A Study Of Stable Marriage Problems With Ties

by

Sandy Scott

A thesis submitted to the
Faculty of Information and Mathematical Sciences
(Department of Computing Science)
at the University of Glasgow
for the degree of
Doctor of Philosophy

January 2005

Abstract

We study a number of variants of the *Stable Marriage* problem. Such problems have a long history, but there has been an upsurge in interest in recent years as the various possibilities that arise when ties are allowed in preference lists have been studied. The inclusion of ties in preference lists gives rise to three different versions of stability, so-called *super-stability*, *strong stability* and *weak stability*. The study of these three versions has thrown up a number of challenging problems, and this thesis contributes a range of new results in some of these areas.

The first variant that we study is the Hospitals/Residents problem with ties, a many-to-one variant of the Stable Marriage problem. We present two different polynomial-time algorithms for finding a strongly stable matching, one favouring the residents and the other the hospitals. We also study the *Stable Roommates* problem with Ties, a non-bipartite variant of the Stable Marriage problem, and we again present a polynomial-time algorithm for finding a strongly stable matching.

We then introduce the *Stable Fixtures* problem, a many-to-many variant of the Stable Roommates problem, initially focusing on the version in which preferences are strict. We present an algorithm, which runs in time linear in the input size, to find a stable matching. We then consider the problem when ties are allowed in the preference lists, and we present an algorithm, again linear in the input size, to find a super-stable matching.

We also study the structure underlying the set of super-stable matchings for the Stable Marriage problem with ties (SMT). We extend the concept of a *rotation*, essentially the minimum difference between stable matchings, to super-stability, and show that we can construct a directed acyclic graph to represent precedence amongst these *meta-rotations*. We then use this structure to show that we can find an *egalitarian* super-stable matching, a *minimum regret* super-stable matching and all the *super-stable pairs* in polynomial-time, and generate all the super-stable matchings with polynomial-time between successive matchings.

We then explore some of the issues arising in weak stability, where a number of the key problems are known to be NP-hard. We show a relationship between the sizes of weakly stable matchings and the size of a strongly stable matching in an instance of SMT, and also in a number of the other variants. We give an improved approximation algorithm

for finding a maximum cardinality weakly stable matching when ties are sparse. Efficient generation of weakly stable matchings remains an open problem. As a first step in this direction we show how to determine whether an instance of SMT admits a unique weakly stable matching. However, we also show that the problem of determining whether there is a weakly stable matching for an instance of SMT in which some pairs are *forbidden* is NP-complete. By contrast we present a linear-time algorithm for finding a super-stable matching in such an instance.

Finally we consider the special case of SMT in which one or both sets of participants have preference lists which are sublists of a master list. We show that many weak stability problems remain NP-hard even with this remarkably strong restriction, though there are a limited number of exceptions. We show that we can find an egalitarian weakly stable matching, a minimum regret weakly stable matching, and a *man minimum regret* weakly stable matching in time linear in the number of men in the instance, if all preference lists are complete and both sets have lists that are sublists of a master list. Under the same restrictions we show that we can find all the stable pairs in time linear in the number of stable pairs, and generate all the weakly stable matchings with time linear in the number of men between successive generations. We also show that we can find a minimum regret weakly stable matching in time sublinear in the input size if only one of the sets has lists inherited from a master list, as long as that master list is strict.

Contents

List of Figures

Acknowledgements

I would like to thank Rob Irving for his supervision, and the help and advice he has provided. Rob has been a constant source of inspiration and constructive criticism, without which this thesis would be much the poorer. His obvious enjoyment of the subject has been a source of great motivation for me. I would also like to thank David Manlove, my second supervisor, for his time and advice.

My research was supported by a University of Glasgow 2001 Postgraduate Research Scholarship.

Finally, I would like to thank my parents for their support.

Declaration

This thesis is submitted in accordance with the rules for the degree of Doctor of Philosophy at the University of Glasgow. None of the material contained herein has been submitted for any other degree. The ideas underlying Sections 7.3 and 7.4 are due to Rob Irving, as is the observation that Lemma 1 of [40] can have master lists imposed directly (see Chapter 8). The current forms of the proofs of Lemmas 2.2.3 and 2.2.4 are due to David Manlove, while the current forms of Lemma 4.2.6, and Algorithm MinReg in Section 6.8 are due to Rob Irving. Otherwise all the results contained herein are claimed as original.

Publications

R.W. Irving, D.F. Manlove and S. Scott. Strong Stability in the Hospitals/Residents Problem. In *Proceedings of STACS 2003: the 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 439-450. Springer-Verlag, 2003. (This paper is based on Sections 2.1, 2.2 and 2.3.)

M. Halldórsson, R.W. Irving, K. Iwama, D.F. Manlove, S. Miyazaki, Y. Morita and S. Scott. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*, 306:431-447, 2003. (This paper includes material based on Section 7.3.)

R.W. Irving and S. Scott. An Algorithm for the Stable Fixtures Problem. Submitted to *Discrete Applied Mathematics*, 2002. (This paper is based on Chapter 4.)

Glossary

Here we include a number of terms which are defined in Chapter 1 and are used throughout. Others terms are defined locally so we do not include them here.

Chapter 1

A selective review of Stable Marriage literature

1.1 Introduction

In 1962, David Gale and Lloyd Shapley published a paper entitled "College Admissions and the Stability of Marriage" [8] in which, in the course of presenting an algorithm for matching applicants to college places, they introduced and solved the *Stable Marriage* problem. This problem involves a set of men and a set of women, each of whom have ranked all the members of the other set in strict order of preference. The aim is to find a one-to-one matching between the men and the women such that there is no unmatched couple each of whom prefers the other to their partner in the matching. The authors used their solution to this problem as a basis for solving the extended problem where one of the sets consists of college applicants, and the other consists of colleges, each of which has a quota of places to fill. Since the publication of this paper many additional variants of the Stable Marriage problem have been discussed in the literature. The problem has engendered interest from a number of communities, most notably from game theorists, economists and algorithmists. Additionally, there are a number of matching schemes in operation around the world which make use of some form of the algorithm presented by Gale and Shapley. The best known of these is the NRMP (National Residents Matching Program) in the United States of America [44], which has used a form of the Gale-Shapley algorithm since 1952, predating the publication of the paper by some ten years. The

Canadian Resident Matching Service [2] and the recent Scottish Pre-Registration House Officer Allocations (SPA) scheme [50], which was implemented by Low [32] as a result of work conducted at the University of Glasgow by Irving [23], are both also of interest in this context. Here we are interested in the Stable Marriage problem from an algorithmic point of view. In this chapter we study how the field has evolved from the publication of "College Admissions and the Stability of Marriage" through to the present day.

1.2 The classical Stable Marriage problem

An instance of the classical Stable Marriage problem (SM) involves two disjoint sets of size n, the *men* and the *women*. In line with the terminology of [15], we say that such an instance is of *size* n. Note, however, that the total amount of data for an instance of size n is actually $O(n^2)$. Each person has a *strictly* ordered *preference list* containing *all* the members of the other set, such that a person p *prefers* a person q to a person r if and only if q precedes r on p's preference list.

For such an instance, a *matching* M is a one-to-one mapping between the men and the women. For a pair $(m, w) \in M$, we say that m is the *partner* of w in M, or $m = p_M(w)$, and similarly for w. A *blocking pair* for a matching M is a man-woman pair $(m, w) \notin M$, each of whom prefers the other to their partner in M. A matching is *unstable* if it admits one or more blocking pairs, and is otherwise *stable*. A pair (m, w) which appears in some stable matching is a *stable pair*. Given a matching M we can verify, in time linear in the input size, if M is stable, with suitable data structures to represent the preference information (see, for example, [8]). The method is simple. For each man m we check, for every woman whom he prefers to his partner in M, whether that woman prefers m to her partner in M. If so we have a blocking pair, but if there are no blocking pairs then M is stable.

Gale and Shapley [8] proved that, for any instance of the classical Stable Marriage problem, there is at least one solution, and they presented an algorithm, now widely known as the Gale/Shapley (GS) algorithm, to find a solution. The algorithm runs in $O(n^2)$ time for an instance of size n [31], and thus is linear in the input size. Gale and Shapley also noted that this algorithm, which involves a sequence of proposals from the men to the women, produces a matching which is *man-optimal*, in the sense that every man has the best

partner he can have in any stable matching. We say that the algorithm is *man-oriented*, and if the roles of the men and women are reversed then the *woman-oriented* algorithm will produce the *woman-optimal* matching. Returning to the man-optimal stable matching, McVitie and Wilson later observed that this man-optimality was at the expense of the women as each woman has the worst partner she can have in any stable matching [42]. Hence the man-optimal stable matching is also *woman-pessimal*. There is an element of non-determinism inherent in the algorithm, in that the order in which the men propose is not given, but Gusfield and Irving [15] noted that, whatever the order of the proposals, the resulting matching is always the same. This result was implicit in [8]. An extended version of this algorithm simplifies the proposal process by deleting from a woman w's list every man m who succeeds a man from whom she has received a proposal, and correspondingly deleting w from m's list. It can be shown that each such pair (m, w) cannot be stable. Such deletions are now standard practice in stable marriage algorithms. This version of the algorithm, the extended Gale-Shapley algorithm, has the additional property that all the stable matchings are contained in the *GS-lists*, the lists obtained from taking the intersection of the preference lists after applying separately the man-oriented and woman-oriented versions of the algorithm [15].

1.3 Simple extensions of the classical problem

There are two simple extensions to the classical Stable Marriage problem - sets of unequal size and incomplete preference lists.

1.3.1 Sets of unequal size

If sets of unequal size are allowed, then it is clear that some person(s) must end up unmatched in any matching. The definition of a blocking pair must be extended to cover this. To this end a pair $(m, w) \notin M$ blocks M if

- m is either unmatched, or prefers w to his partner in M, and
- w is either unmatched, or prefers m to her partner in M.

It is known that, for an instance of SM in which the sets of men and women are of unequal size, there is at least one stable matching in which all the members of the smaller set are matched. Furthermore, all the members of the smaller set are matched in all stable matchings, and the larger set is partitioned into two subsets, the members of one subset being matched in all stable matchings and the members of the other in none [41]. It can be shown that an instance of SM with sets of unequal size has exactly the same set of stable matchings as the same instance with the unmatched people deleted, so the results detailed above for the classical case carry over.

Henceforth we assume, unless stated otherwise, that all instances of SM may have sets of unequal size.

1.3.2 Incomplete lists

We now consider SM when incomplete preference lists are allowed. We denote this problem SMI (Stable Marriage with Incomplete lists). If a person wishes to declare one or more members of the other set unacceptable as a partner then they are omitted from that person's preference list. A preference list is *consistent* if a man m appears on a woman w's list if and only if w appears on m's list. Consistency of preference lists will be assumed throughout, as it can be simply enforced in time linear in the input size without changing the set of stable matchings. A man m and a woman w are *mutually acceptable*, or just *acceptable* in view of the consistency of the preference lists, if they each appear on the other's preference list. The most obvious effect of this extension is that some matchings may not be complete, i.e., there may be people who are unmatched in any given matching. Thus we must redefine the concepts of a matching and a blocking pair. Let A be the set of acceptable pairs in an instance I of SMI, and denote $|A|$ by a. A *matching* M is a subset of A such that $|\{m : (m, w) \in M\}| \leq 1$ for all m and $|\{w : (m, w) \in M\}| \leq 1$ for all w. A pair $(m, w) \in A \setminus M$, *blocks* M if

- m is either unmatched, or prefers w to his partner in M, and
- w is either unmatched, or prefers m to her partner in M.

It is known that, in an instance of SMI, the men and the women are each partitioned into two sets - those that have partners in all stable matchings and those that have partners in

none [9]. It is also known that, as in the case of sets of unequal size, an instance of SMI has the same set of matchings as the SM instance obtained by deleting the unmatched pairs and then, for each remaining person p, appending all remaining people of the opposite sex not already appearing on p's list, so again all the results detailed above carry over. We often use a as a measure for time complexity, where a is the number of acceptable pairs in a problem instance.

Finally we mention in brief SMIF (Stable Marriage with Incomplete Lists and Forbidden pairs). This problem involves an instance of SMI and a set F of forbidden pairs. We look for a stable matching which does not include a pair in F. Note that a pair from F, while it cannot appear in a matching, can block a matching, thus differentiating a forbidden pair from an unacceptable pair. Dias et al. [5] give an algorithm, linear in the input size, to determine whether an instance of SMF (Stable Marriage with Forbidden pairs) admits a stable matching, and if so the algorithm outputs one.

1.4 The structure of Stable Marriage

Now we consider the structure of the set of all stable matchings for an instance of SMI. Thus far we have only encountered the man- and woman-optimal stable matchings, but there may be other stable matchings (or indeed the man-optimal and woman-optimal stable matchings may coincide, in which case there is clearly only one stable matching). In fact, the set of all stable matchings for a given instance of the classical stable marriage problem forms a distributive lattice with the man- and woman-optimal stable matchings representing the two extreme elements of the lattice [31]. It is known that for each $n > 0$, n a power of 2, there is an instance of SM of size n with at least 2^{n-1} stable matchings [24]. Thus the maximum number of stable matchings for a given instance grows exponentially as the size of the instance increases. It follows that, for efficient algorithms to exploit this lattice structure, a compact representation of the structure is required. Such a representation was discovered by Irving and Leather [24], and we describe it below.

For an instance I of SMI, the *shortlists* are the non-empty preference lists after the application of the man-oriented extended Gale-Shapley algorithm to I. A *reduced set of preference lists* is a set obtainable from the shortlists by zero or more deletions such that:

1. no list is empty;

2. woman w is absent from man m's list if and only if m is absent from w's list.

For a given set $\mathcal{L}$ of reduced preferences we denote the first, second and last people on x's list by $f_{\mathcal{L}}(x)$, $s_{\mathcal{L}}(x)$ and $l_{\mathcal{L}}(x)$ respectively, dropping the subscript if it is obvious which set of lists applies. Note that if x has a list of length 1 then $f(x)$ and $l(x)$ will coincide, while $s(x)$ will be undefined.

For ease of exposition, we say that a person prefers every person who appears on their initial preference list to any person who does not. A set $\mathcal{S}$ of reduced preference lists is *stable* if, for each man m and woman w,

1. $w = f_{\mathcal{S}}(m)$ if and only if $m = l_{\mathcal{S}}(w)$;

2. w is absent from m's list if and only if w prefers $l_{\mathcal{S}}(w)$ to m.

The shortlists are a stable set [24].

It is known that, if each man is matched with the first woman on his list in a stable set $\mathcal{S}$, then the result is a stable matching [24]. We call the matching so derived the *matching derived from* $\mathcal{S}$, and we denote it by $M_{\mathcal{S}}$. In fact, for a given SMI instance, there is a one-to-one correspondence between the stable sets and the stable matchings for that instance [24].

A *rotation* ρ is an ordered sequence of pairs $\{(m_0, w_0), (m_1, w_1), ..., (m_{r-1}, w_{r-1})\}$ in a stable set $\mathcal{S}$ where each m_i is a man and each w_i a woman, $r \geq 2$, $w_{i+1} = f(m_{i+1}) = s(m_i)$ $(0 \leq i \leq r-1)$, and the subscripts are taken modulo r. Such a rotation is said to be *exposed* in $\mathcal{S}$. We also say that a rotation is exposed in a stable matching M if it is exposed in the stable set from which M is derived. To *eliminate* a rotation we delete from $\mathcal{S}$ the pairs (m, w_{i+1}) for each i, and for all successors m of m_i on w_{i+1}'s list. The resulting set of lists is denoted by $\mathcal{S} \setminus \rho$. It is known that, if a rotation is eliminated from a stable set $\mathcal{S}$ in which it is exposed, then the resulting set of reduced preference lists is also stable [24]. Further, for a stable set $\mathcal{S}$, and a stable matching M in which man m has a poorer partner than $p_{M_{\mathcal{S}}}(m)$, there is a rotation exposed in $\mathcal{S}$ all of whose male members have worse partners in M than in $M_{\mathcal{S}}$ [24]. It follows that there is a rotation exposed

in any stable set unless the matching derived from the stable set is the woman-optimal stable matching for the original SMI instance. As noted above, for a given SMI instance, there is a one-to-one correspondence between the stable sets and the stable matchings, and further, each stable set can be obtained from the shortlists by a sequence of zero or more rotation eliminations [24].

A rotation ρ_1 is said to *precede* a rotation ρ_2 if ρ_1 must be eliminated before any stable set in which ρ_2 is exposed can be obtained. It is possible to find the rotations for an instance of SM in $O(n^2)$ time [13]. We say that a pair is deleted by a rotation if it is deleted when that rotation is eliminated from a stable set in which it is exposed. It is known that no pair is deleted by more than one rotation [24]. When finding the rotations it is an easy matter to note which rotation any given pair is deleted by, and whether that pair is in the rotation it is deleted by. We denote by $\rho_{(m,w)}$ the rotation that deletes the pair (m, w). Clearly a given rotation may have more than one such designation. We can then construct a representation of the *rotation poset*, where the partial order is that of precedence defined above, in $O(a)$ time, using the algorithm in Figure 1.1 [15].

```
for each man m
{   ϱ := null;
    set all women on m's list to be unmarked;
    while there is some unmarked woman on m's list
        let w be the first unmarked woman on m's list;
        if (m, w) ∈ ρ(m,w)
        {   make ϱ precede ρ(m,w);
            ϱ := ρ(m,w);}
        else
            make ρ(m,w) precede ϱ;}
```

Figure 1.1: Algorithm POSET: constructing a representation of the rotation poset for an instance of SMI

A *closed subset* of a poset P is a subset S of the elements in P, such that, for each element $e \in S$, every predecessor f of e in P is in S. It is known that the stable matchings for an instance of SMI are in one-to-one correspondence with the closed subsets of the rotation poset for that instance [24].

1.5 Exploiting the lattice

There are a number of problems which can be solved by exploiting the rotation poset, or by making use, directly or indirectly, of rotations. These problems include generating all the stable matchings for an instance of SM, and determining whether a given pair is stable. As was noted previously, the number of stable matchings for a given instance of SM may be exponential in the size of the instance. However, Gusfield [13] showed that, given the rotation poset, the set of stable matchings for an instance I of SM can be generated in $O(nk)$ time, for a problem of size n with k matchings. Additionally Gusfield showed that all the stable pairs for I can be found in $O(n^2)$ time. Indeed, it can be shown that every stable pair must either appear in some rotation, or in the woman-optimal stable matching.

Furthermore, the algorithms for finding stable matchings discussed previously all find matchings that are optimal for one of the two sets involved in the matching process, and we have also seen that this optimality forces the matching to be pessimal for the second set. This raises the question: Can we find a matching that treats the two sets more equally? Below we describe two possible "equitable" matchings.

1.5.1 An egalitarian stable matching

If $mr(m, w)$ is the position of woman w in man m's list, and $wr(m, w)$ is the position of m in w's list, then the man-optimal stable matching minimises

$$\Sigma_{(m,w)\in M} mr(m, w)$$

and maximises

$$\Sigma_{(m,w)\in M} wr(m, w)$$

over all stable matchings for a given instance of SM. To obtain a matching which treats the two sets more fairly, the *egalitarian* matching minimises the total *weight* of the matching over all stable matchings, where the weight, $w(M)$ is the total of the two summations above, i.e.,

$$w(M) = \Sigma_{(m,w)\in M}(mr(m,w) + wr(m,w)).$$

Clearly this treats the men and women equally, as the formula being minimised involves a term for each person regardless of their gender. Irving et al. [25] showed that an egalitarian stable matching can be found in $O(n^4)$ time, by first finding all the rotations for the given instance of SM and then exploiting the structure of the rotation poset. Later Feder [6] improved on this bound, with an $O(n^{2.5} \log n)$ time algorithm.

1.5.2 A minimum regret stable matching

For a stable matching M the *regret* of person x is the position in x's preference list of $p_M(x)$. The regret of M is the maximum regret of any person in M. In other words the regret of M is a measure of how badly off the worst-off person in the matching is. The problem of finding a *minimum regret* stable matching over all the stable matchings for a given instance of SM was first proposed by Knuth [31], with a solution presented in [31] attributed to Selkow. Gusfield [13] gave an alternative algorithm that runs in $O(n^2)$ time as opposed to the $O(n^4)$ complexity of Selkow's solution.

1.6 Indifference

A natural extension to the classical problem, which leads to a variety of interesting problems and results, is to allow indifference in the lists. This is highly relevant in practice, as one can easily envisage a matching scheme in which some participant is particularly popular, and is therefore unable or unwilling to produce a strict ranking over all those participants it is required to rank.

The most natural form of indifference involves ties. A set W of k women forms a *tie* of length k in the preference list of man m if m does not prefer w_i to w_j for any $w_i, w_j \in W$ (i.e., m is *indifferent* between w_i and w_j), while for any other woman w who is acceptable to m, either m prefers w to all of the women in W, or m prefers all of the women in W to w. A tie on a woman's list is defined analogously. For convenience we consider an untied entry in a preference list as a tie of length 1 throughout. By the *head* and *tail* of a

preference list we mean the first and last ties respectively on that list. We consider both Stable Marriage with Ties and complete lists (SMT) and Stable Marriage with Ties and Incomplete lists (SMTI).

Given an instance I of SMT(I), we often wish to work with one or more instances of SM(I) which are obtainable from I by breaking the ties. Such an instance of SM(I) is called a *derived* instance.

The inclusion of indifference forces a re-evaluation of the concept of stability, via blocking pairs. We could view a pair, not in a matching, as blocking that matching if, by coming together,

- both parties would be better off, or
- neither party would be worse off, or
- one party would be better off and the other no worse off.

These three possibilities give rise to the notions of weak stability, super-stability, and strong stability, respectively, first considered by Irving [22]. We now formally define these three forms of stability, by defining a blocking pair for each case, and we discuss the properties they exhibit in both SMT and SMTI.

1.6.1 Stable Marriage with ties

Weak Stability

The first notion of stability that we consider is the weakest of the three. Let I be an instance of SMT. A pair which does not appear in M blocks M if each member of the pair prefers the other to their partner in M. If there is no blocking pair for M, M is said to be *weakly stable*. Alternatively M is weakly stable in I if it is stable in *at least one* instance I' of SM which can be derived from I, a result which is implicit in [10] and was explicitly noted in [15].

A weakly stable matching exists for every instance of SMT, and can be found by forming a derived instance of SM, and applying the GS algorithm [10]. However, beyond this, weak

stability has proven to be a difficult concept. For example, it is known that the problems of finding an egalitarian weakly stable matching and finding a minimum regret weakly stable matching are both NP-hard, even if the ties are on one side only, there is at most one tie per list and each tie has length 2 [40]. It has since been shown that both problems are also not approximable within $\Omega(n)$ unless P=NP [16]. Additionally, determining whether a given man-woman pair is weakly stable is NP-complete, even if the ties are at the tails of lists and on one side only, and each tie has length 2 [40]. In fact, there is an instance of SMT which admits neither a man-optimal weakly stable matching nor a woman-optimal weakly stable matching [47], thus apparently precluding the existence of a lattice structure similar to that discussed for SM.

Super-stability

The second notion of stability we consider is the strongest of the three. Let I be an instance of SMT. In this case, a pair which does not appear in M blocks M if each member of the pair prefers the other to their partner in M, or is indifferent between them. If there is no such blocking pair for M, M is said to be *super-stable*. Alternatively a matching is super-stable if it is stable in *every* instance I' of SM which can be derived from I [35].

