The Stable Roommates Problem with Ties*

Robert W. Irving and David F. Manlove

Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK.
Email: {rwi,davidm}@dcs.gla.ac.uk.

Abstract

We study the variant of the well-known Stable Roommates problem in which participants are permitted to express ties in their preference lists. In this setting, more than one definition of stability is possible. Here we consider two of these stability criteria, so-called *super-stability* and *weak stability*. We present a linear-time algorithm for finding a super-stable matching if one exists, given a Stable Roommates instance with ties. This contrasts with the known NP-hardness of the analogous problem under weak stability. We also extend our algorithm to cope with preference lists that are incomplete and/or partially ordered. On the other hand, for a given Stable Roommates instance with ties and incomplete lists, we show that the weakly stable matchings may be of different sizes, and the problem of finding a maximum cardinality weakly stable matching is NP-hard, though approximable within a factor of 2.

Keywords: stable matching problem; indifference; super-stability; weak stability; linear-time algorithm; NP-completeness; approximation algorithm

1 Introduction

In a given instance of the Stable Roommates problem (SR), each of $2n$ participants ranks the others in strict order of preference. A *matching* is a set of n disjoint (unordered) pairs of participants. A matching M in an instance of SR is *stable* if there are no two participants x and y, each of whom prefers the other to his partner in M. Such a pair is said to *block* M, or to be a *blocking pair* with respect to M. Gale and Shapley [2] were the first to study SR, and gave an example instance which does not admit a stable matching. Irving [7] solved a problem posed by Knuth [12] when he described an $O(n^2)$ algorithm (henceforth Algorithm SR) – linear in the input size – to find a stable matching if one exists, for a given instance of SR. Subsequently, Gusfield [4] and Irving [8] explored structural aspects of SR, and a comprehensive discussion of the problem is given by Gusfield and Irving [5]. In addition, Tan [18] introduced the concept of a *stable partition* in an instance I of SR, which provides a 'succinct certificate' in the case that I does not admit a stable matching. More recently, Feder et al. [1] presented an $O(n \log^3 n)$ parallel algorithm for finding a stable matching if one exists, for a given instance of SR.

Note that SR is a non-bipartite extension of the classical Stable Marriage problem (SM) [5]. In an instance of SM, the set of participants is partitioned into two disjoint sets, the *men* and *women*, and each person ranks all members of the opposite sex in strict order of preference. An analogous stability definition holds in this context. It is known that every instance of SM admits at least one stable matching, and such a matching may be found in linear time using the Gale/Shapley algorithm [2].

*This work was supported by Engineering and Physical Sciences Research Council grant number GR/M13329.

1.1 Incomplete preference lists

SR may be generalised by allowing the preference lists of those involved to be incomplete (we refer to this problem as SRI). In this case, we say that participant p is *acceptable* to participant q if p appears on the preference list of q, and *unacceptable* otherwise. A matching M in an instance of SRI must satisfy the property that $\{x, y\} \in M$ implies that x, y find each other acceptable. The revised notion of stability may be defined as follows: given an instance of SRI, a matching M is stable if there are no two participants x and y, each of whom either is unmatched in M and finds the other acceptable, or prefers the other to his partner in M. (It follows that, from the point of view of finding stable matchings, we may assume without loss of generality that p is acceptable to q if and only if q is acceptable to p.) A stable matching for an instance of SRI need not be a complete matching. However, all stable matchings for a given instance have the same size and match exactly the same set of participants [5, Section 4.5.2] (assuming that one exists). It is not difficult to extend Algorithm SR to cope with SRI [5, Section 4.5.2].

1.2 Ties in the preference lists

Another natural generalisation of SR arises when participants are permitted to express ties in their preference lists. We refer to this problem as SRT (Stable Roommates with Ties). In the context of SM where ties are allowed (henceforth SMT), Irving [9] has defined three notions of stability, and these extend easily to SRT. Under the weakest of these three, a matching M is *weakly stable* if there are no two participants x and y, each of whom strictly prefers the other to his partner in M [1]. The weak stability definition is equivalent to a condition based on tie-breaking, as is indicated by the following proposition, whose proof is straightforward and is omitted.

Proposition 1.1. *Let M be a matching in an instance I of SRT. Then M is weakly stable in I if and only if M is stable in some instance of SR obtained from I by breaking the ties.*

In general, different ways of breaking the ties may yield some SR instances that admit stable matchings and some that do not, and there may be exponentially many ways of breaking the ties. Ronn [16] has shown that the problem of deciding whether an instance of SRT admits a weakly stable matching is NP-complete. The proof incorporates an ingenious but lengthy transformation from 3SAT; in this paper we formulate an alternative, shorter reduction starting from a graph matching problem.

According to the strongest of the above-mentioned stability criteria, a matching M is *super-stable* if there are no two participants x and y, each of whom either strictly prefers the other to his partner in M or is indifferent between them. Clearly a super-stable matching is weakly stable. In addition, the super-stability criterion gives rise to the following analogue of Proposition 1.1 (again the proof is straightforward and is omitted).

Proposition 1.2. *Let M be a matching in an instance I of SRT. Then M is super-stable in I if and only if M is stable in every instance of SR obtained from I by breaking the ties.*

A super-stable matching need not exist, even in an instance of SMT (see [9] for further details). However, Irving [9] has formulated a linear-time algorithm (henceforth Algorithm SMT-Super) for finding a super-stable matching if one exists, given an instance of SMT. Recently, Irving et al. [10] have extended this algorithm to cope with a many-one stable matching problem (the so-called Hospitals/Residents problem with Ties). The existence of

[1] Implicitly here, and henceforth for other stability definitions, such a pair $\{x, y\}$ is defined to *block* M, or to be a *blocking pair* with respect to M, as for the SR case.

an efficient algorithm for SRT under super-stability has remained open until now. In this paper we present a linear-time algorithm, Algorithm SRT-Super, which will determine, given an instance of SRT, whether a super-stable matching exists, and if so finds such a matching. Algorithm SRT-Super subsumes both Algorithms SR and SMT-Super, firstly since, for an SRT instance without ties Algorithm SRT-Super reduces to Algorithm SR, and secondly because SMT under super-stability is a special case of SRT under super-stability[2].

1.3 Extensions of SRT

When both generalisations of SR are in place simultaneously, we obtain the Stable Roommates problem with Ties and Incomplete lists (henceforth SRTI). The aforementioned stability definitions extend naturally to SRTI.

