict successor of any undeleted entries. □

Lemma 4.3. *A matching generated by Algorithm STRONG is strongly stable.*

Proof. Suppose that some execution of Algorithm STRONG generates matching M, and suppose that matching M is blocked by the pair (m, w). By Lemma 4.2, the pair (m, w) cannot have been deleted. So, if (m, w) blocks M, w must be at the head of m's list. So the only possibility is that w is tied with m's partner in M and w strictly prefers m to her partner m' in M. But m must have proposed to w, so that the pair (m', w) must have been deleted, giving a contradiction. □

```
assign each person to be free;
repeat
  while some man m is free do
    for each woman w at the head of m's list do
    begin
      m proposes, and becomes engaged, to w;
      for each strict successor m' of m on w's list do
      begin
        if m' is engaged to w then
          break the engagement;
        delete the pair (m', w)
      end
    end;
  if the engagement relation does not contain a perfect matching then
  begin
    find the critical set Z of men;
    for each woman w who is engaged to a man in Z do
    begin
      break all engagements involving w;
      for each man m at the tail of w's list do
        delete the pair (m, w)
    end
  end
until (some man's list is empty) or (everyone is engaged);
if everyone is engaged then
  any perfect matching in the engagement relation is strongly stable
else
  no strongly stable matching exists
```

Fig. 4. Algorithm STRONG.

To complete the proof of the correctness of the algorithm, it will suffice to show that no strongly stable pair is ever deleted during its execution. For then the algorithm can never fail to find a strongly stable matching if one exists.

Lemma 4.4. *No strongly stable pair is ever deleted during an execution of Algorithm STRONG.*

Proof. Suppose, for a contradiction, that (m, w) is the first strongly stable pair deleted during some execution of the algorithm. Let M be a strongly stable matching in which m and w are partners.

Case 1. Suppose that (m, w) is deleted as a result of some other man, say m', becoming engaged to w. Then w strictly prefers m' to m. Also, there is no strongly stable matching in which m' has a partner, say w', whom he strictly prefers to w; for then the strongly stable pair (m', w') would have been deleted before (m, w). So the supposed strongly stable matching M is blocked by the pair (m', w), giving a contradiction.

Case 2. Suppose that (m, w) is deleted because w is engaged to a man in the critical set Z at some point, and at that point m is at the tail of w's list. We refer to the set of

lists at that point as the current lists. Let $U = \mathcal{N}(Z)$, let Z' be the set of men in Z who are matched in M with a woman from the head of their current list, and let U' be the set of women in U who are matched in M with a man from the tail of their current list. Then $w \in U'$, so that U' is nonempty. Further, any man m' in Z who is engaged to w must be in Z', otherwise (m', w) would block M, so that Z' is nonempty. Also, $Z \backslash Z'$ is nonempty, because $|Z| - |U| = \delta(G) > 0$, where G is the current engagement graph, while $|Z'| - |U| \leq |Z'| - |U'| \leq 0$, since every man in Z' is matched in M with a woman in U'.

Now, there must be an edge (x, y) in the engagement graph with $x \in Z \backslash Z'$ and $y \in U'$. For otherwise $\mathcal{N}(Z \backslash Z') \subseteq U \backslash U'$, and $|Z \backslash Z'| - |\mathcal{N}(Z \backslash Z')| \geq |Z \backslash Z'| - |U \backslash U'|$ $= |Z| - |U| - (|Z'| - |U'|) \geq |Z| - |U|$, since, as observed above, $|Z'| - |U'| \leq 0$; because $Z \backslash Z' \subset Z$, this contradicts the fact that Z is critical. This means that x strictly prefers y to his partner in M, and y is indifferent between x and her partner in M. So the supposed strongly stable matching M is blocked by the pair (x, y), giving a contradiction. □

If Algorithm STRONG finds a strongly stable matching, then because of Lemma 4.4, no man can have a strictly better partner in any other strongly stable matching. So, if strongly stable matchings exist for a given problem instance, the algorithm always finds one that is man-optimal in this sense. Note that it may not be unique, since there may be alternative perfect matchings in the engagement graph. Further, it is easy to see that man-optimal is woman-pessimal, in the sense that no woman can have a worse partner in any strongly stable matching (though again there may be other strongly stable matchings in which she has a different partner of equal ranking).

We summarize our findings as a theorem.

Theorem 4.5. *For every instance of stable marriage with ties, Algorithm STRONG determines whether or not a strongly stable matching exists. If such a matching does exist, all possible executions of the algorithm find one in which every man has as good a partner, and every woman as bad a partner, as in any strongly stable matching.*

Implementation and analysis of Algorithm STRONG. During each iteration of Algorithm STRONG, we need to search for a perfect matching in the bipartite graph formed by the engagement relation. Such a search may be carried out by a standard maximum cardinality matching algorithm.

The key to the analysis of Algorithm STRONG is bounding the total amount of work done in finding maximum cardinality matchings in the engagement graph. Once such a matching is known, the critical set can easily be obtained, using, say, breadth-first search based on the following lemma, in which we continue to assume that our bipartite graph is based on vertex sets X of men and Y of women.