It can be shown that an instance of SMT may admit no super-stable matching (trivially, if each preference list for a given instance is a complete tie, so no strict preferences are stated, then there can be no super-stable matching), but for an instance with n men and n women, there is an $O(n^2)$ algorithm which determines if such a matching does exist, and if so finds one [22].

Additionally the set of super-stable matchings for a given instance of SMT forms a finite distributive lattice. This was first proved by Spieker [51], but an alternative and more accessible proof was given by Manlove [38], in which he also showed that if a person has different partners in two super-stable matchings then he or she cannot be indifferent between them.

Note that indifference can also represent incomplete information. For example, suppose we do not know whether a man m prefers a woman w_1 to a woman w_2, or vice versa. Then, if we place w_1 and w_2 in a tie on m's list and find a super-stable matching for the instance, we can conclude that it does not matter what m's preference is, as this matching will be stable

in every instance of SM which can be derived from the original instance of SMT. Thus super-stable matchings are the matchings of choice when indifference represents incomplete information.

Strong Stability

The third and final notion of stability we consider falls between the two versions of stability already described. Let I be an instance of SMT. In this intermediate case, a pair which does not appear in M blocks M if one member of the pair prefers the other to their partner in M while the other is at least indifferent. If there is no such blocking pair for M, M is said to be *strongly stable*. Alternatively, the following is shown to be an equivalent definition of a strongly stable matching in [35]: a matching M is strongly stable in an instance I of SMT if and only if

1. there is some instance I' of SMT obtainable from I by breaking the ties on the men's side, such that for every instance of SM obtainable from I' by breaking the ties (on the women's side), M is stable, and
2. there is some instance I' of SMT obtainable from I by breaking the ties on the women's side, such that for every instance of SM obtainable from I' by breaking the ties (on the men's side), M is stable.

It is clear that a super-stable matching is strongly stable, and a strongly stable matching is weakly stable.

It is known that, for a given instance of SMT, a strongly stable matching may not exist [10]. For example, in the instance of SMT in Figure 1.2, in which parentheses enclose people tied in a preference list (this notation is used throughout), the matching $\{(m_1, w_1), (m_2, w_2)\}$ is blocked by (m_2, w_1) while the matching $\{(m_1, w_2), (m_2, w_1)\}$ is blocked by (m_2, w_2). However, for an instance of SMT with n men and n women, there is an $O(n^4)$ algorithm which determines if such a matching does exist, and if so finds one [22] [1].

As with the set of super-stable matchings for a given instance of SMT, the set of strongly stable matchings forms a finite distributive lattice [38], though in this case the strongly

[1] Kavitha et al. [30] have recently published an algorithm that finds a strongly stable matching in SMTI in $O(ka)$ time, if one exists, where k is the total number of men and women in the instance.

$m_1 : w_1\ w_2$ $\quad$ $w_1 : m_2\ m_1$

$m_2 : (w_1\ w_2)$ $\quad$ $w_2 : m_2\ m_1$

Figure 1.2: An instance admitting no strongly stable matching

stable matchings must first be grouped into equivalence classes. To this end, two matchings M and M' are in the same equivalence class if and only if every man is indifferent between M and M'.

1.6.2 Stable Marriage with ties and incomplete lists

Each of the three versions of stability displays different properties when the preference lists are incomplete.

1.6.3 Weak stability with incomplete lists

A matching M in an instance I of SMTI is weakly stable if and only if M is stable in some instance I' of SMI which can be derived from I. The definition of a blocking pair is a straightforward extension of that for SMT, taking into account the possibility that a person may be unmatched in a given matching. Again, a weakly stable matching exists for every instance of SMTI, and can be found by forming a derived instance of SMI, and applying the GS algorithm [10].

Weak stability has proven to be an even more difficult concept with incomplete lists. Most intriguingly, in an instance of SMTI it is known that weakly stable matchings may have different cardinality. Furthermore, finding the maximum (or minimum) cardinality weakly stable matching for a given instance of SMTI is NP-hard. This holds even if the ties are at the tails of lists and on one side only, and each tie has length 2 [40], though the largest matching is at most twice the size of the smallest [40]. More recently, it has been established that these problems are not approximable within δ, unless P=NP, for some $\delta > 1$, even if the preference lists are of constant length, there is at most one tie per list, and the ties occur on one side only [16]. We denote the problem of finding a maximum (resp. minimum) cardinality weakly stable matching by Maximum (resp. Minimum) Cardinality

SMTI.

Above we noted that a maximum cardinality weakly stable matching is at most twice the size of a minimum cardinality weakly stable matching. Hence if we break all ties in an arbitrary way and apply the GS algorithm to the resulting instance of SMI we get what is simultaneously an approximation algorithm for both Maximum Cardinality and Minimum Cardinality SMTI with a performance ratio of 2. In Section 7.3 we demonstrate an improved performance bound for instances of SMTI with sparse ties. This work appeared in [16]. Recently three other pieces of work relating to approximating maximum cardinality weakly stable matchings have appeared in the literature. In [18], Halldórsson et al. present a randomised approximation algorithm with expected performance guarantee $\frac{10}{7}$ for instances of SMTI in which ties occur on one side only, there is at most one tie per list, and each tie has length 2. In [17], the same authors present an approximation algorithm with performance guarantee $\frac{2}{(1+\frac{1}{L^2})}$ for instances of SMTI in which ties occur on one side only, and each tie has length at most L. Additionally they show a ratio of $\frac{13}{7}$ where ties are allowed on both sides, and are of length 2. Finally, in [29] Iwama et al. present an approximation algorithm for a general instance of SMTI with guarantee $2 - c\frac{\log(n)}{n}$, for an instance of size n, where c is an arbitrary positive constant.

Additionally, for a given instance I of SMTI, the following problems are all known to be NP-complete [39]:

- determining whether a given person is matched in some weakly stable matching, even if the ties are at the tails of lists, on one side only, and each tie has length 2;
- given a complete weakly stable matching S, determining whether I admits a complete weakly stable matching $T \neq S$, even if the ties occur on one side only, there is at most one tie per list, and each tie has length 2;
- given a weakly stable matching S, determining whether I admits a weakly stable matching T such that $|T| < |S|$, even if the ties occur on one side only, there is at most one tie per list, and each tie has length 2;
- given a weakly stable matching S, determining whether I admits a weakly stable matching of size $|S| + 1$.

One of the rare positive results to have been proven with regard to SMTI under weak stability is that weak stability is an *interpolating invariant*, i.e., for an instance I of SMTI, there is a matching of size k, weakly stable with respect to I, for each $p \leq k \leq q$, where p is the size of a minimum cardinality weakly stable matching in I and q is the size of a maximum cardinality weakly stable matching in I [40].

In Chapters 7 and 8 we present a number of results relating to weak stability in SMT and SMTI, and relating to some restrictions of these problems which are of interest.

1.6.4 Super-stability with incomplete lists

A matching M in an instance I of SMTI is super-stable if and only if it is stable in every instance I' of SMI which can be derived from I. The definition of a blocking pair is again a straightforward extension of that for SMT, taking into account the possibility that a person may be unmatched in a given matching.

It can be shown that an instance of SMTI may admit no super-stable matching (for the same reason as with complete lists), but, for an instance with a acceptable pairs, there is an $O(a)$ algorithm which determines if such a matching does exist, and if so finds one [34]. This algorithm, Algorithm SUPER2, is heavily used in later chapters, so we include it in Section 1.7 below for completeness. It is also known that the existence of a super-stable matching M for a given instance of SMTI forces all weakly stable matchings for that instance to have the same size, namely $|M|$ [27].

For a given instance of SMTI, if M and M' are two super-stable matchings, then, for any person p in the instance, p is matched in M if and only if p is matched in M' [34]. The main consequence of this is that, for a given instance of SMTI, the set of people can be partitioned into two disjoint sets, those matched in all super-stable matchings, and those matched in none. In view of this and the fact that the set of super-stable matchings for a given instance of SMT forms a finite distributive lattice, it can be shown that the set of super-stable matchings for a given instance of SMTI forms a finite distributive lattice [34]. Further, since a super-stable matching M for an instance I of SMTI is stable in every instance I' of SMI which can be derived from I, and we can partition the people in each I' into those who are matched in every stable matching and those who are matched in none, it follows that a person p is matched in M if and only if p is matched in every weakly

stable matching for I.

We show in Chapter 6 that there is a non-trivial extension of the concept of a rotation which allows us to exploit the lattice structure for the set of super-stable matchings in an instance of SMTI in an efficient manner. We make use of this to find all the super-stable pairs and generate all the super-stable matchings for an instance of SMTI, and to find egalitarian and minimum regret super-stable matchings for an instance of SMT.

1.6.5 Strong stability with incomplete lists

The definition of a strongly stable matching in an instance of SMTI, in terms of tie-break instances, is analogous to the definition for SMT, but with each reference to SMT replaced by SMTI. The definition of a blocking pair is again a straightforward extension of that for SMT, taking into account the possibility that a person may be unmatched in a given matching.

It can be shown that, for a given instance of SMTI, a strongly stable matching may not exist (see example 1.2 above), but, for an instance with a acceptable pairs, there is an $O(a^2)$ algorithm which determines if such a matching does exist, and if so finds one [34] [2].

For a given instance of SMTI, if M and M' are two strongly stable matchings, then, for any person p in the instance, p is matched in M if and only if p is matched in M' [34]. The main consequence of this is that, for a given instance of SMTI, the set of people can be partitioned into two disjoint sets, those matched in all strongly stable matchings, and those matched in none. In view of this and the fact that the set of strongly stable matchings for a given instance of SMT forms a finite distributive lattice, it can be shown that the set of strongly stable matchings for a given instance of SMTI forms a finite distributive lattice [34].

Finally we note that Stable Marriage with Ties, Incomplete lists and Forbidden pairs (SMTIF) extends SMIF in a completely straightforward manner. We omit the details.

[2] Kavitha et al. [30] have recently published an algorithm that finds a strongly stable matching in SMTI in $O(ka)$ time, if one exists, where k is the total number of men and women in the instance.

1.7 Algorithm SUPER2

As noted above, the algorithm for finding a super-stable matching in an instance of SMTI, Algorithm SUPER2 [34], is heavily used in later chapters. We therefore present it here. The algorithm involves a sequence of proposals from the men to the women. If a man m has proposed to a woman w then, as long as w has not been deleted from m's preference list, we say that m and w are *engaged*. To *delete* a pair (m, w) is to break any engagement between m and w, and remove m from w's preference list, and w from m's.

Algorithm SUPER2 begins by assigning each person to be free, and recording that every woman has yet to receive a proposal. Then, so long as some man m is free and has a non-empty list, m proposes to each woman w at the head of his preference list, and in so doing becomes engaged to each of them. Additionally it is noted that w has received a proposal. If w is multiply engaged then she cannot be matched in a super-stable matching with any of the men in the tail of her list, and so the pairs (m', w) for each man m' in w's tail, are deleted, and hence any engagements involving any of these pairs are broken. Otherwise, the pairs (m', w) for each man m' to whom w prefers m, are deleted, as w cannot have a partner from amongst these men in any super-stable matching. Once the while loop has terminated, if some woman is not engaged but received a proposal during the execution of the algorithm, or if some man is multiply engaged, then no super-stable matching exists. Otherwise the engagement relation specifies a super-stable matching. Note that this algorithm outputs the man-optimal super-stable matching, and, if the roles of the men and women are reversed, it will output the woman-optimal matching.

1.8 The Hospitals/Residents problem

We now consider the many-to-one extension of SMI which Gale and Shapley [8] called the College Admissions problem, and which is now referred to as the *Hospitals/Residents* problem (HR). An instance of HR involves two sets, a set R of *residents* and a set H of *hospitals*. Each resident in R seeks to be *assigned* to exactly one hospital, and each hospital $h \in H$ has a specified number p_h of posts, referred to as its *quota*. Each resident ranks a subset of H in strict order of preference, and each hospital ranks, again in strict order, those residents who have ranked it, so clearly preference lists may be incomplete.

```
assign each person to be free;
for each woman w
    proposed(w) := false;
while some man m is free and has a nonempty list
    for each woman w at the head of m's list{
        m proposes, and becomes engaged, to w;
        proposed(w) := true;
        if w is multiply engaged and
        w is indifferent between her two fiancés
            for each man m' at the tail of w's list
                delete the pair (m', w)
        else
            for each strict successor m' of m on w's list
                delete the pair (m', w)}
if (some woman w is not engaged and proposed(w))
or some man is multiply engaged
    no super-stable matching exists
else
    output the engagement relation, a super-stable matching;
```

Figure 1.3: Algorithm SUPER2

The pairs contained within the preference lists are the *acceptable pairs*. We do not assume that the number of residents is equal to the number of posts.

Let A be the set of acceptable pairs in an instance I of HR, and denote $|A|$ by a. A *matching* M is a subset of A such that $|\{h : (r,h) \in M\}| \leq 1$ for all $r \in R$ and $|\{r : (r,h) \in M\}| \leq p_h$ for all $h \in H$. For a hospital h we extend the notation $p_M(h)$ to denote the set of residents assigned to a hospital in M. A hospital h is *full* if $|p_M(h)| = p_h$, and is *under-subscribed* if $|p_M(h)| < p_h$. A pair $(r,h) \in A \setminus M$, *blocks* M if

- either r is unmatched in M, or r prefers h to $p_M(r)$, and
- either $|p_M(h)| < p_h$, or h prefers r to at least one member of $p_M(h)$.

A matching M is *stable* if there is no blocking pair for M.

It is known that the Hospitals/Residents problem can be transformed to an instance of SMI by making p clones, h_1, ..., h_p of a hospital h with p posts, with each clone having a quota of one. When h appears on a resident's preference list it is replaced by h_1 to h_p, in that order. It can then be shown that the stable matchings in this derived instance are in one-to-one correspondence with the stable matchings in the original instance. However, this is not the approach that has been most studied in the literature. One weakness of this approach is that it can increase the size of the instance by a factor of $\Omega(n)$, and hence the GS algorithm applied to the resultant instance could be slower by a factor of $\Omega(n)$ when compared to the direct approach described below [36]. Instead it is normal to solve the original instance directly. An instance of HR can be solved by an extension of the GS algorithm for SMI [8, 15]. SMI can be viewed as a restriction of HR in which each hospital has quota 1 (and the residents and hospitals are re-named men and women). As with SMI, a stable matching exists for every instance of HR, and all stable matchings for a given instance have the same size [15]. The extension of the GS algorithm finds one such matching in $O(a)$ time [15]. Recent pressure from student bodies associated with the NRMP has ensured that the extended version of the GS algorithm that is employed by the scheme is now *resident-oriented*, meaning that it produces the *resident-optimal* stable matching for a given instance of HR [43]. This is the unique stable matching M_0 in which every resident assigned in M_0 is assigned the best hospital that he/she could obtain in any stable matching, and any resident unassigned in M_0 is unassigned in any stable matching. Previously the Program had produced the hospital-optimal stable matching (see Theorem 1.8.1 below).

Although an instance of HR may admit more than one stable matching, it is known that every stable matching has the same size, matches the same set of residents and fills the same number of posts at each hospital. Further, any hospital that is under-subscribed in some stable matching is assigned the same set of residents in every stable matching. Collectively these results are known as the Rural Hospitals Theorem [9, 47, 48], because usually it is hospitals in rural areas which end up under-subscribed.

The following result addresses the issue of preference from a hospital's point of view.

Theorem 1.8.1. *Suppose that M and M' are two stable matchings for an instance of HR, and suppose that hospital h is assigned non-identical sets of residents in M and M'. If h prefers the least favoured resident in $p_M(h) \setminus p_{M'}(h)$ to the least-favoured resident in*

$p_{M'}(h) \setminus p_M(h)$, then h prefers all residents in $p_M(h)$ to all residents in $p_{M'}(h) \setminus p_M(h)$.

This result, due to Roth and Sotomayor [49], means that, excluding residents who are assigned to the hospital in both matchings, a hospital prefers all its partners in one matching to all its partners in the other. It is therefore reasonable to say that the hospital prefers the first matching to the second. This result can be used to show, in a manner similar to that adopted earlier, that there is a finite distributive lattice under this dominance relation [15].

1.8.1 The Hospitals/Residents Problem with ties

We now consider HRT, the Hospitals/Residents Problem with Ties. We assume that preference lists may be incomplete, but do not make any assumption as to how the number of residents and posts are related.

Weak Stability

Again we first consider the weakest version of stability. A pair $(r, h) \in A \setminus M$ blocks the matching M if

- either r is unassigned in M, or r prefers h to $p_M(r)$, and
- either $|p_M(h)| < p_h$, or h prefers r to at least one member of $p_M(h)$.

A matching that does not admit any such blocking pair is *weakly stable*. A weakly stable matching exists for every instance of HRT, and can be found by forming a derived instance of HR, and applying the resident-oriented GS algorithm. It turns out that, in contrast to HR, weakly stable matchings for an instance of HRT may have different sizes, though the size of the largest is at most twice the size of the smallest [40]. In addition, the hardness and inapproximability results listed above for SMT and SMTI under weak stability hold for HRT by restriction.

Super-Stability

A pair $(r, h) \in A \setminus M$ blocks the matching M if

- r is unassigned in M, or r prefers h to $p_M(r)$, or is indifferent between them, and
- $|p_M(h)| < p_h$, or h prefers r to at least one member of $p_M(h)$, or is indifferent between r and at least one member of $p_M(h)$.

A matching that does not admit any such blocking pair is *super-stable*. As with SMT(I), it is trivial to show that there are instances of HRT for which no super-stable matching exists. However, there is an $O(a)$ algorithm to determine whether an instance of HRT admits a super-stable matching, and to find one if it does [27]. Additionally, the Rural Hospitals Theorem holds for HRT under super-stability [27].

Strong Stability

A pair $(r, h) \in A \setminus M$ blocks the matching M if either

- r is unassigned in M, or r prefers h to $p_M(r)$, and
- $|p_M(h)| < p_h$, or h prefers r to at least one member of $p_M(h)$, or is indifferent between r and at least one member of $p_M(h)$.

or

- r is unassigned in M, or r prefers h to $p_M(r)$, or is indifferent between them, and
- $|p_M(h)| < p_h$, or h prefers r to at least one member of $p_M(h)$.

A matching that does not admit any such blocking pair is *strongly stable*. As with SMT(I), it is trivial to show that there are instances of HRT for which no strongly stable matching exists. In Chapter 2 we present a resident-oriented algorithm and a hospital-oriented algorithm for HRT under strong stability, and we prove the Rural Hospitals Theorem for strong stability directly. This result was already known [37], but our proof is shorter and follows as a consequence of the correctness of the resident-oriented algorithm.

1.9 Stable Roommates

In the Stable Roommates problem (SR), introduced by Gale and Shapley [8], each member of a set of even cardinality ranks every other member of the set in strict order of preference. In this context a *matching* is a partition of the set into disjoint pairs. Such a matching is *unstable* if there is a pair, not in the matching, each of whom prefers the other to their assigned roommate, and is otherwise *stable*. As in SM, such a pair is said to *block* the matching, or to be a *blocking pair* for the matching. The terminology used for SR is essentially the same as for SM, with the only significant differences being that pairs are unordered, so the notation used to represent pairs is $\{x, y\}$ as opposed to (x, y), and that we refer to *agents* instead of men and women.

Stable Roommates with Incomplete lists is a generalisation of SM. There is, however, also a sense in which SR is a generalisation of SM. Given an instance I of SM involving n men and n women, there is an instance I' (in fact there are many instances) of SR involving these $2n$ agents, such that the stable matchings for I' are precisely the stable matchings for I [15]. There are, nonetheless, also some significant differences between SM and SR. The most striking of these is that there are instances of SR which do not admit a stable matching [8]. For example, in the instance in Figure 1.4, agent 4 cannot be matched with any of the other three agents in the instance without creating a blocking pair. Thus there cannot be a stable matching for the instance.

1 : 3 2 4

2 : 1 3 4

3 : 2 1 4

4 : 1 2 3

Figure 1.4: An instance admitting no stable matching

Knuth [31] presented the possibility of a polynomial-time algorithm to solve SR as an open problem. This question was resolved by Irving [20], who gave an $O(n^2)$ algorithm which determines whether a given instance of SR admits a stable matching, and if so finds one.

As with SM, we are interested in various generalisations of SR. If preference lists may be incomplete, so that agents are free to designate certain others as unacceptable, then

a stable matching, if one exists, may or may not be complete [15]. However, if a stable matching does exist, then in every stable matching the same set of agents are matched, so that, in particular, all stable matchings have the same size [15]. Irving's algorithm [20] works for this case, and has $O(a)$ complexity, where a is the number of acceptable pairs.

It is also known that the stable matchings for a solvable instance of Stable Roommates with Incomplete lists (SRI) exhibit a semi-lattice structure. This structure can be exploited to solve in polynomial-time a number of problems for instances of SR which admit a stable matching, specifically finding a minimum regret stable matching, finding the stable pairs, and finding all stable matchings in time $O(n^2)$, $O(n^3 \log n)$ and $O(n^3 \log n + n^2 k)$ respectively [12,14,21], where n is the number of agents in the instance and k is the number of stable matchings. It is, however, known that finding an egalitarian stable matching is NP-hard [6].

1.10 Stable Roommates with ties and incomplete lists

We consider Stable Roommates with Ties and Incomplete lists (SRTI). As before there are three types of stability to consider.

For *weak stability*, a blocking pair for a matching M is a pair that does not appear in M such that each member of the pair is either unmatched in M, or prefers the other to their partner in M. Unlike SMTI, for which it is easy to find a weakly stable matching, the problem of finding a weakly stable matching in an instance of SRTI is NP-hard, even if there is at most one tie per list, each tie has length 2, and all lists are complete [46].

For *super-stability*, a blocking pair for a matching M is a pair that does not appear in M such that each member of the pair is either unmatched in M, or prefers the other to their partner in M, or is indifferent between them. There is an algorithm, linear in the input size, which determines whether an instance of SRTI admits a super-stable matching, and if so finds one [26].

Finally, for *strong stability*, a blocking pair for a matching M is a pair that does not appear in M such that one member of the pair is either unmatched in M, or prefers the other to their partner, while the other is unmatched in M or prefers their partner to the first, or is indifferent between them. In Chapter 3 we present an $O(a^2)$ algorithm to determine if

an instance of SRTI admits a strongly stable matching, and if so find one.

1.11 Many-to-many stable matchings

We introduce, and later study, a many-to-many extension of Stable Roommates. Baïou and Balinski [1] have shown that the extension of SM to a many-to-many bipartite problem can be solved in $O(n^2)$ time, where n is the size of the larger set, and that the major structural results of SM carry over. In Chapter 4 we consider the extension of SRI to a many-to-many non-bipartite matching problem, called the *Stable Fixtures* problem, where each agent seeks to be matched with a number of other agents. The concept of stability is extended in the obvious manner. We present an $O(a)$ algorithm to determine whether a stable matching exists in an instance of this problem, and if so the algorithm finds one. Additionally, in Chapter 5, we consider Stable Fixtures with Ties, where again the definition of stability is extended in the obvious manner. We present an $O(a)$ algorithm to determine whether an instance of Stable Fixtures with Ties admits a super-stable matching, and if so the algorithm finds one. We also briefly discuss the problems of finding weakly and strongly stable matchings. Note that Stable Fixtures is a generalisation of Stable Marriage and Hospitals/Residents as well as Stable Roommates, and in view of this, algorithms for finding stable matchings in variants of Stable Fixtures are powerful tools. Recently an extension of Stable Fixtures, called Stable Multiple Activities (SMA), has been studied by Cechlárová and Fleiner [3]. In this extension, there may be more than one way in which two agents can be matched, and the agents can exhibit preferences over these ways. Cechlárová and Fleiner show that SMA can be efficiently reduced to SRI.