Firstly, a matching M in an instance of SRTI is weakly stable if there are no two participants x and y, each of whom either is unmatched and finds the other acceptable, or strictly prefers the other to his partner in M. Secondly, a matching M in an instance of SRTI is super-stable if there are no two participants x and y, each of whom either is unmatched and finds the other acceptable, or strictly prefers the other to his partner in M or is indifferent between them.

It is possible to generalise the concept of ties in a participant p's preference list to the situation in which p's list is an arbitrary partial order. In this case we obtain an instance of Stable Roommates with arbitrary Partial orders (SRP). Note that p *strictly prefers* a participant q to another participant r if q precedes r in the partial order representing p's list; p is *indifferent* between q and r if q, r are incomparable in this partial order. Clearly the definitions of weak stability and super-stability defined above for SRT extend to SRP. In addition, those stability definitions for SRTI also carry over to SRPI (i.e. SRP in which each participant may declare one or more participants as being unacceptable).

For a given instance of SRTI, SRP or SRPI, the analogues of Propositions 1.1 and 1.2 hold in each case (in the cases of SRP and SRPI, the phrase 'breaking the ties' should be replaced by 'forming a linear extension of each partial order').

We shall indicate how to extend Algorithm SRT-Super to handle an instance of SRTI, SRP or SRPI. We also prove that, if I is an instance of SRTI or SRPI, and M, M' are two super-stable matchings in I, then a participant p is matched in M if and only if p is matched in M'; hence all super-stable matchings in I have the same cardinality.

By contrast, we show that, for a given instance of SRTI, the weakly stable matchings may be of different sizes, and the problem of finding a maximum cardinality weakly stable matching is NP-hard, though approximable within a factor of 2.

1.4 An alternative viewpoint

Among the possible stability criteria that may be applied to stable matching problems involving ties, it is weak stability that has been more commonly studied in the literature [17, 15, 16, 11]. However, an example context in which super-stability is relevant is when there is uncertainty in the preference lists. Suppose that, in a stable roommates instance, we wish to find a stable matching (in the classical sense), but for some or all of the participants we have only partial information regarding preferences. In general, each preference 'list' may be expressable only as a partial order, and the particular linear extension that represents a participant's true preferences is unknown. Therefore in view of the extension of Proposition 1.2 to SRPI, a super-stable matching is one that is stable

[2]Gusfield and Irving [5, Lemma 4.1.1] show that SM is a special case of SR; it is straightforward to adapt their proof in order to show that SMT under super-stability is a special case of SRT under super-stability.

no matter which linear extensions of the various preference posets represent the true preferences.

1.5 Organisation of the paper

The remainder of this paper is organised as follows. Sections 2 and 3 describe Phases 1 and 2 respectively of Algorithm SRT-Super. An analysis of Algorithm SRT-Super is presented in Section 4, whilst Section 5 discusses extensions of the algorithm. The NP-hardness and approximability results concerning weak stability appear in Section 6. Finally Section 7 contains some conclusions and open problems.

2 Phase 1 of Algorithm SRT-Super

As is the case for Algorithm SR [7], Algorithm SRT-Super is in two phases. In Phase 1, entries are deleted from preference lists, and such deletions continue until either

1. one or more of the lists become empty, in which case we can conclude immediately that no super-stable matching exists, or
2. every list contains exactly one entry, in which case these lists constitute a super-stable matching, or
3. the lists reach a certain stable state, but with no list empty and some lists having > 1 entry, in which case we embark on Phase 2 of the algorithm.

Throughout, the lists remain *consistent*, in the sense that, whenever x is deleted from the list of y, y will also be deleted from the list of x; when this happens, the pair $\{x, y\}$ is said to be deleted.

Phase 1 of Algorithm SRT-Super is described in pseudocode in Figure 2. This part of the algorithm can be viewed as a proposal sequence, familiar in the context of stable matching problems, and originating in a simpler form in the seminal paper of Gale and Shapley [2]. Each participant proposes to, and becomes provisionally assigned to, the set of one or more others at the head of his current list. (Note that this assignment relation is typically not symmetric – usually p is assigned to q does not imply that q is assigned to p.) Whenever a participant receives a proposal, all strict successors of the proposer are deleted from his list, and if a participant receives a proposal from two (or more) whom he ranks equally, then they and all others ranked with them in his list are deleted. Deletions, of course, break any provisional assignments among the deleted pairs. Note that the algorithm halts immediately if any participant's list becomes empty during Phase 1. We shall prove that, should this occur, it follows that there cannot exist a super-stable matching. Before doing so, we show that any pair of participants deleted during Phase 1 cannot be partners in any super-stable matching.

Lemma 2.1. *If the pair $\{q, r\}$ is deleted during Phase 1 of the algorithm then there is no super-stable matching in which q and r are partners.*

Proof. We assume, for a contradiction, that the lemma is false, and that $\{q, r\}$ is the first super-stable pair – i.e., pair that belongs to some super-stable matching – to be deleted. There are two places in the algorithm where pairs are deleted, indicated by (1) and (2). We consider these separately.
(1) Suppose, without loss of generality, that $\{q, r\}$ is deleted at this point because r is a strict successor of p in q's list and p has become assigned to q. Suppose that M is a super-stable matching in which q and r are partners. Then q strictly prefers p to r, and because

```
while some participant p has a non-empty list
        and p is not assigned to anyone loop
    for each q at the head of p's list loop
        assign p to q;                                  -- p 'proposes' to q
        for each strict successor r of p in q's list loop
            if r is assigned to q then
                break the assignment;
            end if;
            delete the pair {q, r};                     -- (1)
            if r's list is empty then
                no super-stable matching exists;        -- halt here
            end if;
        end loop;
        if ≥ 2 participants are assigned to q then
            break all assignments to q;
            for each r tied with p in q's list loop;    -- including p
                delete the pair {q, r};                 -- (2)
                if r's list is empty then
                    no super-stable matching exists;    -- halt here
                end if;
            end loop;
            if q's list is empty then
                no super-stable matching exists;        -- halt here
            end if;
        end if;
    end loop;
end loop;
```

Figure 1: Phase 1 of Algorithm SRT-Super

no super-stable pair was previously deleted, p must either strictly prefer q to his partner in M or be indifferent between them. Hence the pair $\{p, q\}$ blocks M, a contradiction.
(2) Suppose, again without loss of generality, that $\{q, r\}$ is deleted at this step because r is tied with p in q's list and, in addition to p, another participant s was assigned to q (note that possibly $p = r$). Again suppose that M is a super-stable matching in which q and r are partners. Then q is indifferent between s and r, and because no super-stable pair was previously deleted, s must either strictly prefer q to his partner in M or be indifferent between them. Hence the pair $\{q, s\}$ blocks M, a contradiction. □

Corollary 2.2. *If some participant's list becomes empty during Phase 1 of the algorithm then there is no super-stable matching.*

Proof. In a super-stable matching, every participant must have a partner, but in view of Lemma 2.1, there is no partner available for a participant whose list becomes empty. □

Next we prove that, under certain conditions, Phase 1 of the algorithm might terminate with a super-stable matching, without the need for Phase 2.