Lemma 4.6. *Given a maximum cardinality matching M in a bipartite graph G, the critical set C consists of the set U of unmatched men together with the set R of men reachable from them via an alternating path.*

Proof. Let $Z = U \cup R$. It is immediate that $\delta(Z) = \delta(G)\,(= |U|)$, for if $\mathcal{N}(Z)$ were to contain a woman other than the mates in M of the men in R, then there would be an augmenting path relative to M, contradicting the maximality of M.

Further, the critical set C must contain every man that is unmatched relative to some maximum cardinality matching in G. For if M' is an arbitrary such matching (of size $|X| - \delta(G)$), and if $x \in X$ is not matched in M', then C must contain at least $|C| - \delta(G) + 1$ matched men, with the implication that $|\mathcal{N}(C)| \geq |C| - \delta(G) + 1$, or $|C| - |\mathcal{N}(C)| \leq \delta(G) - 1$, contradicting the required deficiency of C.

But, for every $x \in R$, there is a maximum cardinality matching in which x is unmatched, obtainable from M via an alternating path from a man in U to x. Hence, $Z \subseteq C$, and since $\delta(Z) = \delta(C)$, the proof of the lemma is complete. □

As in the previous algorithms, work done other than in finding maximum matchings and critical sets is essentially bounded by a constant times the number of deleted pairs, and so is $O(n^2)$.

Suppose that during the ith iteration of the repeat loop, x_i pairs are deleted because of the absence of a perfect matching in the engagement relation. These x_i pairs must include all the engaged pairs involving a man in the critical set C_i. Assuming that Algorithm STRONG finds a maximum cardinality matching M_i at the ith iteration, then the critical set C_i certainly includes all the men who are unmatched in M_i. Hence the number of pairs of M_i that are not deleted is at least $n - x_i$, if that number is positive.

Suppose further that in the $(i + 1)$th iteration, y_i of these matched pairs are deleted during the proposal sequence. Then at least $n - x_i - y_i$ pairs of M_i survive, so that in the $(i + 1)$th iteration we can start from these pairs and find a maximum cardinality matching in $O(\min(n^2(x_i + y_i), n^3))$ time, by the augmenting path method (see, for example, [6, pp. 221–224]).

Suppose that a total of m iterations are carried out, and that in k of these, $\min(n^2(x_i + y_i), n^3) = n^3$. Then the algorithm has time complexity $O(n^3 + kn^3 + n^2\sum(x_i + y_i))$, where the first term is for the first iteration, and the sum is over the appropriate $m - k - 1$ values of i. But $\sum_{i=1}^{m}(x_i + y_i) < n^2$, since the total number of pairs is n^2, and because $x_i + y_i > n$ for the appropriate k values of i, it follows that

$$kn + \sum(x_i + y_i) \leq \sum_{i=1}^{m}(x_i + y_i) < n^2$$

or

$$\sum(x_i + y_i) < n^2 - kn$$

where the sum is over the same $m - k - 1$ values of i.

Hence the overall complexity of Algorithm STRONG is $O((k + 1)n^3 + n^2(n^2 - kn)) = O(n^4)$.

5. Conclusion and open problems

We have shown how the classical Gale/Shapley stable marriage algorithm can be extended to deal with cases in which the preference lists include ties and stability is

defined in one of three possible ways. In the cases of weak stability and super-stability, we described asymptotically optimal $O(n^2)$ algorithms that will find a matching of the appropriate kind (or will show that none exists, in the case of super-stability). We also presented an algorithm that will find a strongly stable matching, if one exists. This algorithm involves a rather more subtle extension of the basic Gale/Shapley method, but can be implemented to run in $O(n^4)$ time using bipartite matching. It may be, however, that a more efficient implementation, or a more rigorous analysis, of Algorithm STRONG would lead to a tighter complexity bound.

As mentioned in the Introduction, the algorithms that we have described can easily be extended to the more general problem in which each person's preferences are expressed as a partial order. This merely involves interpreting the "head" of each person's (current) poset as the set of source nodes, and the "tail" as the set of sink nodes, in the corresponding directed acyclic graph.

Finally, we pose the question of whether efficient algorithms can be found for a super-stable or strongly stable roommates matching, or whether, as with weak stability [7], these problems are NP-complete.

References

[1] D. Gale and L.S. Shapley, College admissions and the stability of marriage, Amer. Math. Monthly 69 (1962) 9–15.

[2] D. Gusfield and R.W. Irving, The Stable Marriage Problem: Structure and Algorithms (MIT Press, Boston, MA, 1989).

[3] R.W. Irving, An efficient algorithm for the stable room-mates problem, J. Algorithms 6 (1985) 577–595.

[4] C.L. Liu, Introduction to Combinatorial Mathematics (McGraw-Hill, New York, 1968).

[5] C. Ng and D.S. Hirschberg, Lower bounds for the stable marriage problem and its variants, SIAM J. Comput. 19 (1990) 71–77.

[6] C.H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity (Prentice-Hall, Englewood Cliffs, NJ, 1982).

[7] E. Ronn, NP-complete stable matching problems, J. Algorithms 11 (1990) 285–304.

[8] A.E. Roth and M. Sotomayor, Two Sided Matching: A Study in Game-Theoretic Modelling and Analysis (Cambridge University Press, Cambridge, 1990).