Chapter 2

Strong Stability in HRT

2.1 Introduction

In this chapter we present two algorithms for HRT under strong stability. The first is resident-oriented, and the second is hospital-oriented. There is a sense in which strong stability can be viewed as the most appropriate criterion for a practical matching scheme when there is indifference in the preference lists, and that in cases where a strongly stable matching exists, it should be chosen instead of a matching that is merely weakly stable. Consider a weakly stable matching M for an instance of HRT, and suppose that r is unassigned in M, or prefers h to $p_M(r)$, while h is indifferent between r and its worst assignee. Such a pair (r, h) would not constitute a blocking pair for weak stability (assuming $|p_M(h)| = p_h$). However, r might have such an overriding preference for h over $p_M(r)$ that he is prepared to engage in persuasion, even bribery, in the hope that h will reject r' and accept r instead. Hospital h, being indifferent between r and r', may yield to such persuasion, and, of course, a similar situation could arise with the roles reversed. However, the matching cannot be undermined in this way if it is strongly stable. On the other hand, insisting on super-stability seems unnecessarily restrictive, since if (r, h) is a blocking pair for super-stability but not for strong stability, neither r nor h has any real incentive to seek a change. Hence strong stability avoids the possibility of a matching being undermined by persuasion or bribery, and is therefore a desirable property in cases where it can be achieved.

In section 2.2 we present an $O(a^2)$ algorithm for finding a strongly stable matching, if

one exists, given an instance of HRT, thus solving an open problem presented in [27]. The algorithm is resident-oriented in that it finds a strongly stable matching with similar optimality properties to those of the resident-optimal stable matching in HR. This algorithm is a non-trivial extension of the strong stability algorithms for SMT and SMTI due to Irving [22] and Manlove [34] respectively. We also show that the analogue of the Rural Hospitals Theorem for HR holds for HRT under strong stability. In Section 2.3 we establish the complexity of the algorithm to be $O(a^2)$. In Section 2.4 we adapt the resident-oriented algorithm to produce a hospital-oriented counterpart, and in Section 2.5 we show that the latter also has $O(a^2)$ complexity. Finally, Section 2.6 presents our conclusions and a number of other advances, some made in response to the publication of the resident-oriented algorithm [28].

2.2 A resident-oriented algorithm for strong stability in HRT

In this section we describe the resident-oriented algorithm for finding a strongly stable matching, if one exists, given an instance of HRT and prove its correctness. Before doing so, we present some definitions relating to the algorithm.

Recall that a hospital h such that $|\{r : (r, h) \in M\}| = p_h$ is said to be full in the matching M. During the execution of the algorithm residents become *provisionally assigned* to hospitals, and it is possible for a hospital to be provisionally assigned a number of residents that exceeds its quota. At any stage, a hospital is said to be *over-subscribed*, *under-subscribed* or *fully-subscribed* according as it is provisionally assigned a number of residents greater than, less than, or equal to its quota. We describe a hospital as *replete* if at any time during the execution of the algorithm it has been over-subscribed or fully subscribed.

The algorithm proceeds by deleting from the preference lists pairs that cannot be strongly stable. By the *deletion* of a pair (r, h), we mean the removal of r and h from each other's lists, and, if r is provisionally assigned to h, the breaking of this provisional assignment. We say that a resident r is *dominated* on a hospital h's list if h prefers to r at least p_h residents who are provisionally assigned to it.

Recall that the tail of a preference list is the last tie on that list. A resident r who is provisionally assigned to a hospital h is said to be *bound* to h if h is not over-subscribed or

r is not in h's tail (or both). The *provisional assignment graph* G is a bipartite graph with a vertex for each resident and each hospital, with (r, h) forming an edge if resident r is provisionally assigned to hospital h. Note that a number of such graphs may be constructed during the execution of the algorithm, and we refer to a resident (resp. hospital) and the vertex that represents that resident (resp. hospital) interchangeably. A *feasible matching* in a provisional assignment graph is a matching M such that, if r is bound to one or more hospitals, then r is matched with one of these hospitals in M, and subject to this restriction, M has maximum possible cardinality.

A *reduced assignment graph* G_R is formed from a provisional assignment graph as follows. For each resident r, and for each hospital h such that r is bound to h, we delete the edge (r, h) from the graph, and we reduce the quota of h by one; furthermore, we remove *all* other edges incident on r. Each isolated resident vertex is then removed from the graph. Finally, if the quota of any hospital h is reduced to 0, or h becomes an isolated vertex, then h is removed from the graph. For each surviving h we denote by p'_h the revised quota of h. Note that the reduced assignment graph is formed afresh from the current provisional assignment graph in each loop iteration.

Given a set Z of residents in G_R, we define $\mathcal{N}(Z)$, the *neighbourhood* of Z, to be the set of hospital vertices adjacent in G_R to a resident vertex in Z. The *deficiency* of Z is defined by $\delta(Z) = |Z| - \sum_{h \in \mathcal{N}(Z)} p'_h$. 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 there is a unique minimal set with maximum deficiency. This is the *critical set*.

The algorithm, displayed in Figure 2.1, begins by assigning each resident to be free (i.e., not assigned to any hospital). The iterative stage of the algorithm involves each free resident in turn being provisionally assigned to the hospital(s) at the head of his list. If, by gaining a new provisional assignee, a hospital h becomes fully- or over-subscribed then each pair (r, h), such that r is dominated on h's list, is deleted. This continues until every resident is provisionally assigned to one or more hospitals or has an empty list. We then find the reduced assignment graph G_R and the critical set Z of residents. As we will see later, no hospital in $\mathcal{N}(Z)$ can be assigned a resident from those in its tail in any strongly stable matching, so all such pairs are deleted. The iterative step is then reactivated, and this entire process continues until Z is empty, which must happen eventually, since if Z is found to be non-empty, then at least one pair is subsequently deleted from the preference

lists.

Let M be any feasible matching in the final provisional assignment graph G. If M is not strongly stable then no strongly stable matching exists, otherwise the algorithm outputs M.

```
assign each resident to be free;
repeat {
    while some resident r is free and has a non-empty list
        for each hospital h at the head of r's list {
            provisionally assign r to h;
            if h is fully-subscribed or over-subscribed
                for each resident r' dominated on h's list
                    delete the pair (r', h); }
    form the reduced assignment graph;
    find the critical set Z of residents;
    for each hospital h ∈ 𝒩(Z)
        for each resident r in the tail of h's list
            delete the pair (r, h);
} until Z = ∅;
let G be the final provisional assignment graph;
let M be a feasible matching in G;
if M is not strongly stable
    no strongly stable matching exists;
else
    output the strongly stable matching specified by M;
```

Figure 2.1: Algorithm HRT-strong-R

The correctness of Algorithm HRT-strong-R, and an optimality property of any strongly stable matching that it finds, are established below.

Lemma 2.2.1. *No strongly stable pair is ever deleted during an execution of Algorithm HRT-strong-R.*

Proof Suppose that the pair (r, h) is the first strongly stable pair deleted during some execution of the algorithm, and let M' be a strongly stable matching in which r is assigned

to h. There are two cases to consider.

Case 1: Suppose (r, h) is deleted as a result of some other resident, r' say, becoming provisionally assigned to h, so that r is dominated on h's list. Call the set of residents provisionally assigned to h at this point R'. None of the residents in R' can be assigned to a hospital they prefer to h in any strongly stable matching, for otherwise some strongly stable pair must have been deleted before (r, h), as h must be in the head of each of the lists of the residents in R'. In M', at least one of the residents in R', r'' say, cannot be assigned to h, so r'' is either unmatched in M', or prefers h to $p_{M'}(r'')$, or is indifferent between h and $p_{M'}(r'')$. It follows that (r'', h) blocks M', a contradiction.

Case 2: Suppose that (r, h) is deleted because h is provisionally assigned a resident in the critical set Z at some point, and at that point r is in h's tail. We refer to the set of lists at that point as the *current lists*. Let Z' be the set of residents in Z who are assigned in M' to a hospital from the head of their current list, and let H' be the set of hospitals in $\mathcal{N}(Z)$ who are assigned in M' at least one resident from the tail of their current list. Then $h \in H'$, so $H' \neq \emptyset$. Consider $r* \in Z$. Now $r*$ cannot be matched in M' with a hospital that he prefers to any member of the head of his current list, for otherwise some strongly stable pair must have been deleted before (r, h). Hence, any resident r^* in Z who is provisionally assigned to h must be in Z', otherwise (r^*, h) would block M'. Thus $Z' \neq \emptyset$.

We now claim that $\mathcal{N}(Z \setminus Z')$ is not contained in $\mathcal{N}(Z) \setminus H'$. For, suppose that the containment does hold. Then

$$\begin{aligned} |Z \setminus Z'| - \textstyle\sum_{h \in \mathcal{N}(Z \setminus Z')} p_h &\geq |Z \setminus Z'| - \textstyle\sum_{h \in \mathcal{N}(Z) \setminus H'} p_h \\ &= |Z| - \textstyle\sum_{h \in \mathcal{N}(Z)} p_h - (|Z'| - \textstyle\sum_{h \in H'} p_h). \end{aligned}$$

But $|Z'| - \sum_{h \in H'} p_h \leq 0$, because every resident in Z' is matched in M' with a hospital in H'. Hence $Z \setminus Z'$ has deficiency greater than or equal to that of Z, contradicting the fact that Z is the critical set. Thus the claim is established.

Hence there must be a resident $r_1 \in Z \setminus Z'$ and a hospital $h_1 \in H'$ such that r_1 is provisionally assigned to h_1. Since r_1 is either unmatched in M' or prefers h_1 to $p_{M'}(r_1)$

and h_1 is indifferent between r_1 and at least one member of $p_{M'}(h_1)$, the pair (r_1, h_1) blocks M', a contradiction. □

We continue with three auxiliary lemmas.

Lemma 2.2.2. *Every resident who is assigned to a hospital in the final provisional assignment graph G must be assigned in any feasible matching M.*

Proof The result is true by definition for any bound resident. Consider the other residents assigned in G. Any x of them must be collectively adjacent in G_R to hospitals with at least x posts, otherwise one of them is in the critical set Z, and hence $Z \neq \emptyset$. But, by a simple extension of Philip Hall's Theorem, this means that they are all matched in any maximum cardinality matching in G_R, and hence they must be matched in any feasible matching M. □

Lemma 2.2.3. *Let M be a feasible matching in the final provisional assignment graph G. If (a) some non-replete hospital h has fewer assignees in M than provisional assignees in G, or (b) some replete hospital h is not full in M, then no strongly stable matching exists.*

Proof Suppose that M' is a strongly stable matching for the instance. Every resident provisionally assigned to a hospital in the final assignment graph G must be assigned to a hospital in M (by Lemma 2.2.2), and any resident not provisionally assigned in G must have an empty list and hence no strongly stable partners (by Lemma 2.2.1). It follows that $|M'| \leq |M|$.

Suppose that condition (a) is satisfied. Then some non-replete hospital h' satisfies $|p_M(h')| < d_G(h')$, where $d_G(h')$ is the degree of vertex h' in G, i.e., the number of residents provisionally assigned to h'. As h' is non-replete, it follows that $d_G(h') < p_{h'}$. Now $|p_M(h)| \leq \min(d_G(h), p_h)$ for all $h \in H$. Hence

$$|M| = \sum_{h \in H} |p_M(h)| < \sum_{h \in H} \min(d_G(h), p_h). \tag{2.1}$$

Now suppose that $|p_{M'}(h)| \geq \min(d_G(h), p_h)$ for all $h \in H$. Then $|M'| > |M|$ by 2.1, a contradiction. Hence $|p_M(h'')| < \min(d_G(h''), p''_h)$ for some $h'' \in H$. Hence h'' is undersubscribed in M', and some resident r' is provisionally assigned to h'' in G but not assigned

to h'' in M'. By Lemma 2.2.1, r' is not assigned to a hospital in M' that he prefers to h''. Hence (r', h'') blocks M', a contradiction.

Now suppose that condition (b) is satisfied. Let H_1 and H_2 be the set of replete and non-replete hospitals respectively. Then some $h' \in H_1$ satisfies $|p_M(h')| < p_{h'}$. Condition (a) cannot be satisfied, for otherwise the first part of the proof shows that M' does not exist. Hence $|p_M(h)| = d_G(h) < p_h$ for all $h \in H_2$. Now $p_M(h) \leq p_h$ for all $h \in H_1$. Hence

$$|M| = \sum_{h \in H_1} |p_M(h)| + \sum_{h \in H_2} |p_M(h)| < \sum_{h \in H_1} p_h + \sum_{h \in H_2} d_G(h). \tag{2.2}$$

Now suppose that $|p_{M'}(h) \geq p_h$ for all $h \in H_1$ and $|p_{M'}(h) \geq d_G(h)$ for all $h \in H_2$. Then $|M'| > |M|$ by 2.2, a contradiction. Hence either (i) $|p_{M'}(h'') < p''_h$ for some $h'' \in H_1$ or (ii) $|p_{M'}(h'')| < d_G(h'')$ for some $h'' \in H_2$. In Case (ii) we reach a similar contradiction to that arrived at for condition (a). In Case (i), h'' is under-subscribed in M'. As h'' is replete, there exists some resident r' who was provisionally assigned to h'' during the execution of the algorithm, but is not assigned to h'' in M'. By Lemma 2.2.1, r' is not assigned to a hospital in M' that he prefers to h''. Hence (r', h'') blocks M', a contradiction. □

Lemma 2.2.4. *Suppose that, in the final assignment graph G, a resident is bound to two different hospitals. Then no strongly stable matching exists.*

Proof Suppose that a strongly stable matching exists for the instance. Let M be a feasible matching in the final provisional assignment graph G. Denote by H_1 the set of over-subscribed hospitals in G, let $H_2 = H \setminus H_1$ and let $d_G(h)$ denote the degree of the vertex h in G - i.e., the number of residents provisionally assigned to h. Denote by R_1 the set of residents bound to one or more hospitals in G, and by R_2 the other residents assigned to one or more hospitals in G. Note that for each $h \in H_2$, any resident assigned to h in G is bound to h, and hence is in R_1.

By Lemma 2.2.2, we have

$$|M| = |R_1| + |R_2|. \tag{2.3}$$

Also, by Lemma 2.2.3, we have

$$|M| = \sum_{h \in H_1} p_h + \sum_{h \in H_2} d_G(h). \tag{2.4}$$

If some resident is bound to more than one hospital then, by considering how quotas are reduced when the residents of R_1 are removed in deriving G_R from G, it follows that

$$\sum_{h \in H_1} (p_h - p'_h) + \sum_{h \in H_2} d_G(h) > |R_1|. \tag{2.5}$$

Combining 2.3, 2.4 and 2.5 gives

$$\sum_{h \in H_1} p'_h < |R_2|.$$

Since no member of H_2 belongs to G_R, the residents in R_2 are collectively adjacent only to hospitals in H_1, and so the foregoing inequality suffices to establish that the critical set is non-empty, a contradiction. □

The next lemma, along with Lemma 2.2.1, is key to showing that Algorithm HRT-strong-R is correct.

Lemma 2.2.5. *Let M be a feasible matching in the final provisional assignment graph. If M is not strongly stable then there is no strongly stable matching for the instance.*

Proof Suppose M is not strongly stable, and let (r, h) be a blocking pair for M. Suppose r prefers h to $p_M(r)$, or r is unassigned in M. Then (r, h) has been deleted, which can only happen if h is replete. To see this, suppose that h is not replete, but (r, h) was deleted because h was a neighbour of some resident $r' \in Z$ at a point when r was in h's tail. Suppose that the residents in $Z' \subseteq Z$ are provisionally assigned to h' in G_R. Then $0 < |Z'| \leq p'_{h'}$. Let $Z^* = Z \setminus Z'$. Then $\mathcal{N}(Z^*) \subseteq \mathcal{N}(Z) \setminus \{h'\}$ so that

$$\sum_{h \in \mathcal{N}(Z^*)} p'_h \leq \sum_{h \in \mathcal{N}(Z)} (p'_h - p'_{h'}).$$

Hence

$$\begin{aligned}
\delta(Z^*) &= |Z^*| - \textstyle\sum_{h\in\mathcal{N}(Z^*)} p'_h \\
&= |Z| - |Z'| - \textstyle\sum_{h\in\mathcal{N}(Z^*)} p'_h \\
&\geq |Z| - |Z'| - (\textstyle\sum_{h\in\mathcal{N}(Z)}(p'_h - p'_{h'})) \\
&= |Z| + p'_{h'} - |Z'| - \textstyle\sum_{h\in\mathcal{N}(Z)} p'_h \\
&\geq |Z| - \textstyle\sum_{h\in\mathcal{N}(Z)} p'_h \\
&= \delta(Z).
\end{aligned}$$

If $\delta(Z^*) > \delta(Z)$ then Z^* contradicts the fact that Z is maximally deficient. Hence $\delta(Z^*) = \delta(Z)$. But $Z^* \subset Z$, contradicting the minimality of Z. Thus h is replete. If h is full in M then h prefers all its assignees to r, since r is a strict successor of any undeleted entries in h's list, contradicting the fact that (r, h) is a blocking pair for M. If h is not full in M then h is a replete hospital which is not full in M, so by Lemma 2.2.3, no strongly stable matching exists for the instance, and we are done. We now consider the case where r is indifferent between h and $p_M(r)$.

Suppose h is not full in M. If h is replete then, by Lemma 2.2.3, no strongly stable matching exists for the instance and we are done. If h is not replete then r must be bound to h, and since r is not assigned to h in M, by the definition of a feasible matching r must be bound to $p_M(r)$. But then r is bound to two hospitals, so by Lemma 2.2.4 no strongly stable matching exists for the instance.

Now suppose h is full in M. For (r, h) to block M, h must prefer r to at least one of its assignees in M. But then r is bound to h, and since r is not assigned to h in M, r must be bound to $p_M(r)$. But then again r is bound to two hospitals, so by Lemma 2.2.4 no strongly stable matching exists for the instance. □

Lemma 2.2.5 proves the correctness of Algorithm HRT-strong-R. Further, Lemma 2.2.1 shows that there is an optimality property for each assigned resident in any strongly stable matching output by the algorithm. To be precise, we have proved:

Theorem 2.2.6. *For a given instance of HRT, Algorithm HRT-strong-R 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 assigned resident is assigned as*

favourable a hospital as in any strongly stable matching, and any unassigned resident is unassigned in every strongly stable matching.

For obvious reasons, we call any matching found by the above algorithm *resident-optimal*.

Now we show that the Rural Hospitals Theorem holds for HRT under strong stability. For the following lemma and theorem we assume that we have an HRT instance that admits a strongly stable matching.

Lemma 2.2.7. *For a given HRT instance, let M be the matching obtained by Algorithm HRT-strong-R and let M' be any strongly stable matching. If a hospital h is not full in M' then every resident assigned to h in M is also assigned to h in M'.*

Proof Suppose r is assigned to h in M, but not in M'. Then (r, h) blocks M' since h is undersubscribed in M' and r cannot prefer any of his strongly stable partners to h. □

Theorem 2.2.8. *For a given HRT instance I,*

1. *each hospital is assigned the same number of residents in every strongly stable matching;*
2. *the same residents are matched in every strongly stable matching;*
3. *any hospital that is undersubscribed in some strongly stable matching is assigned the same set of residents in every strongly stable matching.*

Proof Let M be the strongly stable matching obtained by Algorithm HRT-strong-R, and let M' be any strongly stable matching such that $M' \neq M$.

1. We first observe that any resident r who is unmatched in M cannot be matched in M', since r must have an empty list (hence (r, h) has been deleted for every hospital h that r finds acceptable, and by Lemma 2.2.1 no strongly stable pair is deleted during the execution of Algorithm HRT-strong-R). It follows that $|M'| \leq |M|$. By Lemma 2.2.7, any hospital that is full in M is also full in M', while any hospital that is not full in M fills at least as many posts in M' as in M. It follows that $|M'| \geq |M|$, and so, combining this with the earlier inequality, $|M| = |M'|$. This equality and the conclusions drawn earlier from Lemma 2.2.7 imply that every hospital is assigned the same number of residents in M and M'.

2. As has already been observed, $|M| = |M'|$, and no resident who is unmatched in M can be matched in M', so the same set of residents are matched in M and M'.

3. As has already been observed, $|M'| \leq |M|$, and the result follows for M' by Lemma 2.2.7 and the first part of the proof.

Since M' is an arbitrary strongly stable matching, these results follow for every strongly stable matching. □

Example 2.2.1. An example instance is displayed in Figure 2.2. The residents are labeled r_i $(1 \leq i \leq 6)$ and the hospitals are labeled h_i $(1 \leq i \leq 3)$. The entry for hospital h_i takes the form $h_i : (p_{h_i})\, P_{h_i}$, where P_{h_i} is h_i's preference list. The entry for a resident is similar, but without the quota element.

r_1:	$(h_2\ h_3)\ h_1\ h_4$	r_1:	$h_1\ h_4$
r_2:	$h_2\ h_1$	r_2:	$h_2\ h_1$
r_3:	$h_3\ h_2\ h_1$	r_3:	$h_2\ h_1$
r_4:	$h_2\ (h_1\ h_3)$	r_4:	$(h_1\ h_3)$
r_5:	$h_2\ (h_1\ h_3)$	r_5:	$(h_1\ h_3)$
r_6:	h_3	r_6:	
h_1:(2)	$r_2\ (r_1\ r_3)\ (r_4\ r_5)$	h_1:(2)	$r_2\ (r_1\ r_3)\ (r_4\ r_5)$
h_2:(2)	$r_3\ r_2\ (r_1\ r_4\ r_5)$	h_2:(2)	$r_3\ r_2$
h_3:(1)	$(r_4\ r_5)\ (r_1\ r_3)\ r_6$	h_3:(1)	$(r_4\ r_5)$
h_4:(1)	r_1	h_4:(1)	r_1
	Initial preference lists		Lists after first loop iteration

Figure 2.2: The preference lists for an example HRT instance

We assume the residents become assigned to the hospitals at the head of their lists in subscript order. The while loop of Algorithm HRT-strong-R terminates with every resident except r_6 provisionally assigned to every hospital in the first tie on their preference list. Resident r_6 has an empty list, because (r_6, h_3) was deleted as a result of h_3 receiving a proposal from r_1, causing r_6 to be dominated on the list of h_3. Only one edge is removed from the provisional assignment graph to form the reduced assignment graph, as only resident r_2 is bound to a hospital, namely

h_2. The isolated vertices are then removed from the graph, leaving residents r_1, r_3, r_4 and r_5, and hospitals h_2 and h_3. It can then be shown that every resident in the reduced assignment graph is in the critical set, and the neighbourhood of the critical set is $\{h_2, h_3\}$. The lists after the relevant deletions have been made are displayed in Figure 2.2. By following the same process, it can be shown that a second iteration of the main loop of Algorithm HRT-strong-R terminates with an empty critical set, and there are two feasible matchings, $(r_1, h_1), (r_2, h_2), (r_3, h_2), (r_4, h_3), (r_5, h_1)$ and $(r_1, h_1), (r_2, h_2), (r_3, h_2), (r_4, h_1), (r_5, h_3)$, one of which is output by the algorithm, and both are strongly stable.