Lemma 2.3. *If the pair $\{q, r\}$ is deleted during an execution of Phase 1 of the algorithm, then either q and all of its equals and successors were deleted from r's list, or r and all of its equals and successors were deleted from q's list.*

Proof. This result follows immediately by inspection of the points (1) and (2) at which deletions occur during Phase 1. □

Corollary 2.4. *If Phase 1 of the algorithm ends with every participant's list having exactly one entry then the matching represented by these lists is super-stable.*

Proof. It is immediate that the lists are consistent, and therefore that they do represent a matching, say M. Suppose that the pair $\{q, r\}$ blocks M, and let the partner in M of a participant s be denoted by $p_M(s)$. By Lemma 2.3, either q strictly prefers $p_M(q)$ to r, or r strictly prefers $p_M(r)$ to q. Hence $\{q, r\}$ does not block M after all. □

We refer to the preference lists that remain after Phase 1 of the algorithm as the *reduced lists*. It therefore remains to decide how to proceed if none of the reduced lists is empty but one or more (indeed necessarily two or more) contain more than one entry. As will become apparent in Section 3 below, a central feature of Phase 2 of the algorithm is the reactivation of Phase 1, but on a sequence of nested *super-stable tables* – such a structure is a generalisation of the *stable table* as defined in [5, page 169]. An appropriate sequence of such eliminations will lead to a super-stable matching, if one exists, or otherwise to the conclusion that no super-stable matching is possible.

3 Phase 2 of Algorithm SRT-Super

We begin by defining the term *super-stable table*.

Definition 3.1. *A* super-stable table $\mathcal{T}$ *for an instance I of SRT is a set of preference lists, derived from the original preference lists in I by the deletion of zero or more pairs, such that*

1. *for each participant x, the first entry $f_{\mathcal{T}}(x)$ in x's list is not tied with any other entry in x's list in $\mathcal{T}$;*
2. *$y = f_{\mathcal{T}}(x) \Rightarrow x \in l_{\mathcal{T}}(y)$, where $l_{\mathcal{T}}(y)$ is the tie of one or more entries in last place in y's list in $\mathcal{T}$;*
3. *$x \neq x' \Rightarrow f_{\mathcal{T}}(x) \neq f_{\mathcal{T}}(x')$;*
4. *the pair $\{u, v\}$ is a deleted pair if and only if u strictly prefers each member of $l_{\mathcal{T}}(u)$ to v or v strictly prefers each member of $l_{\mathcal{T}}(v)$ to u (or both)*[3]*;*
5. *no list in $\mathcal{T}$ is empty.*

We firstly establish that, under certain conditions, the reduced lists output by Phase 1 of the algorithm are a super-stable table.

Lemma 3.2. *Let $\mathcal{T}$ be the set of reduced lists produced by Phase 1 of Algorithm SRT-Super. Then*
(i) each list in $\mathcal{T}$ has just one participant in first place, and no participant is first in more than one list;
(ii) if $y = f_{\mathcal{T}}(x)$ then $x \in l_{\mathcal{T}}(y)$.

Proof. (i) If some list has ≥ 2 participants at the head then some participant would have ≥ 2 others assigned to him, and these assignments would have been broken by Phase 1 of the algorithm. The same would be true if some participant were first in two different lists. (ii) This is immediate from the algorithm, since all strict successors of x in y's list will have been deleted. □

[3] Henceforth we use the abbreviation 'v strictly prefers $l_{\mathcal{T}}(v)$ to u' etc.

Corollary 3.3. *If Phase 1 of the algorithm does not terminate with an empty list, then the reduced lists that it produces are a super-stable table.*

Proof. Lemma 3.2 establishes Properties 1,2 and 3 of Definition 3.1, and Lemma 2.3 establishes Property 4 of Definition 3.1. □

We now present some notation and terminology regarding preference lists. For a given set of consistent preference lists $\mathcal{L}$, we let $\{x, y\} \in \mathcal{L}$ denote the fact that x appears on y's list in $\mathcal{L}$ (and vice-versa, since $\mathcal{L}$ is consistent). We say that a matching M is *embedded* in a set of preference lists $\mathcal{L}$ if $\{x, y\} \in M$ implies that $\{x, y\} \in \mathcal{L}$. Our aim is to demonstrate how an appropriate sequence of reactivations of Phase 1 allows us to extract an embedded super-stable matching, if one exists, from a super-stable table. Before doing so, we introduce some further notation concerning super-stable tables.

Let $\mathcal{T}$ be a super-stable table in which some participant x's list has length > 1, and let $z = f_{\mathcal{T}}(x)$. Define $\mathcal{L}_{\mathcal{T},x}$ to be the set of preference lists obtained from $\mathcal{T}$ by deleting the pairs $\{z, w\}$, for each $w \in l_{\mathcal{T}}(z)$ (this includes the pair $\{x, z\}$). We denote by $\mathcal{T}_x$ the preference lists obtained by applying Phase 1 of Algorithm SRT-Super to $\mathcal{L}_{\mathcal{T},x}$. If this application of Phase 1 results in a list becoming empty, we define $\mathcal{T}_x$ to be the null table, represented by $\langle\rangle$. Otherwise, $\mathcal{T}_x$ must be a super-stable table, as we now demonstrate.