2.3 Implementation and analysis of Algorithm HRT-strong-R

For the implementation and analysis of Algorithm HRT-strong-R, we require to describe the efficient construction of maximum cardinality matchings and critical sets in a context somewhat more general than that of simple bipartite graphs.

Consider a *capacitated* bipartite graph $G = (V, E)$, with bipartition $V = R \cup H$, in which each vertex $h \in H$ has a positive integer *capacity* c_h. In this context, a *matching* is a subset M of E such that $|\{h : \{r, h\} \in M\}| \leq 1$ for all $r \in R$, and $|\{r : \{r, h\} \in M\}| \leq c_h$ for all $h \in H$. For any vertex x, a vertex joined to x by an edge of M is called a *mate* of x. A vertex $r \in R$ with no mate, or a vertex $h \in H$ with fewer than c_h mates, is said to be *exposed*. An *alternating path* in G relative to M is any simple path in which edges are alternately in, and not in, M. An *augmenting* path is an alternating path of odd length both of whose endpoints are exposed. It is immediate that an augmenting path has one endpoint in R and the other in H.

The following lemmas may be established by straightforward extension of the corresponding results for one-to-one bipartite matching.

Lemma 2.3.1. *Let P be the set of edges on an augmenting path relative to a matching M in a capacitated bipartite graph G. Then $M' = M \oplus P$ is a matching of cardinality $|M|+1$ in G.*

Lemma 2.3.2. *A matching M in a capacitated bipartite graph has maximum cardinality if and only if there is no augmenting path relative to M in G.*

The process of replacing M by $M' = M \oplus P$ is called *augmenting* M along path P.

With these lemmas, we can extend to the context of capacitated bipartite graphs the classical augmenting path algorithm for a maximum cardinality matching. The algorithm starts with an arbitrary matching – say the empty matching – and repeatedly augments the matching until there is no augmenting path. The search for an augmenting path relative to M is organised as a restricted breadth-first search in which only edges of M are followed from vertices in H and only edges not in M are followed from vertices in R, to ensure alternation. The number of iterations is $O(\min(|R|, \sum c_h))$, and each search can be completed in $O(|E|)$ time, since there are no isolated vertices. During the breadth-first search, we record the parent in the BFS spanning tree of each vertex. This enables us to accomplish the augmentation in $O(|E|)$ time, observing that, for each vertex $h \in H$, the set of mates can be updated in constant time by representing the set as, say, a doubly linked list, and storing a pointer into this list from any child node in the BFS spanning tree. Hence, overall, the augmenting path algorithm in a capacitated bipartite graph can be implemented to run in $O((\min(|R|, \sum c_h))|E|)$ time.

Now that we have ascertained that we can efficiently find a maximum cardinality matching in the reduced assignment graph, the following lemma points the way to finding the critical set.

Lemma 2.3.3. *Given a maximum cardinality matching M in the capacitated bipartite graph G_R, the critical set Z consists of the set U of unmatched residents together with the set U' of residents reachable from a vertex in U via an alternating path.*

Proof Let $C = U \cup U'$. It is immediate that $\delta(C) = \delta(G)(= |U|)$, for if $\mathcal{N}(C)$ were such that

$$\sum_{h \in \mathcal{N}(C)} p'_h > |U'|$$

there would be an augmenting path relative to M, contradicting the maximality of M.

Further, the critical set Z must contain every resident who is unmatched in some maximum cardinality matching in G. For if M' is an arbitrary such matching (of size $|R| - \delta(G)$), and if $r \in R \setminus Z$ is not matched in M', then Z must contain at least $|Z| - \delta(G) + 1$ matched residents. To see this consider that there must be $\delta(G)$ unmatched residents, with at most

$\delta(G) - 1$ of these residents contained in Z. Hence Z contains at most $|Z| - \delta(G) + 1$ residents. It follows that

$$\sum_{h \in \mathcal{N}(Z)} p'_h \geq |Z| - \delta(G) + 1$$

or

$$|Z| - \sum_{h \in \mathcal{N}(Z)} p'_h \leq \delta(G) - 1$$

contradicting the required deficiency of Z.

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

During each iteration of the repeat-until loop of Algorithm HRT-strong-R we need to form the reduced assignment graph, which takes $O(a)$ time, then search for a maximum cardinality matching in the bipartite graph G_R. This allows us to use Lemma 2.3.3 to find the critical set. The key to the analysis of Algorithm HRT-strong-R, as with Algorithm STRONG in [22], is bounding the total amount of work done in finding the maximum cardinality matchings.

It is clear that work done other than in finding the maximum cardinality matchings and critical sets is bounded by a constant times the number of deleted pairs, and so is $O(a)$[1].

Suppose that Algorithm HRT-strong-R finds a maximum cardinality matching M_i in the reduced assignment graph G_R at the ith iteration. Suppose also that, during the ith iteration, x_i pairs are deleted because they involve residents in the critical set Z, or residents tied with them in the list of a hospital in $\mathcal{N}(Z)$. Suppose further that in the $(i+1)$th iteration, y_i pairs are deleted before the reduced assignment graph is formed. Note that any edge in G_R at the ith iteration which is not one of these $x_i + y_i$ deleted pairs must be in G_R at the $(i+1)$th iteration, since a resident can only become bound to a hospital when he becomes provisionally assigned to it. In particular at least $|M_i| - x_i - y_i$ pairs of M_i remain in G_R at the $(i+1)$th iteration. Hence, in that iteration, we can start

[1] See, for example, Section 4.4.

from these pairs and find a maximum cardinality matching in $O(min(na, (x_i + y_i + z_i)a))$ time, where n is the number of residents and z_i is the number of edges in G_R at the $(i+1)$th iteration which were not in G_R at the ith iteration.

Let s denote the number of iterations carried out, let $S = \{1, 2, \ldots, s\}$, and let $S' = S \backslash \{s\}$. Let $T \subseteq S'$ denote those indices i such that $\min(na, (x_i + y_i + z_i)a) = na$, and let $t = |T|$. Then the algorithm has time complexity $O(\min(n, p)a + tna + a \sum_{i \in S' \backslash T}(x_i + y_i + z_i))$, where p is the total number of posts, and the first term is for the first iteration. But $\sum_{i \in S'}(x_i + y_i) \leq a$ and $\sum_{i \in S'} z_i \leq a$ (since these summations are bounded by the total number of deletions and provisional assignments, respectively), and since $x_i + y_i + z_i \geq n$ for each $i \in T$, it follows that

$$tn + \sum_{i \in S' \backslash T} (x_i + y_i + z_i) \leq \sum_{i \in S'} (x_i + y_i + z_i) \leq 2a$$

Thus

$$\sum_{i \in S' \backslash T} (x_i + y_i + z_i) \leq 2a - tn.$$

After the end of the final iteration a feasible matching is constructed by taking the final maximum cardinality matching and combining it with the bound resident-hospital pairs. This operation is clearly bounded by the number of bound pairs, hence is $O(a)$. It follows that the overall complexity of Algorithm HRT-strong-R is $O(min(n, p)a + tna + a(2a - tn)) = O(a^2)$.

Note that while there is at least one algorithm for finding maximum cardinality matchings in the context of capacitated bipartite graphs with better time complexity than the one we use here (see e.g. [7]), it is not clear how we can use this algorithm to give an improvement in the running time of Algorithm HRT-strong-R.

2.4 A hospital-oriented algorithm for strong stability in HRT

In this section we describe a second algorithm for finding a strongly stable matching, if one exists, in an instance of HRT, and prove its correctness. Much of the terminology is

the same as that used for Algorithm HRT-strong-R, but there are a few new or adapted definitions.

For a hospital h, $t(h)$ is the first tie on h's list that contains a resident who is not provisionally assigned to h. For a resident r, we denote the tail of r's list by $l(r)$. Deleting the tail of a resident r's list means deleting the pairs (r, h) for each $h \in l(r)$. We say that a hospital h is *dominated* on a resident r's list if r prefers some hospital to which it is provisionally assigned to h. We say that h has a *short list* if there are fewer than p_h residents on h's preference list. A resident r who is provisionally assigned to a hospital h is said to be *bound* to h if h is not over-subscribed, or if h prefers r to at least one other resident provisionally assigned to it.

The *provisional assignment graph* G has a vertex for each resident and each hospital, with (r, h) forming an edge if resident r is provisionally assigned to hospital h. Note that a number of such graphs may be constructed during the execution of the algorithm, and we refer to a resident (resp. hospital) and the vertex that represents that resident (resp. hospital) interchangeably. Note that one consequence of the algorithm is that a resident cannot be bound to more than one hospital, so while the definition of the provisional assignment graph is exactly as for the resident-oriented algorithm, the definition of a feasible matching changes slightly. Here, a *feasible matching* in a provisional assignment graph is a matching M such that, if r is bound to a hospital then r is matched in M with that hospital, and subject to this restriction, M has maximum possible cardinality.

A *reduced assignment graph* G_R is formed from a provisional assignment graph as follows. For each resident r who is bound to a hospital h, we delete the edge (r, h) from G_R, and we reduce the quota of h by one; furthermore, we remove *all* other edges incident on r. Further, for each resident adjacent to only one hospital h in G_R, we remove the edge (r, h) from G_R, and we reduce the quota of h by one. If at any point a hospital has its quota reduced to 0, and there are still edges incident on the hospital vertex, then these edges are removed. This may cause some resident to be adjacent to only one hospital in G_R, and if so he is treated as above. This process continues until every hospital with quota 0 has no edges incident and no resident satisfies the foregoing condition. Each isolated vertex is then removed from G_R. For each surviving hospital h we denote by p'_h the revised quota of h.

Given a set Z of hospitals in G_R, the *neighbourhood* of Z, denoted $\mathcal{N}(Z)$, is the set of resident vertices adjacent in G_R to a hospital vertex in Z. The *deficiency* of Z is defined by $\delta(Z) = \sum_{h \in Z} p'_h - |\mathcal{N}(Z)|$. 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 there is a unique minimal set with maximum deficiency. We call this set the *deficient* set. Let Z be the deficient set. For each hospital $h \in Z$, and for each resident r provisionally assigned to h but not provisionally assigned to any other hospital in Z, we delete the edge (r, h) from G_R, thereby removing r from $\mathcal{N}(Z)$, and we reduce p'_h by one. Note that, by the definition of the deficient set, this cannot leave any hospital in the deficient set with a revised quota of 0 or less. The resulting set $C = \mathcal{N}(Z)$ of residents is the *critical set*. The rationale for the final set of edge deletions is simply that (r, h) may be a strongly stable pair (if (r, h) is in a strongly stable matching, then it can be seen that r cannot be involved in a blocking pair with any member of Z - an illustration of this is included in the example at the end of this section).

The algorithm, displayed in Figure 2.3, begins by assigning each resident to be free (i.e., not assigned to any hospital). The iterative stage of the algorithm involves each under-subscribed hospital h in turn being provisionally assigned the resident(s) in $t(h)$. Then each pair (r, h) such that h is dominated on r's list is deleted. If, when becoming provisionally assigned to a hospital, r becomes bound to two hospitals then the tail of r is deleted. If these deletions cause some resident to become bound to a second hospital then the tail of that resident is deleted. This process continues until every hospital h is provisionally assigned at least p_h residents, or has a short list. We then find the reduced assignment graph G_R and the critical set C of residents. As we will see later, no resident in C can be assigned to a hospital from those in its tail in any strongly stable matching, so all such pairs are deleted. Again, if these deletions cause some resident to become bound to two hospitals then the tail of that resident is deleted. The iterative step is then reactivated, and this entire process continues until C is empty, which must happen eventually, since if C is found to be non-empty, then at least one pair is subsequently deleted from the preference lists. On termination of the main loop, the algorithm finds a feasible matching in the final provisional assignment graph G. If this matching is not strongly stable then no strongly stable matching exists, otherwise the algorithm outputs the matching.

The correctness of Algorithm HRT-strong-H is established below.

assign each resident to be free;
repeat {
  while some hospital h is undersubscribed
  and there is some resident on h's list who is not assigned to h {
    for each resident $r \in t(h)$ {
      provisionally assign r to h;
      for each hospital h' dominated on r's list
        delete the pair (r, h');}
    while some resident r is bound to more than one hospital
      delete (r, h') for each $h' \in l(r)$; }
  form the reduced assignment graph;
  find the critical set C of residents;
  for each resident $r \in C$ {
    for each hospital $h' \in l(r)$
      delete the pair (r, h');
    while some resident r is bound to more than one hospital
      delete (r, h') for each $h' \in l(r)$; }
} **until** $C = \emptyset$;
let G be the final provisional assignment graph;
let M be a feasible matching in G;
if M is strongly stable
  output M;
else
  no strongly stable matching exists;

Figure 2.3: Algorithm HRT-strong-H

Lemma 2.4.1. *No strongly stable pair is ever deleted during an execution of Algorithm HRT-strong-H.*

Proof Suppose that the pair (r, h) is the first strongly stable pair deleted during some execution of the algorithm, and let M be a strongly stable matching in which r is assigned to h. There are three cases to consider.

Case 1: Suppose (r, h) is deleted as a result of r being bound to at least two hospitals, at a time when $h \in l(r)$ (note there are two points in the algorithm where this can happen).

Let $h' \neq h$ be a hospital to which r is bound, and let G denote the provisional assignment graph immediately prior to the deletion of (r, h). As r is bound to h', either (i) h' is not over-subscribed, or (ii) h' prefers r to at least one resident r' provisionally assigned to h' in G. In Case (i), just before (r, h) is deleted, h' is provisionally assigned the first k residents on its list, for some $k \leq p_{h'}$. No strongly stable pair was deleted before (r, h), and $(r, h') \notin M$. Hence, in M, either h' is under-subscribed or h' prefers r to its worst assignee. Hence (r, h') blocks M, a contradiction. In Case (ii), since no strongly stable pair was deleted before (r, h), it follows that, in M, either h' is under-subscribed or prefers r' to its worst assignee, or is indifferent between them. Hence, in M, either h' is under-subscribed or prefers r to its worst assignee, so that (r, h') blocks M, a contradiction.

Case 2: Suppose (r, h) is deleted as a result of r becoming provisionally assigned to some other hospital, h' say, so that h is dominated on r's list. Since (r, h) was the first strongly stable pair to be deleted, h' cannot be matched in any strongly stable matching with a set of $p_{h'}$ residents all of whom he prefers to r, for otherwise some strongly stable pair must have been deleted for r to become provisionally assigned to h'. It follows that h' prefers r to at least one of its assigned residents in M, or is indifferent between them. Since r prefers h' to h, (r, h') blocks M, a contradiction.

Case 3: Suppose that (r, h) is deleted because r is in the critical set C at some point, and at that point $h \in l(r)$. We refer to the set of lists immediately before the critical set deletions as the *current lists*. Since $r \in C$, r is provisionally assigned to at least one hospital $h_1 (\neq h)$ such that r is indifferent between h and h_1. Since r is indifferent between h and h_1, h_1 cannot be matched in M with a resident r' such that h_1 prefers r to r'. Now every strongly stable pair is contained in the current lists, so even if h_1 is matched with every resident r^* such that (r^*, h) was removed when forming the reduced assignment graph or when finding the critical set, it follows that h_1 must be matched in M with at least $p'_{h_1} (\geq 1)$ residents from C. Each of these p'_{h_1} residents must be provisionally assigned to at least one other hospital in the deficient set Z. Suppose r is the only resident from C who is provisionally assigned to h_1 in the current lists. Then, since $(r, h) \in M$, it follows that h_1 must be under-subscribed in M, or prefer r to at least one resident in $p_M(h_1)$. But then (r, h_1) blocks M, a contradiction. Hence there must be a resident, r_1 say, such that $r_1 \in C$, $r_1 \neq r$, and r_1 is provisionally assigned to h_1 in the current lists. Again, since $r_1 \in C$, r_1 is provisionally assigned to at least one hospital $h_2 \in Z$ $(h_2 \neq h_1)$ such that r is

indifferent between h_2 and h_1. Then, by a similar argument to above, h_2 must be matched in M with at least $p'_{h_2}(\geq 1)$ residents from C, each of whom must be provisionally assigned to at least one other hospital in Z.

Now let H' be the set of hospitals $h' \in Z$ such that h' is reachable from h via a path in which each edge has a hospital from Z as one endpoint. Clearly $h \in H'$, so $H' \neq \emptyset$. Suppose there is a hospital $h' \in H'$ such that h' is matched in M with fewer than $p'_{h'}$ residents from C. Without loss of generality we can assume, from the above argument, that there is a hospital $h'' \in Z$ such that h' and h'' are both provisionally assigned some resident $r' \in C$, and h'' is matched in M with $p'_{h'}$ residents from C, including r', as well as every resident r^* such that (r^*, h'') was removed when forming the reduced assignment graph, or when finding the critical set. But, by the above argument h' must be under-subscribed in M, or prefer r' to at least one resident assigned to h' in M, and so (r', h') blocks M, a contradiction. It follows that every hospital in $h' \in H'$ must be matched in M with $p'_{h'}$ residents from C.

Consider $Z \setminus H'$. By the above argument $|\mathcal{N}(H')| \geq \sum_{h? \in H?} p_{h?}?$, and by the construction of H', no resident in $\mathcal{N}(H')$ is in $\mathcal{N}(Z \setminus H')$. It follows that $Z \setminus H'$ must have deficiency at least $\delta(Z)$, a contradiction. Hence there must exist a hospital $h' \in Z$ such that h' is matched in M with fewer than $p'_{h'}$ residents from C, while some resident r' who is provisionally assigned to h' in the current lists is matched in M with a hospital h'' such that r' is indifferent between h' and h'', But, by the above argument h' is either under-subscribed in M, or matched in M with a resident r'' such that h' prefers r' to r''. But then (r', h') blocks M, a contradiction. The result follows. □

Lemma 2.4.2. *Let M be a feasible matching. Then every hospital is either full in M, or matched with every resident on its list.*

Proof By the definition of a feasible matching, a hospital h is assigned every resident who is bound to h. We consider how to assign the remaining residents in a feasible matching M'. For each pair (r, h) such that r is not bound to h but (r, h) is was removed from the provisional assignment graph when the reduced assignment graph G_R was formed, we add (r, h) to M'. It can be verified that every other edge removed from the provisional assignment graph to form G_R has one edge incident on either a hospital which is full in M', or a resident who is assigned in M'. Further, any hospital which is now isolated must

be matched in M' with every resident on its list. Finally, since there is no deficient set in G_R, any set of non-isolated hospitals with x posts must be collectively adjacent in G_R to at least x residents, otherwise one of them is in the deficient set Z, and hence $Z \neq \emptyset$. But, by a simple extension of Philip Hall's Theorem, this means that all these posts are matched in any maximum cardinality matching in G_R, and hence they must be filled in any feasible matching M. The result follows by the definition of a feasible matching. □

Lemma 2.4.3. *Let M be a feasible matching in the final provisional assignment graph G. If M is not strongly stable then no strongly stable matching exists.*

Proof Suppose that M is not strongly stable, but that M' is a strongly stable matching for the instance. Let (r, h) be a blocking pair for M. Suppose r is matched in M, and is indifferent between h and $h' = p_M(r)$. Then (r, h) cannot have been deleted, as (r, h') has not been deleted, and every deleted element on r's list is a strict successor of any undeleted elements. Now, for (r, h) to block M, h must either be under-subscribed in M, or prefer r to at least one resident assigned to it in M. In the former case h must be matched in M with every resident on its list. But then (r, h) must have been deleted, implying that r is either unmatched in M, or prefers $p_M(r)$ to h, a contradiction. In the latter case r is bound to h. But then, by the definition of a feasible matching, r must be assigned to h in M, a contradiction. It follows that either r is unmatched in M, or r prefers h to $h' = p_M(r)$. In the latter case h is either under-subscribed in M or prefers r to at least one of the residents assigned to it in M, or is indifferent between them. If h is under-subscribed we get a contradiction from the same argument as above. Otherwise r must have been provisionally assigned to h at some point. But then (r, h') must have been deleted, and so r cannot be matched with h' in M. Thus r must be unmatched in M.

We now show that $|M| = |M'|$. By Lemma 2.4.2, every hospital is either full in M, or matched with every resident on its list. Since, by Lemma 2.4.1, no strongly stable pair is ever deleted by the algorithm, it follows that M' cannot be larger than M. Now suppose that $|M| > |M'|$. Then there is a resident r' such that r' is matched in M (to h'' say), but unmatched in M'. If h'' is not full in M' then (r', h'') blocks M', a contradiction. Hence h'' is full in M'. For each resident r'' such that $(r'', h'') \in M'$, h'' must prefer r'' to r', or (r', h'') blocks M'. But r' is provisionally assigned to h'' at the termination of Algorithm HRT-strong-H. It follows that some strongly stable pair must have been deleted

by Algorithm HRT-strong-H, a contradiction of Lemma 2.4.1. Hence $|M| = |M'|$.

Suppose (r, h) has been deleted. Then r must have been provisionally assigned to a hospital at some point during the execution of the algorithm. On the other hand, if (r, h) has not been deleted, since h cannot be undersubscribed in M, by the argument above, r must be provisionally assigned to h in the final provisional assignment graph. Additionally every resident r' who is matched in M is provisionally assigned to $p_M(r')$. Hence there are $|M| + 1 = |M'| + 1$ residents who were provisionally assigned at some point during the execution of the algorithm, since r is unmatched in M. In particular there is a resident r' who was provisionally assigned at some point during the execution of the algorithm, say to h'', but is unmatched in M'. Then either h'' is not full in M', or, by Lemma 2.4.1, there is a resident r'' such that $(r'', h'') \in M'$, and h'' prefers r' to r'', or is indifferent between them. In either case (r', h'') blocks M', a contradiction. The result follows. □

Lemmas 2.4.1 and 2.4.3 prove the correctness of Algorithm HRT-strong-H, and Lemma 2.4.1, along with Theorem 2.2.8, demonstrates an optimality property for each hospital in any strongly stable matching output by the algorithm. In particular, we have proved:

Theorem 2.4.4. *For a given instance of HRT, Algorithm HRT-strong-H determines whether a strongly stable matching exists. If such a matching exists, the algorithm outputs one in which every hospital has at least as favourable a set of assignees as it can have in any strongly stable matching.*

For obvious reasons, we call any matching found by the above algorithm *hospital-optimal*.

Example 2.4.1. An example instance is displayed in Figure 2.4. The format is the same as the example in Section 2.2, though the example is somewhat more complex.

We assume the hospitals propose in subscript order. During the execution of the main while loop in Algorithm HRT-strong-H, the pair (r_{10}, h_8) is deleted because h_8 becomes dominated on r_{10}'s list. During the execution of the first inner while loop the pairs (r_7, h_2) and (r_7, h_3) are deleted because r_7 becomes bound to both h_2 and h_3. On termination of the main while loop, hospitals h_1 and h_5 are provisionally assigned the residents in the first two ties on their lists, hospitals h_2 and h_3 are assigned the residents in the second tie on their lists (since (r_7, h_2) and (r_7, h_3) were both deleted), and the remaining hospitals are provisionally assigned the residents in the first tie on their lists. When the reduced assignment graph is formed from the provisional assignment graph the edges $\{r_{10}, h_1\}$ and $\{r_{11}, h_5\}$ are removed because r_{10} and r_{11} are bound to h_1 and h_5 respectively, and so the quotas of h_1 and h_5 are reduced by one. Additionally, the removal of $\{r_{10}, h_1\}$ requires the

removal of the edge $\{r_{10}, h_9\}$, and the reduction of h_9's quota by one. Finally, the edge $\{r_{12}, h_9\}$ is also removed, because r_{12} is provisionally assigned only to h_9, and this causes h_9 to have a quota of 0, so the edge $\{r_4, h_9\}$ is also removed. The isolated vertices (r_{10}, r_{11}, r_{12} and h_9) are then removed from the graph. It is then possible to verify that the deficient set is $\{h_1, h_2, h_3, h_4, h_5, h_6\}$. However, r_9 is provisionally assigned to only one hospital in the deficient set, so we remove the edge $\{r_9, h_4\}$ (note that the pair $\{r_9, h_4\}$ is in both the strongly stable matchings for the instance, listed below), and thus the critical set is $\{r_1, r_2, r_3, r_4, r_5, r_6\}$. The lists after the relevant deletions have been made are listed in Figure 2.4 (no resident becomes bound to two hospitals as a result of these deletions, so no further deletions are necessary). It can be seen that a second iteration of the main loop of Algorithm HRT-strong-H terminates with an empty critical set. One of the two feasible matchings $(\{(r_1, h_4)(r_2, h_5)(r_3, h_6)(r_4, h_1)(r_5, h_2)(r_6, h_3)(r_7, h_7)(r_8, h_8)(r_9, h_4)(r_{10}, h_1)(r_{11}, h_5)(r_{12}, h_9)\}$ or $\{(r_1, h_4)(r_2, h_5)(r_3, h_6)(r_4, h_1)(r_5, h_2)(r_6, h_3)(r_7, h_8)(r_8, h_7)(r_9, h_4)(r_{10}, h_1)(r_{11}, h_5)(r_{12}, h_9)\})$, both of which are strongly stable, is output by the algorithm.

2.5 Implementation and analysis of Algorithm HRT-strong-H

The implementation and analysis of the hospital-oriented algorithm is similar to that for the resident-oriented algorithm. The only additional complication is that we must be able to identify a resident who is bound to more than one hospital, so we can delete the tail of that resident. To this end we maintain an array B of residents, with the array elements being a count of how many hospitals a resident is bound to, and a list of residents who are bound to more than one hospital. A resident can become bound to a hospital h when he is first provisionally assigned to h, or when another resident who is provisionally assigned to h is deleted from h's list. In both cases we need to know how many residents h is currently provisionally assigned, but such a count is already maintained as it is required for other operations. In the former case we also need to know how many residents are in $t(h)$, but a simple scan ahead through h's list to find the end of $t(h)$ suffices, and, in total, all such scans take $O(a)$ time. In the latter case, when h changes from being over-subscribed to fully-subscribed, we need only track back through the tie preceding $t(h)$, and, for each resident in that tie, increment by one the corresponding element in B. Again, in total, all such scans take $O(a)$ time. It follows, by the argument in Section 2.3, that Algorithm HRT-strong-H also has $O(a^2)$ time complexity.

r_1:	$h_4\ (h_1\ h_2\ h_3)$	r_1:	h_4
r_2:	$h_5\ (h_1\ h_2\ h_3)$	r_2:	h_5
r_3:	$h_6\ (h_1\ h_2\ h_3)$	r_3:	h_6
r_4:	$h_1\ (h_4\ h_5\ h_6\ h_9)$	r_4:	h_1
r_5:	$h_2\ (h_4\ h_5\ h_6)$	r_5:	h_2
r_6:	$h_3\ (h_4\ h_5\ h_6)$	r_6:	h_3
r_7:	$(h_7\ h_8)\ (h_2\ h_3)$	r_7:	$(h_7\ h_8)$
r_8:	$(h_7\ h_8)$	r_8:	$(h_7\ h_8)$
r_9:	$(h_4\ h_7)$	r_9:	$(h_4\ h_7)$
r_{10}:	$(h_1\ h_9)\ h_8$	r_{10}:	$(h_1\ h_9)$
r_{11}:	h_5	r_{11}:	h_5
r_{12}:	h_9	r_{12}:	h_9

h_1:(3)	$r_{10}\ (r_1\ r_2\ r_3)\ r_4$	h_1:(3)	$r_{10}\ r_4$
h_2:(1)	$r_7\ (r_1\ r_2\ r_3)\ r_5$	h_2:(1)	r_5
h_3:(1)	$r_7\ (r_1\ r_2\ r_3)\ r_6$	h_3:(1)	r_6
h_4:(3)	$(r_4\ r_5\ r_6\ r_9)\ r_1$	h_4:(3)	$r_9\ r_1$
h_5:(2)	$r_{11}\ (r_4\ r_5\ r_6)\ r_2$	h_5:(2)	$r_{11}\ r_2$
h_6:(1)	$(r_4\ r_5\ r_6)\ r_3$	h_6:(1)	r_3
h_7:(1)	$(r_7\ r_8\ r_9)$	h_7:(1)	$(r_7\ r_8\ r_9)$
h_8:(1)	$(r_7\ r_8)\ r_{10}$	h_8:(1)	$(r_7\ r_8)$
h_9:(1)	$(r_4\ r_{10}\ r_{12})$	h_9:(1)	$(r_{10}\ r_{12})$
Initial preference lists		Lists after first loop iteration	

Figure 2.4: The preference lists for an example HRT instance

2.6 Conclusions and subsequent work in the literature

In this chapter we have described two polynomial-time algorithms for the problem of finding a strongly stable matching, if one exists, given an instance of HRT. The algorithms produce a resident-optimal and a hospital-optimal strongly stable matching respectively. The resident-oriented algorithm was included in [28], in which it was also shown that the complexity of any algorithm for HRT cannot be better than the best possible for the problem of determining if a bipartite graph has a perfect matching. Subsequent to this publication, Kavitha et al. [30] gave an algorithm that finds a strongly stable matching

in SMTI in $O(ka)$ time, if one exists, where k is the total number of men and women in the instance. They also gave a resident-oriented algorithm that finds a strongly stable matching in $O(a(|R| + \sum_{h \in H} p_h))$ time, if one exists. It seems likely that this algorithm can be adapted to give a hospital-oriented algorithm with similar complexity. Additionally Malhotra [33] gave an $O(a^2)$ algorithm, an extension of the resident-oriented algorithm above, to find a strongly stable matching in an instance of many-to-many Stable Marriage. Again it seems likely that this algorithm can be modified to run in $O(a \sum_{q \in Q} p_q)$ time, where Q is the set of all participants in an instance. Malhotra also shows that the set of strongly stable matchings in a many-to-many stable marriage instance form a distributive lattice, when grouped into appropriate equivalence classes.

Chapter 3

Strong Stability in SRTI

3.1 Introduction

The problem of finding a strongly stable matching in an instance of Stable Roommates with Ties and Incomplete lists (SRTI) was listed as an open problem by Irving and Manlove [26]. In this chapter we present an efficient algorithm for this problem. As was noted in Section 2.1, there is a sense in which strong stability can be viewed as the most appropriate criterion for a practical matching scheme when there is indifference in the preference lists, and that in cases where a strongly stable matching exists, it should be chosen instead of a matching that is merely weakly stable. This is because a strongly stable matching is not susceptible to undermining by persuasion or bribery. SRTI is both a generalisation of SMTI and of SR, and so the algorithm presented here, Algorithm SRTI-strong, is a generalisation of those in [34] and [20]. There are, however, some notable differences between Algorithm SRTI-strong and Algorithms HRT-strong-R and HRT-strong-H, presented in Chapter 2. Algorithm SRTI-strong has two phases, in the manner of the algorithms for SR [20] and for SRTI under super-stability [26], whereas Algorithms HRT-strong-R and HRT-strong-H only have one. The first phase of Algorithm SRTI-strong mirrors the two HRT algorithms, though it is somewhat less complex because of the one-to-one nature of SRTI. In Section 3.2 we describe the first phase of Algorithm SRTI-strong, and in Section 3.3 we describe the second phase. In Section 3.4 we establish the complexity of the algorithm to be $O(a^2)$. Finally, we present our conclusions and discuss some advances in the literature in Section 3.5.

3.2 Phase 1 of Algorithm SRTI-strong

In this section we describe the first phase of the algorithm for determining whether a strongly stable matching exists in a given instance of SRTI, and if so finding one such matching. Before doing so we present a number of definitions and constructions relating to the algorithm.

For an agent p, $f(p)$ and $l(p)$ are the head and tail respectively of p's current list. During the execution of the algorithm agents become *semi-assigned* to other agents. Note that, as the name suggests, this is not a symmetric relation. If an agent p is semi-assigned to an agent q this does not imply that q is semi-assigned to p. It is possible for an agent p to be semi-assigned to more than one other agent, but only if all such agents are in $f(p)$.

The algorithm proceeds by deleting from the preference lists pairs that cannot be strongly stable. By the *deletion* of a pair $\{p, q\}$, we mean the removal of p and q from each other's lists, and, if p (resp. q) is semi-assigned to q (resp. p), the breaking of this semi-assignment.

The *semi-assignment graph* is a bipartite graph $G = (U, V, E)$, constructed as follows:

1. add a node for each agent p with a non-empty list to each of U and V, the *proposing node* for p and the *receiving node* for p respectively;
2. for each agent x, add an edge from $x \in U$ to $y \in V$ for every agent $y \in f(x)$.

For a given set Z of nodes, the *neighbourhood* of Z, denoted $\mathcal{N}(Z)$, is the set of nodes adjacent to at least one member of Z. The *deficiency* of Z, $\delta(Z)$, is defined by $\delta(Z) = |Z| - |\mathcal{N}(Z)|$, i.e., the difference in the sizes of the sets Z and $\mathcal{N}(Z)$. Let Z_1 and Z_2 be two sets of maximum deficiency. Then it can be shown that $Z_1 \cap Z_2$ is also a set of maximum deficiency. It follows that there is a unique minimal set of proposing nodes with maximum deficiency, which we refer to as the *critical set*.

The *final assignment graph* is a non-bipartite graph $G = (V, E)$, constructed as follows (note that we assume a total order on the agents):

1. add to V a node for each agent with a non-empty list;

2. for each agent x, add an edge from $x \in V$ to $y \in V$ for each agent $y \in f(x)$ such that $x > y$;

The algorithm, displayed in Figure 3.1, begins by setting each agent to be free (i.e., not semi-assigned to any other agent). The iterative stage of the phase involves each free agent p in turn becoming semi-assigned to the agents at the head of his list. Each pair $\{q, r\}$ such that $q \in f(p)$ and q prefers p to r is deleted. This continues until every agent is semi-assigned to one or more other agents or has an empty list. We then form the semi-assignment graph and find the critical set Z of agents. No agent in $\mathcal{N}(Z)$ can be assigned a resident from those in its tail in any strongly stable matching, so all such pairs are deleted. The iterative step is then reactivated, and this entire process continues until Z is empty, which must happen eventually, since, if Z is found to be non-empty, then at least one pair is subsequently deleted from the preference lists. On termination of the iterative step of phase 1, if there are an odd number of agents with non-empty lists there is no strongly stable matching for the instance. Otherwise we proceed to phase 2.

We now prove two lemmas relating to phase 1 of Algorithm SRTI-strong.

Lemma 3.2.1. *If the pair $\{p, q\}$ is deleted during phase 1 of Algorithm SRTI-strong then $\{p, q\}$ is not a strongly stable pair.*

Proof The proof is similar to Lemma 2.2.1. □

Lemma 3.2.2. *Every agent p who has a non-empty list at the end of phase 1 must be matched in any strongly stable matching.*

Proof At the end of phase 1, some agent q is semi-assigned to p. For, suppose not. Then the set U of all agents must have deficiency at least 1, a contradiction, as the critical set must be empty at the end of phase 1. If, in a strongly stable matching M, p is unmatched then, by Lemma 3.2.1, q must be either unmatched in M, or matched with an agent r such that q prefers p to r or is indifferent between them. In either case $\{p, q\}$ blocks M, and the result follows from this contradiction. □

The following two corollaries are immediate from the above result.

Corollary 3.2.3. *For a given instance of SRTI, the set of agents may be partitioned into those agents matched in all strongly stable matchings and those who are matched in none.*

set each agent to be free;
repeat {
  while some agent p is free **and** has a non-empty list {
    for each $q \in f(p)$ {
      p becomes semi-assigned to q;
      for each r such that q prefers p to r {
        if r is semi-assigned to q
          break the semi-assignment;
        delete the pair $\{q, r\}$ from the preference lists; }}}
  form the semi-assignment graph;
  find the critical set Z of agents;
  for each agent $p \in \mathcal{N}(Z)$
    for each agent q in the tail of p's list
      delete the pair $\{p, q\}$;
} **until** $Z = \emptyset$;
if there are an odd number of agents with non-empty lists
  no strongly stable matching exists; }}

Figure 3.1: Phase 1 of Algorithm SRTI-strong

The former set comprises those agents whose lists are non-empty on termination of phase 1.

Corollary 3.2.4. *For a given instance of SRTI, if the number of agents with non-empty lists at the end of phase 1 is odd then there is no strongly stable matching.*

Note that it would be possible, in certain circumstances, to find a strongly stable matching at the end of phase 1, and additionally, there are certain circumstances beyond those highlighted by Corollary 3.2.4 under which it could be concluded that no strongly stable matching exists. However, including these in phase 1 would unnecessarily complicate the algorithm, and so they will be dealt with in phase 2.

3.3 Phase 2 of Algorithm SRTI-strong

We start by defining a number of terms which are required for the second phase of the algorithm.

We call the lists at the end of phase 1, with those agents who have empty lists removed, the *phase 1 table*, denoted T^1. A *preference table*, or *table* for brevity, is a subset of the pairs contained within T^1. For a given table T, T_p is the list of agent p in T, with order induced from p's original preference list. We denote by $f_T(p)$ and $l_T(p)$ the head and tail respectively of T_p. Finally we extend the notion of preference to sets. An agent p *prefers* a set P of agents to a set Q of agents if and only if p prefers every member of P to every member of Q.

A *strongly stable table* is a table T which satisfies the following conditions:

i) $q \in f_T(p) \Rightarrow p \in l_T(q)$.

ii) the acceptable pair $\{p, q\}$ is absent from T if and only if p prefers $l_T(p)$ to q or q prefers $l_T(q)$ to p.

iii) there is no agent p such that $|T_p| < 1$ (i.e., p has an empty list).

It can be shown that the phase 1 table is itself a strongly stable table.

Let T be a strongly stable table, and let x be an agent with $f_T(x) \neq l_T(x)$. We denote by $T^{f(x)}$ the table formed by deleting $\{y, z\}$ from T for each $y \in f_T(x)$ and $z \in l_T(y)$, and applying the main loop of phase 1. We denote by $T^{l(x)}$ the table obtained by deleting from T $\{x, y\}$ for all $y \in l_T(x)$, and applying the main loop of phase 1. Note that, in the first case x becomes free so that the termination condition of the main loop of phase 1 is no longer satisfied. In the second case some agent may become free, but if not then the critical set must be non-empty, since no agent is semi-assigned to x. The reason for forming $T^{f(x)}$ and $T^{l(x)}$ is as follows. Suppose $M \subseteq T$, for some strongly stable matching M. By Lemma 3.2.2, x must be matched in M. We want to form a strongly stable table contained in T, which also contains a strongly stable matching. If $p_M(x)$ is in $f_T(x)$, then $\{x, p_M(x)\}$ must appear in $T^{l(x)}$, at least before the re-activation of the main loop of phase 1. Otherwise $\{x, p_M(x)\}$ must appear in $T^{f(x)}$, at least before the re-activation of the main loop of phase 1. In fact, in Lemma 3.3.1 we show that one of these tables is a strongly

stable table containing M.

Lemma 3.3.1. *Let T be a strongly stable table that contains a strongly stable matching M, and let x be an agent such that $f_T(x) \neq l_T(x)$. Then either $T^{f(x)}$ or $T^{l(x)}$ is a strongly stable table containing M.*

Proof There are two cases to consider.

Case 1: ($T^{f(x)}$) Suppose $\{x, a\} \in M$ for some $a \in l_T(x)$. Since $f_T(x) \neq l_T(x)$, $\{x, y\} \notin M$, for all $y \in f_T(x)$. Suppose $\{z, y\} \in M$ for some $z \in f_T(x)$, $y \in l_T(z)$. Since $z \in f_T(x)$, $x \in l_T(z)$, by the first defining property of a strongly stable table, so z is indifferent between x and $y = p_M(z)$. Further, since $z \in f_T(x)$ and $a \in l_T(x)$, x prefers z to $a = p_M(x)$, so $\{x, z\}$ blocks M, a contradiction. Thus, after the deletion from T of $\{b, z\}$ for each $b \in l_T(z)$ for each $z \in f_T(x)$ it follows that each pair in M is still contained in the preference lists.

Now suppose that after the application of the main loop of phase 1 some pair $\{p, q\} \in M$ has been deleted. Suppose that this was the first such M-pair to be deleted. There are two cases to consider.

Case 1a: Suppose $\{p, q\}$ was deleted when an agent r whom p prefers to q became semi-assigned to p. Now r cannot be matched in M with an agent s whom he prefers to p, since the pair $\{r, s\}$ must have been deleted for r to become semi-assigned to p, and $\{p, q\}$ was the first M-pair to be deleted. So p prefers r to $q = p_M(p)$, and r either prefers p to $p_M(r)$, or is indifferent between them. But then $\{p, r\}$ is a blocking pair for M, a contradiction.

Case 1b: Suppose $\{p, q\}$ was deleted because $q \in \mathcal{N}(P)$ for some critical set P, at a point at which p is in $l_U(q)$, where $\mathcal{L}$ denotes the preference lists immediately before the critical set deletions are made. Let B be the set of agents r such that $r \in \mathcal{N}(P)$, and $p_M(r) \in l_{\mathcal{L}}(r)$. Then $q \in B$. Now $|\mathcal{N}(B) \cap P| > |B|$, as otherwise $\mathcal{N}(B) \cap P$ is a set of agents contained in P who are collectively semi-assigned to a set of agents of the same size or greater, so removing the agents in $\mathcal{N}(B) \cap P$ from P will leave a set strictly contained in P with deficiency $\delta(P)$ or greater, a contradiction. It follows that there exists an agent a such that $a \in \mathcal{N}(B) \cap P$ and $\{a, p_M(a)\}$ is contained in the preference lists after the critical set deletions. Note that every agent to whom a is semi-assigned is in $\mathcal{N}(P)$, by the definition of $\mathcal{N}(P)$, so it follows that a prefers every member of $f_{\mathcal{L}}(a)$ to $p_M(a)$. So there must be

an agent $b \in B$ such that a prefers b to $p_M(a)$. Further $a \in l_{\mathcal{L}}(b)$, $p_M(b) \in l_{\mathcal{L}}(b)$ (since $b \in B$), so b is indifferent between a, $p_M(b)$. But then $\{a, b\}$ is a blocking pair for M, a contradiction.

It follows that M is contained in $T^{f(x)}$. It remains to prove that $T^{f(x)}$ is a strongly stable table. It follows from the algorithm that the first two defining properties for a strongly stable table hold for $T^{f(x)}$. Since M is a strongly stable matching contained in $T^{f(x)}$ it is clear, by Lemma 3.2.2, that the third defining property holds.

Case 2: ($T^{l(x)}$) Suppose $\{x, a\} \notin M$ for any $a \in l_T(x)$. Then, after the deletion of $\{x, a\}$ for each $a \in l_T(x)$ no pair from M has been deleted from the preference lists. The remainder of the argument is exactly as in Case 1. □

Now we must decide which of $T^{f(x)}$ and $T^{l(x)}$ to work with. It is entirely possible that we have deleted a strongly stable pair when forming one or both of these tables, even though we know that there is a strongly stable matching contained in T which is also contained in at least one of $T^{f(x)}$ and $T^{l(x)}$. The next two results show that, if no agent violates the third defining property of a strongly stable table in $T^{f(x)}$, then $T^{f(x)}$ contains a strongly stable matching.

Lemma 3.3.2. *Let T be a strongly stable table in which some agent x has a list such that $f_T(x) \neq l_T(x)$. Then either $T^{f(x)}$ is a strongly stable table, or some agent in $T^{f(x)}$ has an empty list.*

Proof Suppose no agent has an empty list in $T^{f(x)}$. Then it follows from the algorithm that all three defining properties of a strongly stable table hold. □

Lemma 3.3.3. *Let T be a strongly stable table that contains a strongly stable matching M. Suppose that some agent k has a list such that $f_T(k) \neq l_T(k)$. Suppose further that no agent has an empty list in $T^{f(k)}$. Then $T^{f(k)}$ contains a strongly stable matching.*

Proof By Lemma 3.3.2, $T^{f(k)}$ is a strongly stable table. We describe how to construct a strongly stable matching M' contained in $T^{f(k)}$.

For each pair $\{p, q\} \in M$ such that $\{p, q\} \in T^{f(k)}$, place $\{p, q\}$ in M'. For the remaining agents construct a bipartite graph $G = (U_1, U_2, E)$ as follows: add a node to U_1 for each agent x who prefers $l_{T^{f(k)}}(x)$ to $p_M(x)$, and add a node to U_2 for each agent y who does not

prefer $l_{T^{f(k)}}(y)$ to $p_M(y)$; add an edge from $x \in U_1$ to $y \in U_2$ if and only if $x \in f_{T^{f(k)}}(y)$. Let A be a perfect matching in G (we show below that such a matching must exist). For each edge e in A, we add e to M'.

To prove that there is a perfect matching in G, we firstly need to show that U_1 and U_2 have the same cardinality. First we show that every agent semi-assigned to an agent in U_1 must be in U_2. Let q be any member of U_1. Since no agent in $T^{f(k)}$ has an empty list, there must be some agent semi-assigned to q in $T^{f(k)}$. Let r be any such agent. Suppose $r \notin U_2$. Then r prefers q to $p_M(r)$. But, since $q \in U_1$, q prefers r to $p_M(q)$, and so $\{r, q\}$ blocks M, a contradiction. Now, we show that U_1 and U_2 have the same cardinality. For each agent $v \in U_2$, v prefers $p_M(v)$ to $l_{T^{f(k)}}(v)$, or is indifferent between them. Since $\{v, p_M(v)\}$ has been deleted it follows, by the second defining property of a strongly stable table, that $p_M(v)$ prefers $l_{T^{f(k)}}(p_M(v))$ to v, and hence $p_M(v) \in U_1$. Hence $|U_1| \geq |U_2|$. Suppose $|U_1| > |U_2|$, and let P be the set of all agents. Since $|U_1| > |U_2|$, it follows that $|P \setminus U_1| < |P \setminus U_2|$. But every agent in $P \setminus U_2$ can only be semi-assigned to agents in $P \setminus U_1$, since no agent from $P \setminus U_2$ is semi-assigned to any agent in U_1. Further, no agent in $T^{f(k)}$ has an empty list, so $P \setminus U_2$ must be a set with positive deficiency, implying that the critical set is non-empty, a contradiction. Hence there is an agent $p \notin U_2$ such that p is semi-assigned to some agent $q \in U_1$, a contradiction. It follows that $|U_2| = |U_1|$.