Lemma 3.4. *Let $\mathcal{T}$ be a super-stable table in which some participant x's list has length > 1. Then either $\mathcal{T}_x = \langle\rangle$ or $\mathcal{T}_x$ is a super-stable table.*

Proof. Suppose that the application of Phase 1 to $\mathcal{L}_{\mathcal{T},x}$ does not result in any person's preference list becoming empty. Then Property 5 of Definition 3.1 holds for $\mathcal{T}_x$. The deletions carried out in order to obtain $\mathcal{L}_{\mathcal{T},x}$ from $\mathcal{T}$ satisfy Property 4 of Definition 3.1, as do any further deletions carried out by the application of Phase 1 to $\mathcal{L}_{\mathcal{T},x}$, by an argument similar to Lemma 2.3. Properties 1, 2 and 3 of Definition 3.1 may be established by an argument similar to Lemma 3.2. Hence $\mathcal{T}_x$ is a super-stable table. □

We remark that, as in the no-ties case, any two applications of Phase 1 will generate the same super-stable table – the proof is similar to that of Lemma 4.2.1 in [5]. The following result indicates that the application of Phase 1 to $\mathcal{L}_{\mathcal{T},x}$ never deletes a pair that might belong to some super-stable matching embedded in $\mathcal{L}_{\mathcal{T},x}$.

Lemma 3.5. *Let $\mathcal{T}$ be a super-stable table in which some participant x's list has length > 1. If the pair $\{q, r\}$ is deleted during the application of Phase 1 to $\mathcal{L}_{\mathcal{T},x}$, then there is no super-stable matching embedded in $\mathcal{L}_{\mathcal{T},x}$ in which q and r are partners.*

Proof. The proof is virtually identical to that of Lemma 2.1, with minor modifications. □

The fundamental result that underlies Phase 2 of our algorithm is the following.

Lemma 3.6. *Let $\mathcal{T}$ be a super-stable table that contains an embedded super-stable matching. Suppose that some participant x's list in $\mathcal{T}$ has length > 1. Let $y \in l_{\mathcal{T}}(x)$ be such that $f_{\mathcal{T}}(y) = x$. Then either $\mathcal{T}_x$ or $\mathcal{T}_y$ (or both) contains an embedded super-stable matching.*

Proof. Let M be a super-stable matching embedded in $\mathcal{T}$, and let $z = f_{\mathcal{T}}(x)$. We firstly note that $z \neq y$, for otherwise $z \in l_{\mathcal{T}}(x)$, and since x's list has length > 1, this contradicts Property 1 of Definition 3.1.

Suppose that $\{x, z\} \notin M$. Then $\{w, z\} \notin M$, for any $w \in l_{\mathcal{T}}(z)\backslash\{x\}$, for otherwise $\{x, z\}$ blocks M. Hence M is embedded in $\mathcal{L}_{\mathcal{T},x}$. Then by Lemma 3.5, M is embedded in $\mathcal{T}_x$.

On the other hand suppose that $\{x, z\} \in M$; then $\{y, x\} \notin M$. It follows that $\{w, x\} \notin M$, for any $w \in l_{\mathcal{T}}(x)\backslash\{y\}$, for otherwise $\{x, y\}$ blocks M. Hence M is embedded in $\mathcal{L}_{\mathcal{T},y}$. Then by Lemma 3.5, M is embedded in $\mathcal{T}_y$. □

We now complement Lemma 3.6 by proving that, if $\mathcal{T}_x \neq \langle\rangle$ and $\mathcal{T}$ contains an embedded super-stable matching, then $\mathcal{T}_x$ does also.

Lemma 3.7. *Let $\mathcal{T}$ be a super-stable table that contains an embedded super-stable matching. Suppose that some participant x's list in $\mathcal{T}$ has length > 1. Suppose further that $\mathcal{T}_x \neq \langle\rangle$. Then $\mathcal{T}_x$ contains an embedded super-stable matching.*

Proof. By Lemma 3.4, $\mathcal{T}_x$ is a super-stable table. Let M be a super-stable matching embedded in $\mathcal{T}$. We construct a matching N embedded in $\mathcal{T}_x$ as follows. For each participant p in turn who is as yet unmatched in N, suppose that $\{p, q\} \in M$. We consider two cases:

- Case (i): $\{p, q\} \in \mathcal{T}_x$. Place $\{p, q\}$ in N.
- Case (ii): $\{p, q\} \notin \mathcal{T}_x$. Since $\mathcal{T}_x$ is a super-stable table, it follows that either p strictly prefers $l_{\mathcal{T}_x}(p)$ to q or q strictly prefers $l_{\mathcal{T}_x}(q)$ to p. Suppose that p strictly prefers $l_{\mathcal{T}_x}(p)$ to q. There exists a unique $z \in l_{\mathcal{T}_x}(p)$ such that $p = f_{\mathcal{T}_x}(z)$. Place $\{p, z\}$ in N.

We show that N is a super-stable matching.

If p is matched to z in N using Case (ii) above, then z cannot be matched to some other participant in N. For, p strictly prefers z to q (where $\{p, q\} \in M$), so that z strictly prefers $p_M(z)$ to p, for otherwise $\{p, z\}$ would block M. Now $f_{\mathcal{T}_x}(z) = p$, so that $\{z, p_M(z)\} \notin \mathcal{T}_x$. Thus z cannot be not matched in N using Case (i). Also, since $f_{\mathcal{T}_x}(z) = p$ and z strictly prefers $p_M(z)$ to p, it is impossible for z to be matched to anyone other than p using Case (ii). Finally, a simple counting argument establishes that everyone has a partner in N, so that N is a matching.

Now suppose that $\{r, s\}$ blocks N. If neither r nor s has a strictly poorer partner in N than in M, then $\{r, s\}$ also blocks M, a contradiction. Hence without loss of generality suppose that r strictly prefers $p_M(r)$ to $p_N(r)$. Then r must be matched in N using Case (ii), where r plays the rôle of z. But then $p_N(r) = f_{\mathcal{T}_x}(r)$, so that $\{r, s\} \notin \mathcal{T}_x$. Hence as $\mathcal{T}_x$ is a super-stable table, s strictly prefers $l_{\mathcal{T}_x}(s)$ to r, a contradiction. Hence N is a super-stable matching. □

Lemmas 3.6 and 3.7 suggest a strategy that forms the basis of Phase 2 of Algorithm SRT-Super. Starting with a super-stable table $\mathcal{T}$ which has an embedded super-stable matching, we suppose that some participant x's list in $\mathcal{T}$ has length > 1. If $\mathcal{T}_x \neq \langle\rangle$, then by Lemma 3.7, $\mathcal{U} = \mathcal{T}_x$ contains an embedded super-stable matching. Otherwise, by Lemma 3.6, $\mathcal{U} = \mathcal{T}_y$ contains an embedded super-stable matching (where y is defined as in Lemma 3.6), so that $\mathcal{T}_y \neq \langle\rangle$. In addition, $\mathcal{U}$ is itself a super-stable table, by Lemma 3.4. But $\mathcal{U}$ has been obtained from $\mathcal{T}$ by deleting at least one pair. It follows that repeated application of this procedure will terminate with a super-stable table which has lists of length 1. The following lemma indicates that we have reached a solution in this case.