Now suppose there is no perfect matching in G. Since $|U_1| = |U_2|$, and there is no deficient set in the semi-assignment graph, there must be a set S of agents in U_2 such that the agents in S are collectively semi-assigned to fewer than $|S|$ agents in U_1. But every agent semi-assigned to an agent in U_1 must be in U_2, so the agents in $S \cup (P \setminus U_2)$ are collectively semi-assigned to fewer than $|P \setminus U_1| + |S|$ agents. But then $S \cup (P \setminus U_2)$ has deficiency at least one in the semi-assignment graph, a contradiction. It follows that there is a perfect matching in G.

Finally we show that M' is strongly stable. Suppose, for a contradiction, that it is not. Then there exists a blocking pair $\{p, q\}$ for M'. If neither p nor q has a worse partner in M' than in M then $\{p, q\}$ blocks M, a contradiction. So suppose p prefers $s = p_M(p)$ to $r = p_{M'}(p)$, i.e., $p \in U_2$. By the construction of M', $r \in f_{T^{f(k)}}(p)$, and $p \in l_{T^{f(k)}}(r)$. Since $\{p, q\}$ blocks M', p prefers q to r, or is indifferent between them. Suppose p is indifferent between q and r. Then $q \in f_{T^{f(k)}}(p)$, and so $p \in l_{T^{f(k)}}(q)$. But, for $\{p, q\}$ to block M', q must prefer p to $p_{M'}(q)$, and so q must prefer $l_{T^{f(k)}}(q)$ to $p_{M'}(q)$, a contradiction, as this

implies $\{q, p_{M'}(q)\} \notin T^{f(k)}$. Hence p prefers q to r. It follows that $\{p, q\} \notin T^{f(k)}$, so, by the second defining property of a strongly stable table, either p prefers $l_{T^{f(k)}}(p)$ to q, or q prefers $l_{T^{f(k)}}(q)$ to p. In the former case $\{p, r\}$ cannot, by the second defining property of a strongly stable table, be in $T^{f(k)}$, a contradiction. In the latter case, since q prefers p to $p_{M'}(q)$, or is indifferent between them, it follows that q prefers $l_{T^{f(k)}}(q)$ to $p_{M'}(q)$, a contradiction again. Thus M' is strongly stable and the result follows. □

We are now in a position to describe phase 2 of the algorithm, which is displayed in Figure 3.2. We start by setting T to be the phase 1 table T^1. So long as there is some agent x for whom $f_T(x) \neq l_T(x)$, we form $T^{f(x)}$ and $T^{l(x)}$. If no agent in $T^{f(x)}$ has an empty list we set T to be $T^{f(x)}$. Otherwise, if no agent in $T^{l(x)}$ has an empty list we set T to be $T^{l(x)}$. If neither of these conditions is satisfied, no strongly stable matching exists for the instance. If the iteration in phase 2 terminates, we construct the final assignment graph, and find a perfect matching in that graph, which must exist and be strongly stable (see Lemma 3.3.4).

$T := T^1$;
while some agent x has $f_T(x) \neq l_T(x)$ {
  delete $\{z, w\}$ from T for each $z \in f_T(x)$ and $w \in l_T(z)$,
  and apply main loop of phase 1 to obtain $T^{f(x)}$;
  delete $\{x, y\}$ from T for all $y \in l_T(x)$,
  and apply main loop of phase 1 to obtain $T^{l(x)}$;
  if no agent in $T^{f(x)}$ has an empty list
    $T := T^{f(x)}$;
  else if no agent in $T^{l(x)}$ has an empty list
    $T := T^{l(x)}$;
  else {
    report no strongly stable matching exists;
    halt; }}
let G be the final assignment graph;
let M be a perfect matching in G;
output M, a strongly stable matching;

Figure 3.2: Phase 2 of Algorithm SRTI-strong

We have one final lemma to prove.

Lemma 3.3.4. *Suppose that the iteration in phase 2 terminates, and let G be the final assignment graph. Then there must be perfect matching M in G, and M must be strongly stable.*

Proof Suppose, for a contradiction, that there is no perfect matching in G. Then, by Philip Hall's theorem, there must be a set S_1 of agents who are collectively semi-assigned to a set S_2 of agents such that $|S_1| < |S_2|$. But then S_1 is a set of positive deficiency, implying that the critical set is non-empty, a contradiction.

Now suppose M is a perfect matching in the final assignment graph G, and suppose, for a contradiction, that M is blocked by $\{p, q\}$. It is clear from inspection of the algorithm that, if $\{p, q\}$ was deleted by the algorithm, then either p prefers $p_M(p)$ to q, or q prefers $p_M(q)$ to p. Hence $\{p, q\}$ cannot have been deleted. Clearly neither p nor q can have an empty list, so both must have a single tie as a list, and the other must be contained in that tie. But each of p and q has a partner in M selected from that tie, so both are indifferent between the other and their partner in M, a contradiction. The result follows. □

We now have all the results we need to conclude that Algorithm SRTI-strong is correct. We collate these results in the following theorem.

Theorem 3.3.5. *Algorithm SRTI-strong determines whether a strongly stable matching exists for a given instance of SRTI, and if so it outputs such a matching.*

Proof If phase 1 of Algorithm SRTI-strong reports that no strongly stable matching exists, then no strongly stable matching exists by Corollary 3.2.4. So suppose the algorithm enters phase 2. If the instance does not admit a strongly stable matching then, since there must be at least two deletions from the lists during each iteration of the while loop, ultimately either both $T^{f(x)}$ and $T^{l(x)}$ must contain an agent with an empty list, by Lemma 3.3.4, in which case the algorithm reports that no strongly stable matching exists. Conversely, if the instance admits a strongly stable matching then, by Lemmas 3.3.1, 3.3.2 and 3.3.3, we must end up with a strongly stable table in which every list consists of one tie, and which contains a strongly stable matching. At this point we find a perfect matching, which must exist and be strongly stable, by Lemma 3.3.4. □

Example 3.3.1. An example instance is displayed in Figure 3.3. Agent p_i is labeled i ($1 \leq i \leq 10$) for brevity, and agent i's preference list takes the form $i : P_i$. The instance is constructed so the

initial preference lists coincide with the phase 1 table, and there are an even number of agents with non-empty lists, so phase 2 is required.

1:	3 (2 9)
2:	(1 7) 6 8
3:	(4 6 7) 9 5 (1 10)
4:	(8 10) 5 3
5:	(6 9) 4 3 (7 8 10)
6:	7 2 (3 5)
7:	(5 9) (8 10) (2 3 6)
8:	(2 5) (7 10) (4 9)
9:	8 (1 10) 3 (5 7)
10:	(3 5) (8 9) 7 4

Phase 1 table

Figure 3.3: The phase 1 table T for an example SRTI instance

Let T be the phase 1 table. Since $f_T(1) \neq l_T(1)$, we can form $T^{f(1)}$ and $T^{l(1)}$. To form $T^{f(1)}$, we first delete the tail of agent 3. The reactivation of the main loop of phase 1 then causes agent 1 to become semi-assigned to agents 2 and 9, which in turn causes the deletions of $\{9, 3\}$, $\{9, 5\}$, $\{9, 7\}$, $\{2, 6\}$ and $\{2, 8\}$. The lists at this point are displayed in Figure 3.4. It can be verified that the critical set at this point is $\{7, 8, 10\}$, with neighbourhood $\{5\}$. After deleting the tail of agent 5, it can be verified that $T^{f(1)}$ is as displayed in Figure 3.4. Since every agent has exactly one tie on his list we form the final assignment graph G and look for a perfect matching in G. The only perfect matching in G is $\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}, \{9, 10\}$, and it can be verified that this matching is indeed strongly stable.

For comparison, $T^{l(1)}$ goes through the main loop three times, with the critical sets at the end of the first and second iterations being $\{2, 6\}$ and $\{1, 5, 7, 8, 10\}$ respectively. The lists immediately before these critical set deletions are listed in Figure 3.5. It can be shown that agent 1 (amongst others) will end up with an empty list in $T^{l(1)}$, because his list will be empty immediately after the deletions necessitated by the second critical set.

1:	(2 9)	1:	(2 9)
2:	(1 7)	2:	1
3:	(4 6 7) 5	3:	(4 6)
4:	(8 10) 5 3	4:	3
5:	6 4 3 (7 8 10)	5:	6
6:	7 (3 5)	6:	(3 5)
7:	5 (8 10) (2 3 6)	7:	8
8:	5 (7 10) (4 9)	8:	(7 10)
9:	8 (1 10)	9:	(1 10)
10:	5 (8 9) 7 4	10:	(8 9)
	Intermediate lists		$T^{f(1)}$

Figure 3.4: Forming $T^{f(1)}$

1:	3	1:	3
2:	7 6 8	2:	6
3:	(4 6 7) 9 5 (1 10)	3:	4 9 5 (1 10)
4:	(8 10) 5 3	4:	(8 10) 5 3
5:	(6 9) 4 3 (7 8 10)	5:	9 4 3 (7 8 10)
6:	7 2 (3 5)	6:	2
7:	(5 9) (8 10) (2 3 6)	7:	(5 9) (8 10)
8:	(2 5) (7 10) (4 9)	8:	5 (7 10) (4 9)
9:	8 10 3 (5 7)	9:	8 10 3 (5 7)
10:	(3 5) (8 9) 7 4	10:	(3 5) (8 9) 7 4
	Before 1^{st} critical set deletions		Before 2^{nd} critical set deletions

Figure 3.5: Forming $T^{l(1)}$

3.4 Implementation and analysis of Algorithm SRTI-strong

We show that, for an instance of SRTI involving a mutually acceptable pairs, Algorithm SRTI-strong can be implemented to run in $O(a^2)$ time.

By a similar argument to that for Algorithm HRT-strong-R (see Section 2.3), phase 1 of Algorithm SRTI-strong has $O(a^2)$ time complexity.

Phase 2 is somewhat more complex. We use the method of [26] to limit the amount of work done. Two possible courses of action must be considered in each loop iteration. In the worst case, we may require $\Omega(a^2)$ time in each iteration to calculate $T^{f(x)}$ and $T^{l(x)}$, and yet only make $O(1)$ deletions per iteration, so requiring $\Omega(a)$ iterations, giving overall complexity no better than $O(a^3)$. To solve this problem we make a copy T_2 of the phase 1 table $T_1 = T^1$, which takes $O(a)$ time. We then delete the pairs $\{x, y\}$, for all $y \in l_{T_2}(x)$, from T_2, and we delete $\{z, w\}$ for each $z \in f_{T_1}(x)$ and $w \in l_{T_1}(z)$ from T_1. We then apply the main loop of phase 1 to T_1 and T_2, starting with the one that has had the greatest number of pairs deleted. Until we need to find a critical set in one of T_1 and T_2 we do not allow the number of deletions from the active table to exceed the number from the inactive table by more than 1. When we need to find a critical set in one table, say T_1, we find one augmenting path, taking $O(a)$ time. We then switch back to T_2 until either the application of the main loop of phase 1 to T_2 terminates, or we need to find a critical set in T_2, whichever comes first. In the latter case we find one augmenting path, and then switch back to T_1. By operating in this manner the difference in work done on T_1 and T_2 can never be greater than $O(a)$ time. Finally, we maintain two stacks of deleted pairs, one for each of T_1 and T_2. As soon as one of the phase 1 applications terminates we halt.

Suppose the application of phase 1 to T_1 terminates first (a similar argument applies if the other application terminates first). If no agent in $T_1^{f(x)}$ has an empty list, then we restore T_2 from the stack, and then create another copy of $T_1^{f(x)}$ from T_2 by applying the same deletions to T_2 that were applied to T_1. Otherwise we restore T_1 from the stack, and continue with the application of phase 1 to T_2. If this terminates with $T_2^{l(x)}$ containing an agent who has an empty list then we exit the algorithm with the conclusion that no strongly stable matching exists for the instance. Otherwise we form a second copy of $T_2^{l(x)}$ from T_1 by applying the same deletions to T_1 that were applied to T_2.

If the algorithm has not terminated with the conclusion that no strongly stable matching exists then we have two copies of a strongly stable table. Using the method of Section 2.3, we can show that the total time taken in traversing the path from the phase 1 table to the end of the execution of phase 2 via complete strongly stable tables is $O(a^2)$. Note that, when entering the main loop of phase 2 for the second and subsequent times we can still

make use of the maximum cardinality matching in the final semi-assignment graph, so the method of Section 2.3 will work over multiple applications of phase 1 of the algorithm. Finally, the time taken on an incomplete table cannot exceed the time taken on the parallel (complete) strongly stable table by more than $O(a)$ time. At least one pair is deleted in every iteration of the main loop of phase 2, so there are $O(a)$ such tables. It follows that there is $O(a^2)$ work done on the incomplete tables, and so phase 2 of Algorithm SRTI-strong, and hence Algorithm SRTI-strong, has $O(a^2)$ time complexity.

3.5 Conclusion

We have presented an algorithm which finds a strongly stable matching in an instance of SRTI in $O(a^2)$ time. As noted previously, Kavitha et al. [30] have shown that a strongly stable matching in an instance of SMTI can be found in $O(ka)$ time, as can a resident-optimal strongly stable matching for an instance of HRT (see Chapter 2), where k is the total number of men and women in the instance, and the sum of the number of residents and the number of hospital posts, respectively. It seems likely that the method employed can be extended to Algorithm SRTI-strong to reduce the complexity to $O(ka)$ time, though this remains an open problem at this time.

Chapter 4

Stable Fixtures

4.1 Introduction

In this chapter we introduce a new problem. The *Stable Fixtures problem* is a generalisation of the Stable Roommates problem in which each agent has a fixed integer *capacity*, and is to be assigned a number of partners less than or equal to that capacity subject to the normal stability criterion. The name derives from a possible application in which a set of individuals or teams take part in a competition in which the fixtures are to be specified in advance. Each agent has a specified target number of fixtures, and may play against each of the others at most once. Each ranks a subset of the others - his acceptable partners[1] - in order of preference. A set of fixtures is stable if there are no two mutually acceptable agents, who are not scheduled to play against each other, each of whom either prefers the other to one of his scheduled opponents or has a fixture list shorter than his capacity. Stable Fixtures is of interest in its own right, though it is a generalisation of SMI, SRI and HR. In this chapter we focus on the case of strictly ordered preference lists, but in Chapter 5 we return to preference lists with ties, when we study Stable Fixtures with Ties. We include the version with strictly ordered lists here, despite the title of this thesis, as it is a sensible step to take before embarking on a study of the more general version.

More formally, an instance of the Stable Fixtures problem (SF) consists of

[1]For consistency, we continue to refer to 'partners', although 'opponents' might be a more appropriate term in the context of SF.

- $P = \{p_1, \ldots, p_n\}$, the set of agents;
- for each i $(1 \leq i \leq n)$ an integer capacity c_{p_i} satisfying $1 \leq c_{p_i} \leq n-1$;
- for each i $(1 \leq i \leq n)$ a preference list P_{p_i} comprising a strictly ordered subset of $P \setminus \{p_i\}$, with length at least c_{p_i}.

If p_j appears on P_{p_i} we say that p_j is *acceptable* to p_i, otherwise we say that p_j is *unacceptable* to p_i. If p_j precedes p_k on P_{p_i} we say that p_i *prefers* p_j to p_k. We say that a pair $\{p_i, p_j\}$ is an *acceptable pair* if p_i is acceptable to p_j, and p_j is acceptable to p_i.

A *fixture allocation*, or simply an *allocation* is a set $\mathcal{A}$ of unordered pairs of agents $\{p_i, p_k\}$ such that $p_i \in P_{p_k}$, $p_k \in P_{p_i}$, and, for all i $(1 \leq i \leq n)$,

$$|\{p_j : \{p_i, p_j\} \in \mathcal{A}\}| \leq c_{p_i}.$$

The *size* of $\mathcal{A}$ is merely the number of pairs in $\mathcal{A}$. The members of the set $\{p_j : \{p_i, p_j\} \in \mathcal{A}\}$ are referred to as the *partners* of p_i in $\mathcal{A}$.

An acceptable pair $\{p_i, p_j\} \notin \mathcal{A}$ is a *blocking pair* for allocation $\mathcal{A}$ if

- either p_i has fewer than c_{p_i} partners, or p_i prefers p_j to at least one of his partners in $\mathcal{A}$, and
- either p_j has fewer than c_{p_j} partners, or p_j prefers p_i to at least one of his partners in $\mathcal{A}$.

An allocation that admits no blocking pair is said to be *stable*, and is otherwise *unstable*. A pair that belongs to some stable allocation is a *stable pair*.

Recall that, in SR, if a stable matching exists then it is complete, i.e., every agent is matched to exactly one other agent. By contrast, in SF, it is easy to construct an example to show that, even if all preference lists are complete, a stable allocation may have some agents who are matched with fewer other agents than their capacity. For example, consider an instance involving four agents, each of capacity 2, such that one particular agent is ranked last by each of the others. It can be shown that the sole stable allocation here has

size 3, and the unpopular agent has no partners. So, at least in this respect, SF behaves somewhat differently from SR.

In this chapter we present an extension of Irving's Stable Roommates algorithm which, for a given instance of SF, determines if a stable allocation exists, and if so finds one such allocation, all in $O(a)$ time, where a is the number of acceptable pairs in the instance. As in the Stable Roommates case, the Stable Fixtures algorithm is split into two phases, discussed in Sections 4.2 and 4.3 respectively. In both phases, the preference lists of the agents are successively reduced, though in different ways. In phase 1, unlike the Stable Roommates algorithm, if some agent's preference list ends up with fewer entries than that agent's capacity, then the only conclusion that can be drawn is that, in any stable allocation, the agent in question must be matched with every agent in his (reduced) preference list. In phase 2, however, we show that if some agent's list becomes shorter than the minimum of his capacity and the length of his list at the end of phase 1, we can immediately conclude that no stable allocation exists for the given instance. Finally, in Section 4.4 we consider the implementation and complexity of the algorithm.

4.2 Phase 1 of Algorithm SF

The first phase of the algorithm closely resembles the first phase of the Stable Roommates algorithm [20], in that it involves a sequence of proposals, with the proposer becoming committed to the proposee, and the proposee holding a commitment from the proposer. The main difference is that the commitments made, and the deletions of entries from preference lists, are controlled by the capacities of the agents.

Formally, we say that an agent p_i is *committed* to an agent p_j if p_i has made a proposal to p_j and the pair $\{p_i, p_j\}$ has not been deleted (see below), and we say that an agent p_j *holds a commitment from* an agent p_i if p_j has accepted a proposal from p_i and the pair $\{p_i, p_j\}$ has not been deleted. By the *deletion* of a pair $\{p_i, p_j\}$, we mean the removal of p_i from P_{p_j} and the removal of p_j from P_{p_i}, and if p_i (resp. p_j) is committed to p_j (resp. p_i), the breaking of that commitment. We say that an agent p_i is *dominated* on P_{p_j} if p_j prefers to p_i at least c_{p_j} agents who are committed to p_j. The use of deletions ensures that any proposal received by an agent during the algorithm's execution automatically results in the proposee holding a commitment from the proposer.

Every agent is initially committed to no-one and holds no commitments, and we say that such an agent is *free*. Each successive proposal is made by some agent p_i who is currently committed to fewer than c_{p_i} other agents; p_i will propose, and become committed, to the first agent on P_{p_i} to whom he is not already committed, p_j say. The commitment relation is not symmetric in general, so this need not imply that p_j is committed to p_i. If agent p_j now holds a number of commitments greater than or equal to his capacity c_{p_j} then all the pairs $\{p_j, p_k\}$, such that p_k is dominated on P_{p_j}, are deleted. This means that there will be no immediate rejections in the fixtures algorithm, so an agent p_j will accept any proposal made to him. Further, the deletions from P_{p_j} might include an agent who is committed to p_j, and so this agent now has at least one more proposal to make, unless he is committed to everyone on his current list. This loop continues as long as some agent p_i is committed to fewer than c_{p_i} other agents, and there is some agent on P_{p_i} to whom he is not committed. We call the set of preference lists obtained at the end of phase 1 the *phase 1 table*, denoted T^1. For a given agent x, we denote x's list in T^1 by T^1_x. On termination of the loop, if the sum over all agents of the lesser of an agent's capacity and the length of the agent's list in T^1 is odd then there cannot be a stable allocation for the instance (see Corollary 4.2.8), and so the algorithm reports this. The first phase of the algorithm is displayed in Figure 4.1.

assign each agent to be free;
while some agent p_i is committed to fewer than c_{p_i} agents **and**
there is some agent on P_{p_i} to whom he is not committed {
    $p_j :=$ first agent on P_{p_i} to whom p_i is not committed;
    p_i proposes, and becomes committed to p_j;
    if p_j now holds $\geq c_{p_j}$ commitments
        for each agent p_k dominated on P_{p_j}
            delete the pair $\{p_k, p_j\}$ from the preference lists; }
if $\sum_{p \in P} \min(c_p, |T^1_p|)$ is odd
    no stable allocation exists;

Figure 4.1: Phase 1 of Algorithm SF

Algorithm SF involves some non-determinism but, as with all the algorithms for solving variants of the Stable Marriage problem, this non-determinism makes no difference to the

outcome.

We will show in Lemma 4.2.3 that each agent x for whom $|T_x^1| \leq c_x$ is committed to, and holds a commitment from, every agent on T_x^1, and as a consequence must be matched with exactly this set of agents in any stable allocation.

We denote by $F(x)$ and $L(x)$ the sets of agents to whom x is committed and from whom x holds commitments respectively, and we denote by $l(x)$ the last agent on T_x^1. Clearly $|F(x)| \leq c_x$, $|L(x)| \leq c_x$, $y \in F(x) \Leftrightarrow x \in L(y)$, and $l(x)$ is undefined if T_x^1 is empty.

We are now in a position to give some properties of the phase 1 table.

Lemma 4.2.1. *Let $\{x, y\}$ be an acceptable pair.*

(i) T_x^1 and T_y^1 cannot both be empty.

(ii) If T_x^1 is empty then y prefers $l(y)$ to x.

(iii) If T_x^1 and T_y^1 are both non-empty then $\{x, y\}$ is absent from T^1 if and only if x prefers $l(x)$ to y or y prefers $l(y)$ to x.

Proof (i) It is clear from the algorithm that an agent who receives a proposal at any point must always hold at least one commitment, and cannot have an empty list in T^1. So if both T_x^1 and T_y^1 are empty, neither x nor y could have received a proposal, hence there is no way that the pair $\{x, y\}$ could have been deleted, giving a contradiction.

(ii) Because T_x^1 is empty, $\{x, y\}$ must have been deleted, and this must have resulted from x being dominated on P_y. Hence y prefers $l(y)$ to x.

(iii) If x prefers $l(x)$ to y, then $\{x, y\}$ must be absent from T^1 by the definition of $l(x)$, and similarly if y prefers $l(y)$ to x. On the other hand, if $\{x, y\}$ was deleted during phase 1 of the algorithm, then either x was dominated on P_y, or y was dominated on P_x. In the former case it follows that y prefers all the members of $L(y)$ to x, and in the latter x prefers all the members of $L(x)$ to y. It is clear that, in the first case $l(y) \in L(y)$, and in the second case $l(x) \in L(x)$, and the result follows. □

Lemma 4.2.2. *(i) If $\{x, y\}$ does not belong to T^1 then $\{x, y\}$ is not a stable pair.*

(ii) If $|T_v^1| < c_v$ then v is matched with at most $|T_v^1|$ agents in any stable allocation.