Lemma 3.8. *Let $\mathcal{T}$ be a super-stable table such that each list has length 1. Then $\mathcal{T}$ specifies a super-stable matching.*

Proof. Clearly $\mathcal{T}$ specifies a matching M, by Property 2 of Definition 3.1. The proof that M is super-stable is similar to that of Corollary 2.4; in this case the result follows by Properties 1 and 4 of Definition 3.1. □

Phase 2 of Algorithm SRT-Super is described in pseudocode in Figure 3. The following theorem brings together the various results proved so far concerning Phases 1 and 2 of Algorithm SRT-Super.

Theorem 3.9. *For a given instance of SRT, Algorithm SRT-Super determines whether a super-stable matching exists, and if so, the algorithm finds such a matching.*

```
T := super-stable table generated by Phase 1;
while some list in T has length > 1 loop
    x := some participant whose list in T has length > 1;
    let y ∈ l_T(x) be such that f_T(y) = x;
    calculate T_x; -- do not halt overall algorithm if a list becomes empty
    calculate T_y; -- do not halt overall algorithm if a list becomes empty
    if T_x ≠ ⟨⟩ then
        T := T_x;
    else if T_y ≠ ⟨⟩ then
        T := T_y;
    else
        no super-stable matching exists;        -- halt here
    end if;
end loop;
T specifies a super-stable matching;
```

Figure 2: Phase 2 of Algorithm SRT-Super

Proof. Let I be an instance of SRT and suppose that I admits a super-stable matching. If Phase 1 terminates with every participant's list having length 1, then these lists specify a super-stable matching by Corollary 2.4, and the algorithm terminates. Otherwise, we enter Phase 2, since no list becomes empty by Corollary 2.2. By Corollary 3.3, the reduced lists produced by Phase 1 constitute a super-stable table $\mathcal{T}$, and by Lemma 2.1 there is a super-stable matching embedded in $\mathcal{T}$. The main loop of Phase 2 now iterates until all lists in $\mathcal{T}$ have length 1, by Lemmas 3.6 and 3.7; in this case, $\mathcal{T}$ specifies a super-stable matching, by Lemma 3.8. □

4 Analysis of Algorithm SRT-Super

As is standard in stable matching algorithms, we assume that the input preference lists are pre-processed to produce a ranking array. This enables the position of a given participant in the preference list of another to be determined in constant time. It is clear that such a ranking array can be constructed in $O(n^2)$ time.

With the aid of a ranking array and a suitable representation of the preference lists, it is not hard to see that the number of operations required during Phase 1 of the algorithm is $O(n + d)$, where d is the number of pairs deleted.

As far as Phase 2 is concerned, if the algorithm is implemented exactly as Algorithm 2, then the complexity is no better than $O(n^4)$ in the worst case. This is because calculation of $\mathcal{T}_x$ and $\mathcal{T}_y$ may require $\Omega(n^2)$ operations, and even when this is the case in every loop iteration, the number of deletions per iteration may be $O(1)$, and the number of loop iterations may therefore be quadratic in n. To avoid this worst-case behaviour, we need a more subtle approach.

We initially make a copy $\mathcal{T}'$ of $\mathcal{T}$, which costs $O(n^2)$ time. We then choose x and y as before, and let $z = f_{\mathcal{T}}(x)$. We delete the pairs $\{z, w\}$, for each $w \in l_{\mathcal{T}}(z)$, from $\mathcal{T}$ and $\{x, v\}$, for each $v \in l_{\mathcal{T}'}(x)$, from $\mathcal{T}'$. We then simulate parallel applications of Phase 1 on $\mathcal{T}$ and $\mathcal{T}'$, calculating $\mathcal{T}_x$ and $\mathcal{T}'_y$ respectively, never allowing the number of deletions in one to exceed the number in the other by more than 1. In each case we retain a stack of deleted pairs for possible reinstatement. The parallel simulation is halted as soon as either one of the Phase 1 applications terminates.

Suppose that it is the application of Phase 1 on $\mathcal{T}$ that terminates first (a similar argument applies in the other case). If $\mathcal{T}_x = \langle\rangle$ then we restore $\mathcal{T}$ from the stack (at no

increase in asymptotic cost), and proceed with the application of Phase 1 to $\mathcal{T}'$. If it then transpires that $\mathcal{T}'_y = \langle\rangle$ then we exit Phase 2 with the conclusion that no super-stable matching exists, otherwise we duplicate $\mathcal{T}'_y$ by applying the same execution of Phase 1 to the restored $\mathcal{T}$. We now have two copies of the reduced super-stable table at a total cost that is $O(d)$, where d is the number of pairs deleted during this iteration of the main loop of Phase 2.

On the other hand, if $\mathcal{T}_x \neq \langle\rangle$ then we restore $\mathcal{T}'$ from the stack and apply the same sequence of operations to $\mathcal{T}'$ to produce, again, two copies of the reduced super-stable table in $O(d)$ time.

Repeating this process until one of the two Phase 2 terminating conditions arises gives an overall $O(n^2)$ algorithm for SRT under super-stability.

5 Extensions of Algorithm SRT-Super

In this section we describe, without proof, how to extend Algorithm SRT-Super to cope with an instance of SRTI or SRP. Both modifications may be incorporated simultaneously if we are given an instance of SRPI.

5.1 Incomplete lists

To handle an instance I of SRTI, Phase 1 of the algorithm should be altered as follows. Since one or more participants may be unmatched in a super-stable matching in I, Phase 1 should not terminate as soon as some participant's list becomes empty. However, once Phase 1 terminates, if any participant p ends up with an empty list, yet p received a proposal during execution of Phase 1, then it is not hard to see that no super-stable matching exists. Otherwise, if all lists have length 0 or 1, then the matching specified is a super-stable matching. If neither of these two conditions holds, then Phase 2 is necessary. We may as well discard all participants with empty lists at this point; then the definition of a super-stable table $\mathcal{T}$ remains valid, as does the definition of $\mathcal{T}_x$. Under this assumption, Phase 2 is unchanged and incorporates the original version of Phase 1 – every participant with a nonempty list after Phase 1 *must* be matched in a super-stable matching output by Phase 2, since each has received a proposal.

It may also be demonstrated that the set of participants who are matched and unmatched in a super-stable matching output by Algorithm SRT-Super is the same in any super-stable matching, as follows.

Theorem 5.1. *Let I be an instance of SRTI that admits a super-stable matching. Then the participants in I may be partitioned into two sets: those who are matched in all super-stable matchings and those who are matched in none. A participant p belongs to the former set if and only if p has a non-empty list after Phase 1 of Algorithm SRT-Super.*

Proof. Let M be a super-stable matching output by Algorithm SRT-Super. By the discussion in the first paragraph of this subsection, a participant p is matched in M if and only if p has a non-empty list after Phase 1. Now let M' be any super-stable matching in I. By the analogue of Proposition 1.2 for SRTI, M is stable in some instance I' of SRI obtained from I by breaking the ties, and in addition, M' is stable in I'. The result follows by applying Theorem 4.5.2 of [5] to I'. □

Note that the analogue of Theorem 5.1 also holds for SRPI under super-stability.

1 : 4 3
2 : 4
3 : 1
4 : (1 2)

Figure 3: An instance of SRTI with weakly stable matchings of sizes 1 and 2.

5.2 Partially ordered lists

We begin by generalising to SRP a number of definitions relevant in the context of SRT. Given an instance I of SRP, by the *head* of a participant p's preference poset we mean the set of people, each of whom has no predecessors in p's poset. Analogously, the *tail* of p's preference poset is the set of people, each of whom has no successors in p's poset. By 'each r tied with p in q's list' we now mean each r such that q is indifferent between p and r (including p). Given these definitions, the only other modification of Phase 1 is that the while loop should iterate so long as there is a participant p such that p's poset is non-empty and p is not assigned to *every* member of the head of his poset. As far as Phase 2 is concerned, an alteration of some notation is all that is required: in the definition of a super-stable table, $f_{\mathcal{T}}(x)$ denotes the sole participant who constitutes the head of x's poset, and $l_{\mathcal{T}}(x)$ denotes the tail of x's poset.

6 Weak stability

In this section we consider the weak stability criterion applied to instances of SRT and SRTI. We begin by demonstrating, in contrast with Theorem 5.1, that for a given instance of SRTI, the weakly stable matchings may be of different cardinalities. Consider the instance I of SRTI shown in Figure 6, involving 4 participants (in a given preference list, participants within round brackets are tied). Clearly the matchings $\{\{1,4\}\}$ and $\{\{1,3\},\{2,4\}\}$ are both weakly stable in I.

6.1 Hardness of finding a maximum weakly stable matching

Since weakly stable matchings may be of different sizes in a given instance of SRTI, the question arises as to whether there exist efficient algorithms for finding weakly stable matchings of maximum and minimum size. In fact, each of the problems of finding a weakly stable matching of maximum and minimum size in an instance of SMTI (Stable Marriage with Ties and Incomplete lists) is NP-hard [14], even if each tie is of length 2 and there is at most one tie per list in both cases. These results immediately carry over to SRTI[4]; in the case of the maximisation problem for SRTI, a shorter transformation exists, which we now give. We begin by defining the following related decision problem:

Name: COMPLETE WEAKLY STABLE MATCHING IN SRTI
Instance: Arbitrary instance I of SRTI
Question: Does I admit a complete weakly stable matching?

Our reduction establishing NP-completeness for COMPLETE WEAKLY STABLE MATCHING IN SRTI makes use of the following NP-complete decision problems:

[4]As previously mentioned, Gusfield and Irving [5, Lemma 4.1.1] show that SM is a special case of SR; it is straightforward to adapt their proof in order to show that SMTI under weak stability is a special case of SRTI under weak stability.

Name: MINIMUM (resp. EXACT) MAXIMAL MATCHING
Instance: Graph $G = (V, E)$ and integer $K \in \mathbb{Z}^+$
Question: Does G have a maximal matching M with $|M| \leq K$ (resp. $|M| = K$)?

Note that a matching M in a graph G is *maximal* if no proper superset of M is a matching in G. MINIMUM MAXIMAL MATCHING is NP-complete [19], even for cubic graphs [6] (a graph is *cubic* if each vertex has degree 3). Hence NP-completeness also holds for EXACT MAXIMAL MATCHING under the same restriction (this follows since, by considering augmenting paths, it may be verified that a graph contains maximal matchings of all sizes between the minimum and maximum).

Theorem 6.1. COMPLETE WEAKLY STABLE MATCHING IN SRTI *is NP-complete.*

Proof. Clearly COMPLETE WEAKLY STABLE MATCHING IN SRTI is in NP. To show NP-hardness, we transform from EXACT MAXIMAL MATCHING for cubic graphs, as discussed above. Let $G = (V, E)$ and $K \in \mathbb{Z}^+$ be an instance of this problem. Assume that $V = \{v_1, \ldots, v_n\}$ and, without loss of generality, assume that $K \leq \lfloor \frac{n}{2} \rfloor$.

We construct an instance I of SRTI as follows: let $U \cup X \cup Y \cup Z$ be the set of people, where

$$\begin{array}{rcll} U & = & \left(\bigcup_{i=1}^{i=n} U_i\right), & \\ U_i & = & U_i^1 \cup U_i^2 & (1 \leq i \leq n), \\ U_i^1 & = & \{u_{i,j} : 1 \leq j \leq 3\} & (1 \leq i \leq n), \\ U_i^2 & = & \{u_{i,j} : 4 \leq j \leq 3 + n - 2K\} & (1 \leq i \leq n), \\ X & = & \{x_j : 1 \leq j \leq n - 2K\}, & \\ Y & = & \bigcup_{i=1}^{i=n} Y_i, & \\ Y_i & = & \{y_{i,j} : 1 \leq j \leq 3 + n - 2K\} & (1 \leq i \leq n), \\ \text{and} \quad Z & = & \{z_i : 1 \leq i \leq n\}. & \end{array}$$

For any j $(4 \leq j \leq 3 + n - 2K)$, let $U_j^3 = \{u_{i,j} : 1 \leq i \leq n\}$. For any i $(1 \leq i \leq n)$, let $\{r_{i,1}, r_{i,2}, r_{i,3}\}$ be the set of three distinct integers such that $\{v_i, v_{r_{i,j}}\} \in E$ $(1 \leq j \leq 3)$. For each i $(1 \leq i \leq n)$ and j $(1 \leq j \leq 3)$, define an integer $s_{i,j}$ as follows: if $k = r_{i,j}$, then $v_i = v_{r_{k,l}}$ for some l $(1 \leq l \leq 3)$; let $s_{i,j} = l$. Finally, for each i $(1 \leq i \leq n)$, let $A_i = U_{r_{i,1}}^2 \cup U_{r_{i,2}}^2 \cup U_{r_{i,3}}^2$.