Proof (i) Suppose, for a contradiction, that $\{x, y\}$ is a stable pair that does not belong to T^1. Let $\mathcal{A}$ be a stable allocation containing the pair $\{x, y\}$, and suppose that $\{x, y\}$ was the first stable pair deleted during the execution of phase 1 of the algorithm. Suppose further, without loss of generality, that the deletion of $\{x, y\}$ took place when an agent z became committed to x. Then x must already have held $c_x - 1$ commitments from agents he prefers to y, say $u_1, \ldots, u_{c_x-1}$, and he must also prefer z to y. Let $U = \{u_1, \ldots, u_{c_x-1}, z\}$. Not all of the agents in U can be partners of x in $\mathcal{A}$, for $y \notin U$ is one such partner. So choose $u \in U$ such that $\{x, u\} \notin \mathcal{A}$. Suppose u prefers all of his partners in $\mathcal{A}$ to x. Then for u to have become committed to x during the execution of the algorithm some stable pair must already have been deleted, contradicting the assumption that $\{x, y\}$ was the first such deletion. It follows that u prefers x to at least one of his partners in $\mathcal{A}$, and we know that x prefers u to y, a contradiction of the stability of $\mathcal{A}$.

(ii) This follows at once from (i). □

Lemma 4.2.3. *For an instance of SF that admits a stable allocation, let x be an agent for whom $|T_x^1| \leq c_x$. Then x is committed to, and holds a commitment from, every agent on T_x^1. Further every pair of agents who are committed to each other in T^1 are matched in every stable allocation, so in particular x is matched with every agent on T_x^1 in every stable allocation.*

Proof At the end of phase 1, an agent x for whom $|T_x^1| \leq c_x$ must be committed to every agent in T_x^1, because of the termination condition. Suppose that x holds fewer than T_x^1 commitments, so that x is committed to more agents than the number of commitments he holds. Clearly the total number of commitments made and the total number of commitments held must be equal, so there is at least one agent y who holds more commitments than the number of commitments he has made. But an agent only holds as many commitments as his capacity, so y must be committed to fewer agents than his capacity, and fewer than $|T_y^1|$ agents, contradicting the fact that phase 1 has ended. Thus x must be committed to every agent on T_x^1, and must also hold a commitment from each of them.

Now let $\{r, s\}$ be such that r is committed to s and s is committed to r in T^1. Let $\mathcal{A}$ be a stable allocation, and suppose $\{r, s\} \notin \mathcal{A}$. Then, by Lemma 4.2.2(i), r is either matched with fewer than $|T_r^1|$ partners in $\mathcal{A}$, or r prefers s to at least one of his partners in $\mathcal{A}$, and similarly s is either matched with fewer than $|T_s^1|$ partners in $\mathcal{A}$, or s prefers r to at least

one of his partners in $\mathcal{A}$. But then $\{r, s\}$ blocks $\mathcal{A}$, a contradiction. □

Corollary 4.2.4. *Any agent x for whom $|T_x^1| = 0$ has no stable partners.*

Corollary 4.2.5. *For every agent x for whom T_x^1 is non-empty, $l(x) \in L(x)$.*

Proof If $|T_x^1| \leq c_x$ then, by Lemma 4.2.3, x holds a commitment from every agent on his list, so $L(x) = T_x^1$. Since $l(x) \in T_x^1$ it follows that $l(x) \in L(x)$. Otherwise it is clear, by a simple count of commitments made and held, that x must hold commitments from c_x agents. Thus the deletion step of the algorithm implies that x must hold a commitment from the last agent on his list, so $l(x) \in L(x)$. □

Lemma 4.2.6. *Any agent x for whom $|T_x^1| \geq c_x$ must be matched with exactly c_x agents in any stable allocation.*

Proof At the end of phase 1, x holds commitments from c_x agents, $y_1, ..., y_{c_x}$ say. Then for each i $(1 \leq i \leq c_x)$ x appears in the first c_{y_i} positions in $T_{y_i}^1$. If, in an allocation $\mathcal{A}$, x has fewer than c_x partners then, in particular, one of $y_1, ..., y_{c_x}$, say y_1, is not his partner, so, by Lemma 4.2.2 (i), $\{x, y_1\}$ blocks $\mathcal{A}$, giving a contradiction. □

The following two corollaries are immediate from Lemmas 4.2.3 and 4.2.6. Note that Corollary 4.2.8 is a sufficient, but not necessary, condition for non-existence of a stable allocation.

Corollary 4.2.7. *All stable allocations for a given instance of the Stable Fixtures problem have the same size.*

Corollary 4.2.8. *For a given instance of the Stable Fixtures problem, if $\sum_{p \in P} min(c_p, |T_p^1|)$ is odd, then there is no stable allocation.*

Note that there are certain circumstances under which a stable allocation could be identified at the end of phase 1, but this would unnecessarily complicate the algorithm, so we leave identification of stable allocations exclusively to phase 2.

Example 4.2.1. An example instance is displayed in Figure 4.2. The agent p_i is labeled i $(1 \leq i \leq 10)$ for brevity, and agent i's preference list takes the form $i : (c_i)\ P_i$.

Hand execution of the algorithm shows that, at the end of phase 1, agent 10 has an empty list and must therefore be unmatched in every stable allocation, agents 6 and 7 end up with only each

1:(2)	3 2 4 5 7 8 10	1:(2)	3 2 4 5
2:(2)	1 4 3 5 8 9	2:(2)	1 4 3 5
3:(2)	7 8 9 1 2 4 5 10	3:(2)	8 9 1 2 4
4:(2)	5 3 9 1 8 2	4:(2)	5 3 9 1 8 2
5:(2)	2 3 7 1 4 9 6 10	5:(2)	2 1 4 9
6:(2)	7 9 8 5	6:(2)	7
7:(1)	6 1 8 3 5 10	7:(1)	6
8:(1)	1 4 7 9 2 3 6	8:(1)	4 9 3
9:(1)	2 5 8 4 3 6	9:(1)	5 8 4 3
10:(1)	1 5 7 3	10:(1)	
	Initial preference lists		Phase 1 table

Figure 4.2: The initial preference lists and phase 1 table for an example Stable Fixtures instance

other on their lists and so are matched only with each other in every stable allocation, (thus agent 6, who has a capacity of 2, cannot meet his capacity in any stable allocation), and, by Lemma 4.2.3, agents 1 and 2 must also be matched in every stable allocation. Since $\sum_{p \in P} min(c_p, |T_p^1|)$ is even, phase 2 commences. The execution of phase 2 is described at the end of the next section.

4.3 Phase 2 of Algorithm SF

We first define a number of terms that are required for phase 2 of the algorithm.

A *preference table*, or *table* for brevity, is a subset of the pairs contained within the phase 1 table T^1. For a given table T and agent x, we use T_x to denote the preference list of x in T, with order induced from P_x.

In any preference table T, we denote by $F_T(x)$ the first $min(c_x, |T_x|)$ agents in T_x, and by $L_T(x)$ the set of agents $\{y : x \in F_T(y)\}$. We denote by $l_T(x)$ the last agent in T_x, and by $s_T(x)$ the first agent $y \in T_x$ such that $y \notin F_T(x)$ i.e., the agent in position $c_x + 1$ in T_x (note that $s_T(x)$ is only defined if $|T_x| > c_x$). Further, it is immediate that, at the end of phase 1, $F(x) = F_{T^1}(x)$, $L(x) = L_{T^1}(x)$, and $l(x) = l_{T^1}(x)$, so $l_{T^1}(x) \in L_{T^1}(x)$, for all x

such that T_x^1 is non-empty.

A *stable table* is a preference table T that satisfies the following conditions:

(i) an acceptable pair $\{x, y\}$ is absent from T if and only if x prefers $l_T(x)$ to y or y prefers $l_T(y)$ to x;

(ii) there is no agent x such that $|T_x| < \min(c_x, |T_x^1|)$;

(iii) for each agent x for whom T_x is non-empty, $l_T(x) \in L_T(x)$;

(iv) for each agent x, $|L_T(x)| = |F_T(x)|$.

It follows from Lemma 4.2.1, Corollary 4.2.5 and the execution of the algorithm that the phase 1 table is stable.

The essence of phase 2 of the algorithm is the generation of a sequence of nested preference tables, terminating when a stable allocation is reached, or when condition (ii) is violated, in which case we will show that no stable allocation exists for the instance.

The following lemma starts to explore some of the properties of stable tables.

Lemma 4.3.1. *Let $\mathcal{A}$ be an allocation of $min(c_x, |T_x^1|)$ fixtures for each agent x, and let T be a stable table.*

(i) If $\mathcal{A} \subseteq T$ then no pair that is absent from T can block $\mathcal{A}$.

(ii) If, for every agent x, $|T_x| = \min(c_x, |T_x^1|)$, then T is a stable allocation.

(iii) If T, U are stable tables and $F_T(x) = F_U(x)$ for all x, or equivalently $L_T(x) = L_U(x)$ for all x, then $T = U$.

Proof (i) Let $\{x, y\}$ be a blocking pair for $\mathcal{A}$. It is clear from the execution of phase 1 of the algorithm that if $|T_x^1| < c_x$ and $|T_y^1| < c_y$ then $\{x, y\}$ cannot have been deleted during phase 1. In this case, if $\{x, y\} \notin T$ then $|T_x| < |T_x^1|$, a contradiction. Otherwise, the result is a consequence of a simple extension of the argument in Lemma 4.2.1 (i), Lemma 4.2.6, and the first and second defining properties of a stable table.

(ii) This follows immediately from (i).

(iii) It follows from the definitions of F_T and L_T that $F_T(x) = F_U(x)$ for all x if and only

if $L_T(x) = L_U(x)$ for all x, and it is an easy consequence of the first and third defining properties of a stable table that, if one of these conditions is satisfied, then $T = U$. □

Before we can ascertain whether a stable allocation exists, we must first decide how we are going to further reduce the preference lists. As in the Stable Roommates problem, we make use of rotations to achieve this reduction. A *rotation exposed in a preference table* T is a sequence

$$\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$$

with $x_i \neq x_j$ $(0 \leq i < j \leq r-1)$, such that $x_i = l_T(y_i)$ and $y_{i+1} = s_T(x_i)$ for all i $(0 \leq i \leq r-1)$, where $i+1$ is taken modulo r. For each i we say that x_i, y_i and the ordered pair (x_i, y_i) are *in* the rotation ρ. The set $\{x_0, ..., x_{r-1}\}$ will be called the *X-set* of ρ, and the set $\{y_0, ..., y_{r-1}\}$ will be called the *Y-set* of ρ. All subscripts should be taken modulo r where appropriate, and it should be noted that the starting point of the sequence is arbitrary because of the cyclic nature of a rotation.

We prove that, in a stable table T, no agent x with $|T_x| \leq c_x$ can be in a rotation, and that the X-sets of two rotations are disjoint, as are their Y-sets.

Lemma 4.3.2. *Let T be a stable table. No agent x with $|T_x| \leq c_x$ can be in a rotation exposed in T.*

Proof Let x be an agent with $|T_x| \leq c_x$. It follows that $s_T(x)$ is undefined, so x cannot be in the X-set of any rotation. Suppose that x is in the Y-set of some rotation ρ. Then $x = s_T(y)$ for some y. But $x \notin F_T(y) \Rightarrow y \notin L_T(x)$, and x violates the fourth property of a stable table. The result follows. □

Lemma 4.3.3. *If ρ and σ are two rotations exposed in T then the X-sets of ρ and σ are disjoint, as are their Y-sets.*

Proof If $\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$ is exposed in a table T, then $x_{i+1} = l_T(y_{i+1}) = l_T(s_T(x_i))$, so the cyclic sequence $x_0, ..., x_{r-1}$, and hence the rotation ρ, is completely determined by any one of the x_i. Since $x_i = l_T(y_i)$, ρ is also completely determined by any one of the y_i. □

Lemma 4.3.4. *Let T be a stable table in which $|T_x| > c_x$ for some agent x. Then there is at least one rotation exposed in T.*

Proof We denote by $S(T)$ the set of agents x with $|T_x| > c_x$. Clearly, for any $x \in S(T)$, $s_T(x)$ is well defined, and $s_T(x) \in S(T)$. For, suppose $z = s_T(x) \notin S(T)$. Then $z \notin F_T(x)$, so $x \notin L_T(z)$, and $|F_T(z)| = |L_T(z)| < |T_z|$, a contradiction. Let $y = l_T(z)$. Since $z \in S(T)$ it follows that $y \notin F_T(z)$, so $z \notin L_T(y)$, and a similar argument to that for z shows that $y \in S(T)$.

Let $H(T)$ be a directed graph with node set $S(T)$, and for each node x let there be an outward edge to the node $l_T(s_T(x))$. Because every node in $H(T)$ has out-degree 1, $H(T)$ must contain at least one cycle. Suppose that the nodes in such a cycle are $x_0, ..., x_{r-1}$ in that order. Then, since $x_{i+1} = l_T(s_T(x_i))$ for $0 \leq i \leq r-1$, it follows that $(x_0, s_T(x_{r-1})), (x_1, s_T(x_0)), ..., (x_{r-1}, s_T(x_{r-2}))$ is a rotation exposed in T. □

The above proof gives an easy way to find a rotation exposed in T: starting from any agent x, traverse the unique path in $H(T)$ from the node x until some agent is visited twice. If ρ is the rotation generated by this traversal, we say that x *leads to* ρ *in* T. If x leads to ρ in T, but x is not itself in the X-set of ρ, then the path in $H(T)$ from the node x to the first node in the cycle corresponding to ρ is called the *tail* of the rotation.

For a stable table T and an agent x with $|T_x| > c_x > 1$ we denote by $k_T(x)$ the least favoured member of $L_T(x) \setminus l_T(x)$. If $\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$ is a rotation exposed in T, then we denote by T/ρ the table obtained from T by deleting, for $0 \leq i \leq r-1$, all pairs $\{y_i, z\}$ such that y_i prefers both x_{i-1} and $k_T(y_i)$, if defined, to z. If $k_T(y_i)$ is not defined then we simply delete all pairs $\{y_i, z\}$ such that y_i prefers x_{i-1} to z. The process of replacing T by T/ρ is referred to as *eliminating* the rotation ρ.

In phase 2 of the algorithm the preference table is reduced by successive elimination of rotations. The starting point is the phase 1 table, which we know is stable. Throughout the reduction, provided no agent violates the second defining property of a stable table, the current table will be shown to be stable, so that Lemma 4.3.4 applies throughout, and the reduction process can continue as long as some agent has a list longer than his capacity.

From here onwards a *short list* is a list T_x in a table T such that $|T_x| < \min(c_x, |T_x^1|)$. The

effect of rotation elimination is summarised in the following lemma.

Lemma 4.3.5. *Let $\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$ be a rotation exposed in a stable table T. Then, if T/ρ contains no short list,*

(i) $y_{i+1} \in F_{T/\rho}(x_i)$ for each i, $0 \leq i \leq r-1$.

(ii) $x_{i-1} \in L_{T/\rho}(y_i)$ for each i, $0 \leq i \leq r-1$.

(iii) $F_T(x) = F_{T/\rho}(x)$ for each x not in the X-set of ρ, and $L_T(y) = L_{T/\rho}(y)$ for each y not in the Y-set of ρ.

Proof (i) The pair $\{x_i, y_i\} = \{l_T(y_i), y_i\}$ is deleted, because y_i prefers both x_{i-1} and $k_T(y_i)$, if defined, to x_i. Since $y_{i+1} = s_T(x_i)$, $y_{i+1} \in F_{T/\rho}(x_i)$ unless $\{x_i, y_{i+1}\} = \{x_i, s_T(x_i)\}$ is also deleted. This deletion could only happen because x_i is y_j for some j, and y_j prefers both $k_T(y_j)$, if defined, and x_{j-1} to y_{i+1}, in which case all successors of y_{i+1} in $y_j = x_i$'s list are also deleted. As a consequence, $|T_{x_i}| < c_{x_i}$, contrary to the assumption.

(ii) This follows from (i) and the definitions of F_T and L_T.

(iii) The facts that, when ρ is eliminated, no agent x who is not in the X-set of ρ can lose any entry of $F_T(x)$, and no agent y who is not in the Y-set of ρ can lose any entry of $L_T(y)$, are immediate consequences of the definition of rotation elimination. □

By Lemma 4.3.2 and Lemma 4.3.5 (iii) no agent x with $|T_x^1| \leq c_x$ can have any agent deleted from his list in a stable table T when a rotation exposed in T is eliminated. Therefore any short list in a stable table T will be for an agent x with $|T_x^1| > c_x$, but $|T_x| < c_x$. The next lemma details some further properties of rotation elimination.

Lemma 4.3.6. *Let $\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$ be a rotation exposed in a stable table T. Then, provided that T/ρ contains no short list, it is a stable table contained in T.*

Proof If T/ρ contains no short list, then in order to prove that T/ρ is stable it suffices to prove that T/ρ satisfies properties (i), (iii) and (iv) for a stable table, namely

(i) an acceptable pair $\{x, y\}$ is absent from T/ρ if and only if x prefers $l_{T/\rho}(x)$ to y or y prefers $l_{T/\rho}(y)$ to x,

(iii) for each agent z for whom T_z is non-empty, $l_{T/\rho}(z) \in L_{T/\rho}(z)$, and

(iv) for each agent z, $|L_{T/\rho}(z)| = |F_{T/\rho}(z)|$.

(i) This follows immediately from the stability of T if $\{x, y\}$ is absent from T. Otherwise, if $\{x, y\}$ is deleted when ρ is eliminated, this must be because y is y_i for some i and y_i prefers both $k_T(y_i)$, if defined, and x_{i-1} to x, or x is y_i for some i and y_i prefers both $k_T(y_i)$, if defined, and x_{i-1} to y. This is sufficient to establish (i) for all cases.

(iii) By Lemma 4.3.5(iii), if z is not in the Y-set of ρ then $L_{T/\rho}(z) = L_T(z)$. Since T is stable we have $l_T(z) \in L_T(z)$ and it follows at once that $l_{T/\rho}(z) = l_T(z) \in L_{T/\rho}(z)$.

On the other hand, suppose that z is in the Y-set of ρ, say $z = y_1$. Then by the definition of rotation elimination, $l_{T/\rho}(y_1)$ is either x_0 or $w = k_T(y_1)$, if defined. In the former case, $x_0 \in L_{T/\rho}(y_1)$, by Lemma 4.3.5 (ii). In the latter case, $w \in L_T(y_1)$ by definition, and so $w \in L_{T/\rho}(y_1)$ also.

(iv) Since T is a stable table, we have $|F_T(z)| = |L_T(z)|$ for all z. If z is not in the Y-set of ρ then, by Lemma 4.3.5 (iii), $L_{T/\rho}(z) = L_T(z)$, and so the required result follows. Otherwise, if z is in the Y-set of ρ, say $z = y_1$, then $L_{T/\rho}(y_1) = (L_T(y) \setminus \{x_1\}) \cup \{x_0\}$, so $|L_{T/\rho}(y_1)| = |L_T(y_1)| = |F_T(y_1)| = |F_{T/\rho}(y_1)|$.

Finally, the fact that T/ρ is obtained by deleting pairs from T implies that $T/\rho \subseteq T$. □

We have one final lemma to prove before we state the main theorem for phase 2.

Lemma 4.3.7. *Suppose that T and U are stable tables, and that $U \subseteq T$. If $\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$ is a rotation exposed in T and if $F_U(x) \neq F_T(x)$ for at least one x that leads to ρ, then $U \subseteq T/\rho$.*

Proof If $x = u_0$ leads to ρ in T then there is a sequence $(u_1, v_1), ..., (u_{t-1}, v_{t-1})$ such that $v_i = s_T(u_{i-1})$ and $u_{i+1} = l_T(s_T(u_i))$ for $i = 0, ..., t-1$, and $u_t = x_k$ for some k $(0 \leq k \leq r-1)$.

If $F_T(u_0) \neq F_U(u_0)$ then there is at least one agent in $F_U(u_0)$ who is $s = s_T(u_0) = v_1$ or someone below s in T_{u_0}. Thus $l_U(s)$ must be someone equal to, or better than, the least favoured of u_0 or $k_T(s)$, if defined. But $u_1 = l_T(s)$ is below both u_0 and $k_T(s)$, if defined, and hence below $l_U(s)$ in T_s. It follows that $F_T(u_1) \neq F_U(u_1)$. This argument may now be repeated for $u_1, ..., u_t = x_k, x_{k+1}, ...$ to show that, for every j $(0 \leq j \leq r-1)$,

$F_T(x_j) \neq F_U(x_j)$ and $l_U(y_{j+1})$ is someone equal to, or better than, the least favoured of x_j or $k_T(y_{j+1})$, if defined. So none of the pairs deleted when ρ is eliminated from T is present in U, and as a consequence, $U \subseteq T/\rho$. □

For ease of reference we state two corollaries of Lemma 4.3.7.

Corollary 4.3.8. *Suppose that T and U are stable tables, and that $U \subseteq T$. If ρ is a rotation exposed in T, and if $F_U(x) \neq F_T(x)$ for some x in the X-set of ρ, then $U \subseteq T/\rho$.*

Corollary 4.3.9. *Suppose that T and U are stable tables, and that $U \subseteq T$. Then U can be obtained from T by eliminating a sequence of rotations. In particular, every stable table can be obtained from the phase 1 table by eliminating an appropriate sequence of rotations.*

Proof If $T \neq U$ then, by Lemma 4.3.1 (iii), $F_T(x) \neq F_U(x)$ for some x, and hence $|T_x| > c_x$. So, if x leads to ρ, Lemma 4.3.7 implies that $U \subseteq T_1 = T/\rho$. This argument may be repeated to produce a sequence $T_1, T_2, ...$ of stable tables, all of which contain U, until, for some k, $F_{T_k}(x) = F_U(x)$ for all x. It then follows, by Lemma 4.3.1 (iii), that $T_k = U$.

Since every stable table is contained within the phase 1 table, the second part of the corollary follows at once. □

We are now in a position to prove the main result for phase 2 of the algorithm.

Theorem 4.3.10. *If there is a stable allocation contained within a stable table T, and if ρ is a rotation exposed in T, then there is a stable allocation contained within T/ρ.*

Proof Let $\mathcal{A}$ be a stable allocation contained within T, and suppose that there is a rotation exposed in T. Let $\rho = (x_0, y_0), (x_1, y_1), ..., (x_{r-1}, y_{r-1})$ be such a rotation. If, for some i $(0 \leq i \leq r-1)$, x_i and y_i are not partners in $\mathcal{A}$ then, since a stable allocation is a special case of a stable table, it follows from Corollary 4.3.8 that $\mathcal{A} \subseteq \mathcal{T}/\rho$. So suppose x_i and y_i are partners in $\mathcal{A}$ for all i $(0 \leq i \leq r-1)$. Let $\mathcal{A}_\rho$ be obtained from $\mathcal{A}$ by replacing the pair $\{x_i, y_i\}$ by $\{x_i, y_{i+1}\}$ for each i $(0 \leq i \leq r-1)$.