Create preference lists for each participant as follows:

$$\begin{array}{rcll} u_{i,j}: & y_{i,j} \;\; u_{r_{i,j},s_{i,j}} & (1 \leq i \leq n, \;\; 1 \leq j \leq 3) \\ u_{i,j}: & y_{i,j} \;\; [A_i] \;\; x_{j-3} & (1 \leq i \leq n, \;\; 4 \leq j \leq 3 + n - 2K) \\ x_j: & [U_{j+3}^3] & (1 \leq j \leq n - 2K) \\ y_{i,j}: & (u_{i,j} \;\; z_i) & (1 \leq i \leq n, \;\; 1 \leq j \leq 3 + n - 2K) \\ z_i: & [Y_i] & (1 \leq i \leq n) \end{array}$$

In a preference list, participants within square brackets are listed in arbitrary strict order. It is straightforward to verify that these preference lists are indeed consistent (in the sense that, for any participants p and q, p finds q acceptable if and only if q finds p acceptable). We claim that G has a maximal matching of size exactly K if and only if I admits a complete weakly stable matching.

For, suppose that G has a maximal matching M where $|M| = K$. We construct a matching M' in I as follows. Consider any edge $\{v_i, v_k\}$ in M. Then $v_i = v_{r_{k,l}}$ for some l $(1 \leq l \leq 3)$, and $v_k = v_{r_{i,j}}$ for some j $(1 \leq j \leq 3)$. Add the following pairs to M': $\{u_{i,j}, u_{k,l}\}$, $\{u_{i,r}, y_{i,r}\}$ $(1 \leq r \leq 3 + n - 2K,\ r \neq j)$, $\{u_{k,r}, y_{k,r}\}$ $(1 \leq r \leq 3 + n - 2K,\ r \neq l)$, $\{y_{i,j}, z_i\}$, and $\{y_{k,l}, z_k\}$. There remain $n - 2K$ sets U_{p_i} $(1 \leq i \leq n - 2K)$ such that every member of U_{p_i} is as yet unmatched. For each i $(1 \leq i \leq n - 2K)$, add the following

pairs to M': $\{u_{p_i,i+3}, x_i\}$, $\{u_{p_i,r}, y_{p_i,r}\}$ $(1 \leq r \leq 3+n-2K,\ r \neq i+3)$, and $\{y_{p_i,i+3}, z_{p_i}\}$. Clearly M' is a complete matching in I. It remains to show that M' is weakly stable.

It is straightforward to verify that no member of $X \cup Y \cup Z$ can be involved in a blocking pair of M'. Now suppose that some unmatched pair $\{u_{i,j}, u_{k,l}\}$ blocks M'. Then $i \neq k$, $j \geq 4$ and $l \geq 4$. Thus $\{u_{i,j}, x_{j-3}\} \in M'$ and $\{u_{k,l}, x_{l-3}\} \in M'$, so that no edge of M is incident in G to either v_i or v_k. Thus $M \cup \{\{v_i, v_k\}\}$ is a matching in G, contradicting the maximality of M.

Conversely suppose that M' is a complete weakly stable matching in I. For each i $(1 \leq i \leq n)$, z_i is matched in M' to $y_{i,j}$, for some j $(1 \leq j \leq 3+n-2K)$, and hence, for that j, $u_{i,j}$ is matched in M' to some participant in $U \cup X$. If $\{u_{i,j}, u_{k,l}\} \in M'$ for some k $(1 \leq k \leq n)$ and l $(1 \leq l \leq 3+n-2K)$, then we add the edge $\{v_i, v_k\}$ to M. Note that (i) $u_{i,r}$ is matched in M' to $y_{i,r}$ $(1 \leq r \leq 3+n-2K,\ r \neq j)$, (ii) $u_{k,r}$ is matched in M' to $y_{k,r}$ $(1 \leq r \leq 3+n-2K,\ r \neq l)$, and (iii), each x_q $(1 \leq q \leq n-2K)$ is matched in M' to some $u_{p,q+3}$ $(1 \leq p \leq n)$. Thus M is a matching in G of size K.

To complete the proof, it remains to show that M is maximal. For, suppose not. Then there is some edge $\{v_i, v_k\}$ in G such that no edge of M is incident on either v_i or v_k. Thus $\{u_{i,j}, x_{j-3}\} \in M'$ for some j $(4 \leq j \leq 3+n-2K)$, and $\{u_{k,l}, x_{l-3}\} \in M'$ for some l $(4 \leq l \leq 3+n-2K)$. But then $\{u_{i,j}, u_{k,l}\}$ blocks M', a contradiction. □

6.2 Approximating maximum weakly stable matchings

Given the NP-hardness of each of the problems of finding a weakly stable matching of maximum and minimum size in an instance of SRTI, the approximability of each of these problems is of interest. It turns out that each is approximable within a constant factor of 2; indeed, in a given instance of SRTI, the cardinality of any weakly stable matching is at most twice the size of the minimum, and is also at least half the size of the maximum, as we now demonstrate.

Theorem 6.2. *Let I be an instance of SRTI. Then the size of a maximum cardinality weakly stable matching is at most twice the size of a minimum cardinality weakly stable matching.*

Proof. Let M, M' be weakly stable matchings in I of maximum and minimum cardinality respectively. Let R be the set of participants who are matched in M but not in M', let S be those participants who are matched in M' but not M, and let T be those participants who are matched in both M and M'. Let $r = |R|$, $s = |S|$ and $t = |T|$. No pair of participants $p, q \in R$ can find each other acceptable, for otherwise $\{p, q\}$ blocks M'. Thus each participant in R is matched in M to some participant in T, so that $t \geq r$. Also

$$|M| = \frac{r+t}{2} \leq t \leq 2\left(\frac{s+t}{2}\right) = 2|M'|$$

as required. □

6.3 Weakly stable matchings in SRT

As mentioned in Section 1, Ronn [16] has shown that the following decision problem is NP-complete:

Name: WEAKLY STABLE MATCHING IN SRT
Instance: Arbitrary instance I of SRT
Question: Does I admit a weakly stable matching?

Ronn's proof is based on a transformation from 3SAT [3, problem LO2]. Here we suggest an alternative, shorter transformation, as a Corollary to Theorem 6.1.

In order to prove NP-completeness for WEAKLY STABLE MATCHING IN SRT, we create a new instance I' of SRT from the instance I of SRTI as constructed in Theorem 6.1 as follows. For any participant p in I who has a preference list L_p in I that is incomplete, we create two new people, p' and p''. The preference lists of p, p' and p'' in I' are as follows:

$$\begin{array}{rl} p: & L_p \;\; p' \;\; p'' \;\; \ldots \\ p': & p'' \;\; p \;\; \ldots \\ p'': & p \;\; p' \;\; \ldots \end{array}$$

In a participant's preference list in I', the symbol '...' denotes all remaining people in I' in arbitrary strict order.

Given any complete weakly stable matching M in I, we form a complete matching M' in I' by adding the pairs $\{p', p''\}$ to M, for every participant p who has an incomplete preference list in I; clearly M' is weakly stable in I'. Conversely, given a weakly stable matching M' in I', we claim that, for any participant p who has an incomplete list in I, p is matched in M' to some participant appearing in L_p (and hence $\{p', p''\} \in M'$). For if $\{p, p'\} \in M'$, then $\{p', p''\}$ blocks M' in I'; if $\{p, p''\} \in M'$, then $\{p, p'\}$ blocks M' in I'; finally if p has a partner in M less desirable than p'', then $\{p, p''\}$ blocks M' in I'. Thus we may form a complete matching M in I by removing from M' all pairs of the form $\{p', p''\}$; clearly M is weakly stable in I.

Hence the above transformation, together with the reduction in the proof of Theorem 6.1, establishes the following:

Corollary 6.3. WEAKLY STABLE MATCHING IN SRT *is NP-complete.*

Note that, in an instance of SRT constructed by the reduction of Corollary 6.3, any tie is of length 2 and occurs at the head of a given preference list, as is the case in Ronn's reduction. A simpler reduction exists, also starting from EXACT MAXIMAL MATCHING, if we are not concerned with the restrictions on the ties.

7 Conclusions and open problems

For a given instance of SRT, the linear-time solvability of the problem of deciding whether a super-stable matching exists, as demonstrated by Algorithm SRT-Super, contrasts with the NP-completeness of the analogous problem under weak stability. In addition, in contrast with the super-stability case, weakly stable matchings in an instance of SRTI may be of different cardinalities. It is open as to whether there exist approximation algorithms with performance guarantees better than 2 for the problems of finding minimum and maximum cardinality weakly stable matchings in an instance of SRTI.

Note that a third type of stability, *strong stability*, may be applied to SRT. A matching M in an instance of SRT is *strongly stable* if there are no two participants x and y such that x strictly prefers y to his partner in M, and y either strictly prefers x to his partner in M or is indifferent between them. Clearly a super-stable matching is strongly stable, and a strongly stable matching is weakly stable. Irving [9] has described an $O(n^4)$ analogue of Algorithm SMT-Super for SMT under strong stability. The existence of a polynomial-time algorithm for SRT under strong stability is open. However, the more general problem of deciding whether an instance of SRP admits a strongly stable matching is NP-complete [13].

An additional open problem concerns structural aspects of SRT under super-stability. In the no-ties case, Phase 2 of Algorithm SR [7] may be described in terms of *rotation*

eliminations [5, Section 4.2.3]. Rotations play a key role in efficient algorithms for solving various problems connected with SR, for example finding all stable pairs (i.e. all pairs of participants $\{x, y\}$ such that $\{x, y\}$ belongs to some stable matching), finding a *minimum regret* stable matching (i.e. a stable matching for which the k such that every participant receives their kth choice partner or better is minimised) and for generating all stable matchings (see Sections 4.4.2, 4.4.3 and 4.4.4 of [5] respectively). It remains open to define and exploit the more general concept of a rotation in the context of SRT under super-stability.

Finally, we finish with an open problem regarding 'succinct certificates' in an instance of SRT under super-stability. As mentioned in Section 1, for a given instance of SR with n participants, the construction of a *stable partition* (which may be accomplished in $O(n^2)$ time [18]) provides a means of verifying (in $O(n)$ time) the case that no stable matching exists. Can this structure be extended to SRT in order to provide a means of verifying the case that no super-stable matching exists?

Acknowledgement

The authors would like to thank Sandy Scott for observations which led to improvements in the presentation of Lemmas 3.4 to 3.6.

References

[1] T. Feder, N. Megiddo, and S. Plotkin. A sublinear parallel algorithm for stable matching. In *Proceedings of SODA '94: the 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 632–637. ACM-SIAM, 1994.

[2] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA., 1979.

[4] D. Gusfield. The structure of the stable roommate problem – efficient representation and enumeration of all stable assignments. *SIAM Journal on Computing*, 17(4):742–769, 1988.

[5] D. Gusfield and R.W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[6] J.D. Horton and K. Kilakos. Minimum edge dominating sets. *SIAM Journal on Discrete Mathematics*, 6:375–387, 1993.

[7] R.W. Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6:577–595, 1985.

[8] R.W. Irving. On the stable room-mates problem. Technical Report CSC/86/R5, University of Glasgow, Department of Computing Science, 1986.

[9] R.W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48:261–272, 1994.

[10] R.W. Irving, D.F. Manlove, and S. Scott. The Hospitals/Residents problem with Ties. In *Proceedings of SWAT 2000: the 7th Scandinavian Workshop on Algorithm Theory*,

volume 1851 of *Lecture Notes in Computer Science*, pages 259–271. Springer-Verlag, 2000.

[11] K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita. Stable marriage with incomplete lists and ties. In *Proceedings of ICALP '99: the 26th International Colloquium on Automata, Languages, and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 443–452. Springer-Verlag, 1999.

[12] D.E. Knuth. *Stable Marriage and its Relation to Other Combinatorial Problems*, volume 10 of *CRM Proceedings and Lecture Notes*. American Mathematical Society, 1997. English translation of *Mariages Stables*, Les Presses de L'Université de Montréal, 1976.

[13] D.F. Manlove. NP-completeness of stable marriage with partially ordered lists under strong stability. Unpublished manuscript, 2000.

[14] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. To appear in *Theoretical Computer Science*, 2002.

[15] E. Ronn. *On the complexity of stable matchings with and without ties*. PhD thesis, Yale University, 1986.

[16] E. Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11:285–304, 1990.

[17] A.E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984.

[18] J.J.M. Tan. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12:154–178, 1991.

[19] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 18(1):364–372, 1980.