First we show that $\mathcal{A}_\rho$ is an allocation. Suppose x appears in both the X-set and the Y-set of ρ, and $c_x = 1$. Then $x = x_i = y_j$ for some i, j $(0 \leq i, j \leq r-1)$. But then the pairs $\{x_i, y_i\} = \{y_j, y_i\}$ and $\{x_j, y_j\}$ can both be in $\mathcal{A}$ only if $x_j = y_i$, implying that $F_T(x_i) = \{y_i\} = \{x_j\} = \{l_T(y_j)\} = \{l_T(x_i)\}$, with the contradictory implication that T_{x_i}

contains just one entry. So if x appears in both the X-set and Y-set of ρ then $c_x > 1$. Further, it is clear that $x \notin F_{T/\rho}(x)$, as $x \notin T_x$. It follows that such an x has the same number of partners in $\mathcal{A}$ and $\mathcal{A}_\rho$. Further it is clear that any x appearing only once, or not at all, in the union of the X-set and Y-set of ρ has the same number of partners in $\mathcal{A}$ and $\mathcal{A}_\rho$, so $\mathcal{A}_\rho$ is certainly an allocation.

Next we prove that $\mathcal{A}_\rho \subseteq \mathcal{T}/\rho$. By Lemma 4.3.5 (i) $y_{i+1} \in F_{T/\rho}(x_i)$ for all i, and no pair that does not contain some y_i can be deleted when ρ is eliminated. Now suppose that some pair $\{x, y\} \in \mathcal{A}_\rho$ is such that $\{x, y\} \notin T/\rho$. Then one of x, y must be y_i for some i $(0 \leq i \leq r-1)$. Without loss of generality suppose $y = y_i$ for some i $(0 \leq i \leq r-1)$. Then y prefers $l_{T/\rho}(y)$ to x. We now show that $c_y > 1$. First suppose $(x, y) = (x_i, y_i) \in \rho$. Since $\{x_i, y_i\} \in \mathcal{A}_\rho$, it follows that $\{x_i, y_i\} = \{y_{j+1}, x_j\}$ for some $j \neq i$. If $x = x_j$ then x appears twice in the X-set of ρ, a contradiction. Thus $x = y_{j+1}$. But then y appears in both the X-set and the Y-set of ρ, and so, by the above argument, $c_y > 1$. Now suppose $(x, y) \notin \rho$. It follows that $\{x, y\} \in \mathcal{A} \subseteq \mathcal{T}$, and hence, in $\mathcal{A}$, y is assigned both x_i and x, so $c_y > 1$. It follows that $k_T(y)$ is defined.

Since $y = y_i$ for some i $(0 \leq i \leq r-1)$, $x_i (= l_T(y))$ is removed from T_y when ρ is eliminated, but no other member of $L_T(y)$ is removed from T_y. Hence $L_{T/\rho}(y) = \{x_{i-1}\} \cup L_T(y) \setminus l_T(y)$. If $\{x, y\} \notin T/\rho$, y must prefer $l_{T/\rho}(y)$ to x, so in particular y must prefer $k_T(y)$ to x, as $l_{T/\rho}(y)$ can be no better than this agent. Now, in $\mathcal{A}$, y is matched with both $x_i = l_T(y)$ and x (note that $x \neq l_T(y)$), so there is at least one member of $L_T(y) \setminus l_T(y)$ whom y is not matched with in $\mathcal{A}$, z say, though y prefers z to $l_T(y)$. Now, $z \in L_T(y)$, so $y \in F_T(z)$, but $y \notin F_{\mathcal{A}}(z)$, so z is matched with at least one agent in $\mathcal{A}$ to whom he prefers y. But then $\{z, y\}$ blocks $\mathcal{A}$, a contradiction, so $\mathcal{A}_\rho \subseteq \mathcal{T}/\rho$.

Finally we need to prove that $\mathcal{A}_\rho$ is stable. Suppose not. Then there exists a pair $\{u, v\}$ that blocks $\mathcal{A}_\rho$. Since $\mathcal{A}$ is stable, $\{u, v\}$ cannot block $\mathcal{A}$. Only the $x_i \in X$, $y_i \in Y$ have different partners in $\mathcal{A}$ and $\mathcal{A}_\rho$ and, of these, only the x_i have poorer partners. Hence one of u, v must be x_i for some i. Without loss of generality let $u = x_i$. But it is known from Lemma 4.3.1 (i) that $\{u, v\}$ must be in T/ρ to block $\mathcal{A}_\rho$, since $\mathcal{A}_\rho \subseteq \mathcal{T}/\rho$.

Now for $\{u, v\}$ to block $\mathcal{A}_\rho$ but not $\mathcal{A}$, u must prefer v to at least one agent he is assigned to in $\mathcal{A}_\rho$, but he must prefer every agent he is assigned to in $\mathcal{A}$ to v. Since the only change for the worse in u's partners in $\mathcal{A}$ and $\mathcal{A}_\rho$ is that y_i has been replaced by y_{i+1}, and y_{i+1} is the

least favoured member of $F_{T/\rho}(u)$, it follows that u must be matched with exactly $F_{T/\rho}(u)$ in $\mathcal{A}_\rho$. But then $\{u, v\} \in \mathcal{A}_\rho$, since u prefers v to y_{i+1} and $\{u, v\} \in T/\rho$, contradicting the fact that $\{u, v\}$ blocks $\mathcal{A}_\rho$. Hence $\mathcal{A}_\rho$ is stable. This completes the proof. □

This theorem, together with Corollary 4.3.9, has the following immediate corollary.

Corollary 4.3.11. *For an instance of Stable Fixtures that admits a stable allocation, every stable table contains at least one stable allocation.*

We now describe phase 2 of the algorithm. Phase 2 operates on a table T, initially the phase 1 table T^1. As long as there are no short lists in T, and some agent p has a list of length greater than c_p then there must be at least one rotation exposed in T, ρ say, and this rotation is eliminated from T. If the while loop terminates because there is a short list in T then there cannot be a stable allocation for the instance, so the algorithm reports that there is no stable allocation. On the other hand, if the loop terminates because no agent has a list of length greater than his capacity, then T, which specifies a stable allocation, is output by the algorithm. The second phase of Algorithm SF is displayed in Figure 4.3.

$T := T^1$;
while (there are no short lists in T) **and**
(some agent $p \in T$ is such that $|T_p| > c_p$) {
    find a rotation ρ exposed in T;
    $T = T/\rho$; }
if some list in T is short
    no stable allocation exists;
else
    output T, a stable allocation;

Figure 4.3: Phase 2 of Algorithm SF

Theorem 4.3.12. *Algorithm SF correctly determines whether an instance of SF admits a stable allocation, and outputs one if it does.*

Proof Suppose first that there is a stable allocation for the given instance. If phase 1 of the algorithm terminates with the conclusion that no stable allocation exists for the instance then we know, by Corollary 4.2.8, that there is no stable allocation for the instance, a

contradiction. Hence we must enter phase 2. We know, from Lemma 4.2.2 (i), that every stable allocation is contained within the phase 1 table and, as long as the condition of the while loop is satisfied we know, by Lemma 4.3.4, that there is a rotation exposed in the current table, and, by Lemma 4.3.6, the table formed by eliminating that rotation must be stable. Furthermore, by Corollary 4.3.11, there is a stable allocation contained within each successive stable table generated by the algorithm. It follows that the algorithm must terminate with a stable table T in which every agent x has a list of length $min(c_x, |T_x^1|)$, and T must be a stable allocation, by Lemma 4.3.1 (ii).

Conversely, suppose the instance does not admit a stable allocation. Suppose the algorithm does not terminate after phase 1. If the algorithm terminates with a stable table T in which every agent x has a list of length $min(c_x, |T_x^1|)$, T must specify a stable allocation, by Lemma 4.3.1 (ii), a contradiction. Hence some table produced during the execution of phase 2 must contain an agent with a short list, and the algorithm will conclude that there is no stable allocation for the instance. □

As with the classical Stable Roommates problem [20], the particular stable allocation generated for an instance of the Stable Fixtures problem depends on the particular set of rotations eliminated.

Example 4.3.1. Recall that Figure 4.2 shows the phase 1 table T^1 for an example Stable Fixtures instance. There is only one rotation exposed in T^1, namely $(4,3),(3,9),(5,1),(2,4)$. When it is eliminated we obtain T^2, displayed in Figure 4.4, in which there is again only one rotation exposed, namely $(5,2),(4,9),(3,8)$. When it is eliminated we obtain $T^3 = \mathcal{A}$, a stable allocation, displayed in Figure 4.4.

4.4 Implementation and analysis of Algorithm SF

We now show that the Stable Fixtures algorithm can be implemented in such a way as to have $O(a)$ worst-case complexity for an instance involving a acceptable pairs.

In phase 1 of the algorithm, deletion of a pair $\{p_i, p_j\}$ can be achieved in constant time by holding a separate ranking array for each agent such that, in the array for p_i, position j holds the index of the position that p_j occupies in P_{p_i}. This means that, when agent p_j is deleted from the list of agent p_i, p_i can be deleted from p_j's list using one look-up of the ranking array. This second part of the deletion is achieved in constant time if the

1:(2)	3 2	1:(2)	3 2
2:(2)	1 3 5	2:(2)	1 3
3:(2)	8 1 2	3:(2)	1 2
4:(2)	5 9 8	4:(2)	5 8
5:(2)	2 4 9	5:(2)	4 9
6:(2)	7	6:(2)	7
7:(1)	6	7:(1)	6
8:(1)	4 9 3	8:(1)	4
9:(1)	5 8 4	9:(1)	5
10:(1)		10:(1)	
Stable table T^2		Stable allocation $\mathcal{A}$	

Figure 4.4: The intermediate stable table and stable allocation for the example Stable Fixtures instance

preference lists are stored in an indexed, doubly linked structure. The number of deletions is bounded by a, the number of acceptable pairs, while the number of commitments is bounded by $2a$. At the point at which an agent p_i receives a commitment for the c_{p_i}th time it is simple to ascertain which element in P_{p_i} represents the least preferred of the c_{p_i} agents from whom p_i holds a commitment. Thereafter, when an agent p_i receives a further commitment, which he will automatically accept, the c_{p_i}th ranked of all the commitments he holds can be found by stepping backwards in P_{p_i} from the previous worst commitment to the first commitment found. The total number of such backward steps is bounded by $2a$, so phase 1 certainly has worst case complexity $O(a)$.

For phase 2 of the algorithm a stack is used to house a path traced out in the directed graph $H(T)$. When a vertex is reached that is already on the stack a rotation has been found and it may be recovered by popping the stack as far as the first occurrence of that vertex. The rotation may then be eliminated by deleting the necessary pairs, and each deletion is a constant time operation, as has been noted above. Each successive rotation search begins from the end of the previous tail, so each pair encountered is a new pair. This ensures that the same long tail will not be traversed more than once. Since the stack must be empty at the end of the execution of the algorithm (it only ever contains agents

who have lists longer than their capacities) the number of push and pop operations is the same. Further, at least one pair is deleted every time a pop operation is performed, so the number of pop (and hence push) operations cannot exceed a, and since every operation associated with finding a rotation and deleting a pair can be achieved in constant time, it follows that the whole algorithm is $O(a)$.

Chapter 5

Stable Fixtures with Ties

5.1 Introduction

In this chapter we extend our study of Stable Fixtures to the case where preference lists may include ties - we refer to this problem as Stable Fixtures with Ties (SFT). As with the problems previously discussed, the addition of ties to SF gives rise to three possible definitions of stability. Here our main focus is super-stability, but we briefly discuss both weak and strong stability as well.

We define an instance of SFT in the same manner in which we defined an instance of SF, except that preference lists may now contain ties. If p_j and p_k are contained in the same tie on P_{p_i} we say that p_i is *indifferent* between p_j and p_k.

We now define the three variants of stability, by giving the three possible definitions of a blocking pair.

An acceptable pair $\{p_i, p_j\} \notin \mathcal{A}$ is a *blocking pair for weak stability* for allocation $\mathcal{A}$ if

- either p_i has fewer than c_{p_i} partners, or p_i prefers p_j to all of the least preferred of its partners in $\mathcal{A}$, and
- either p_j has fewer than c_{p_j} partners, or p_j prefers p_i to all of the least preferred of its partners in $\mathcal{A}$.

An allocation that admits no such blocking pair is said to be *weakly stable*.

An acceptable pair $\{p_i, p_j\} \notin \mathcal{A}$ is a *blocking pair for super-stability* for allocation $\mathcal{A}$ if

- either p_i has fewer than c_{p_i} partners, or p_i prefers p_j to all of the least preferred of its partners in $\mathcal{A}$, or is indifferent between them, and
- either p_j has fewer than c_{p_j} partners, or p_j prefers p_i to all of the least preferred of its partners in $\mathcal{A}$, or is indifferent between them.

An allocation that admits no such blocking pair is said to be *super-stable*. A pair that belongs to some stable allocation is a *super-stable pair*.

An acceptable pair $\{p_i, p_j\} \notin \mathcal{A}$ is a *blocking pair for strong stability* for allocation $\mathcal{A}$ if

- either p_i has fewer than c_{p_i} partners, or p_i prefers p_j to all of the least preferred of its partners in $\mathcal{A}$, or is indifferent between them, and
- either p_j has fewer than c_{p_j} partners, or p_j prefers p_i to all of the least preferred of its partners in $\mathcal{A}$.

An allocation that admits no such blocking pair is said to be *strongly stable*.

Note that we usually just refer to a *blocking pair* as the context will make clear which type of blocking pair is intended.

SRTI is a restriction of SFT in which every agent has capacity 1. Ronn [46] showed that finding a weakly stable matching in an instance of SRTI is NP-hard. It follows that finding a weakly stable allocation in an instance of SFT is NP-hard. Additionally, it can be shown that weakly stable allocations in an instance of SFT may have different cardinalities. However, in common with the other matching problems discussed here, it can be shown that a maximum cardinality weakly stable allocation in an instance of SFT is at most twice the size of a minimum cardinality weakly stable allocation.

The remainder of this chapter is organised as follows. Firstly, we present an algorithm, linear in the input size, which determines whether an instance of SFT admits a super-stable allocation, and if so outputs one. The algorithm is split into two phases. Phase 1 is described in Section 5.2, and phase 2 is described in Section 5.3. An implementation of the algorithm which can be shown to run in time $O(a)$, where a is the number of acceptable

pairs, is described in Section 5.4. Finally, in Section 5.5 we make some brief comments on the problem of finding a strongly stable allocation in an instance of SFT. Note that some of the terminology and notation used in the following sections is identical to that of Chapter 4, while some is slightly different. To avoid confusion we include all the relevant definitions in this chapter.

5.2 Phase 1 of Algorithm SFT-super

The first phase of the algorithm closely resembles the first phase of the Stable Fixtures algorithm, in that it involves a sequence of proposals, with the proposer becoming committed to the proposee, and the proposee holding a commitment from the proposer. Mirroring the SF terminology, we say that an agent p_i is *committed* to an agent p_j if p_i has proposed to p_j and the pair $\{p_i, p_j\}$ has not been deleted (see below), and we say that an agent p_j *holds a commitment from* an agent p_i if p_j has accepted a proposal from p_i and the pair $\{p_i, p_j\}$ has not been deleted. The main change from Algorithm SF is that an agent may now become committed to more agents than his capacity. Note that an agent will always become committed to every agent in a given tie on his list at the same time. An agent may also hold more commitments than his capacity if he holds commitments from at least two agents in the last tie on his list, and there are enough of them that, were all to be rejected, the agent would hold fewer commitments than his capacity.

We make a number of additional definitions. By the *deletion* of a pair $\{p_i, p_j\}$, we mean the removal of p_i from P_{p_j} and the removal of p_j from P_{p_i}, and if p_i (resp. p_j) is committed to p_j (resp. p_i), the breaking of that commitment. The use of deletions ensures that any commitment received by an agent during the execution of the algorithm is automatically accepted. We say that an agent p_i is *tie-dominated* on P_{p_j} if p_j holds commitments from at least $c_{p_j} + 1$ agents he likes at least as much as p_i (possibly including p_i), at least two of whom are in a tie with p_i (again, possibly including p_i). We say that an agent p_i is *dominated* on P_{p_j} if p_j holds commitments from at least c_{p_j} agents he prefers to p_i. We describe an agent as *replete* if, at any time during the execution of the algorithm he holds a number of commitments greater than or equal to his capacity. Thus, once an agent becomes replete he remains so forever. Finally, we denote by $s(p_i)$ the first tie on p_i's list, the members of whom p_i is not committed to (note that a tie may be of unit length).

Every agent is initially committed to no-one and holds no commitments, and we say that such an agent is *free*. Each successive proposal is made by some agent p_i who is currently committed to fewer than c_{p_i} other agents. Agent p_i will propose to the members of $s(p_i)$, and as a result p_i will become committed to every member of $s(p_i)$. As in the case of SF, this commitment relation is not symmetric in general, so this need not imply that the members of $s(p_i)$ are committed to p_i. If an agent $p_j \in s(p_i)$ now holds a number of commitments greater than or equal to his capacity, we set p_j to be replete and delete all pairs $\{p_j, p_k\}$ such that p_k is dominated on P_{p_j}, thereby breaking any corresponding commitments. Note that this may still leave p_j holding more than c_{p_j} commitments. In this case every pair $\{p_j, p_k\}$ such that p_k is tie-dominated on P_{p_j} is deleted, and again any corresponding commitments are broken. As with the Algorithm SF, these deletions mean that there will be no immediate rejections, so an agent will accept any proposal made to him. This phase of the algorithm will continue as long as some agent p_i is committed to fewer than c_{p_i} other agents, and has some agent in his preference list to whom he is not committed. We call the set of preference lists obtained at the end of phase 1 the *phase 1 table*, denoted T^1. For a given agent x, we denote x's list in T^1 by T^1_x. On termination of the loop, if the sum over all agents of the lesser of an agent's capacity and the length of the agent's list in T^1 is odd then there cannot be a super-stable allocation for the instance (see Corollary 5.2.8), and so the algorithm reports this. The algorithm is displayed in Figure 5.1.

The algorithm described in Figure 5.1 involves some non-determinism, but as with all the algorithms for solving variants of SM, this makes no difference to the outcome.

We will show in Lemma 5.2.5 that each agent x for whom $|T^1_x| \leq c_x$ is committed to, and holds a commitment from, every agent on T^1_x, and as a consequence must be matched with exactly this set of agents in any stable allocation.

We denote by $F(x)$ and $L(x)$ the sets of agents to whom x is committed and from whom x holds commitments respectively, and we denote by $l(x)$ the last tie on T^1_x.

We are now in a position to give some properties of the phase 1 table.

Lemma 5.2.1. *In T^1, no agent x is committed to more than c_x agents.*

Proof Suppose, for a contradiction, that some agent x is committed to more than c_x agents in T^1. No agent y holds more than c_y commitments, otherwise at least one agent

assign each agent to be free and non-replete;
while some agent p_i is committed to fewer than c_{p_i} agents **and**
there is some agent in his preference list to whom he is not committed {
  for each $p_j \in s(p_i)$ {
    p_i becomes committed to p_j;
    if p_j holds c_{p_j} commitments {
      set p_j to be replete;
      for each p_k dominated on P_{p_j}
        delete the pair $\{p_j, p_k\}$ from the preference lists; }
    if p_j holds $c_{p_j} + 1$ commitments
      for each p_k tie-dominated on P_{p_j}
        delete the pair $\{p_j, p_k\}$ from the preference lists; }}
if $\sum_{p \in P} \min(c_p, |T_p^1|)$ is odd **or if**
some replete agent p_i has $|T_{p_i}^1| < c_{p_i}$
  no super-stable allocation exists;
else
  proceed to phase 2;

Figure 5.1: Phase 1 of Algorithm SFT-super

z is either dominated or tie-dominated on T_y^1, and hence $\{z, y\}$ will be deleted. But, since the number of commitments made must be the same as the number of commitments held, some agent p must hold more commitments than he has made. But an agent only holds as many commitments as his capacity, so p must be committed to fewer agents than his capacity, contradicting the fact that phase 1 has ended. □

Lemma 5.2.2. *Let $\{x, y\}$ be an acceptable pair.*

(i) T_x^1 and T_y^1 cannot both be empty.

(ii) If T_x^1 is empty then y prefers $l(y)$ to x.

(iii) If T_x^1 and T_y^1 are both non-empty then $\{x, y\}$ is absent from T^1 if and only if x prefers $l_T(x)$ to y or y prefers $l_T(y)$ to x.

Proof (i) The proof is identical to Lemma 4.2.1 (i).

(ii) The proof is identical to Lemma 4.2.1 (ii).

(iii) If x prefers $l(x)$ to y, then $\{x, y\}$ must be absent from T^1 by the definition of $l(x)$, and similarly if y prefers $l(y)$ to x. On the other hand, if $\{x, y\}$ was deleted during phase 1 of the algorithm, then there are two cases to consider. In the first case, either y was dominated on P_x, or x was dominated on P_y. In the former case it follows that y prefers all the members of $L(y)$, including $l(y)$, to x, and in the latter x prefers all the members of $L(x)$, including $l(x)$, to y. In the second case, y was tie-dominated on P_x, or x was tie-dominated on P_y. In the former case the tie containing x is deleted from y's preference list, and $l(y)$ will be at worst some agent from the tie preceding that which contained x, so y prefers $l(y)$ to x, and by a similar argument x prefers $l(x)$ to y in the latter case. □

Lemma 5.2.3. *(i) If $\{x, y\}$ does not belong to T^1 then $\{x, y\}$ is not a super-stable pair.*

(ii) If $|T_v^1| < c_v$ then v is matched with at most $|T_v^1|$ agents in any super-stable allocation.

Proof (i) Let $\mathcal{A}$ be a super-stable allocation containing the pair $\{x, y\}$, and suppose that $\{x, y\}$ was the first super-stable pair deleted during the execution of phase 1 of the algorithm. Suppose further, without loss of generality, that the deletion of $\{x, y\}$ took place when a person z became committed to x. For $\{x, y\}$ to have been deleted the commitment of z to x must either have caused y to be dominated on P_x, or caused y to be tie-dominated on P_x. Call the set of agents x holds commitments from at this point L_x, excluding y in the second scenario, if necessary. Now, none of the members of L_x can have a set of partners in $\mathcal{A}$ who are all better than x. For, suppose some such agent u does have a set of partners in $\mathcal{A}$ who are all better than x. Then, for u to have proposed to x, some super-stable pair must have been deleted, contradicting the fact that no super-stable pairs were previously removed from the table. Since $\{x, y\} \in \mathcal{A}$, x is not matched in $\mathcal{A}$ with at least one member of L_x, r say. It follows that $\mathcal{A}$ will be blocked by $\{x, r\}$, giving a contradiction.

(ii) This follows at once from (i). □

Lemma 5.2.4. *Suppose some replete agent p_i has $|T_{p_i}^1| < c_{p_i}$. Then there is no super-stable allocation for the instance.*

Proof Suppose, for a contradiction, that $\mathcal{A}$ is a super-stable allocation. By Lemma 5.2.3, p_i can be matched with at most $|T_{p_i}^1|$ agents in $\mathcal{A}$. Since p_i is replete there is some agent, p_j say, from whom p_i held a commitment during the execution of phase 1 of the algorithm,

such that p_i is not matched with p_j in $\mathcal{A}$. But, again by Lemma 5.2.3, p_j cannot be matched in $\mathcal{A}$, with c_{p_j} agents, all of whom he prefers to p_i, otherwise he would never have been committed to p_i. But then p_j is either matched with fewer than c_{p_j} agents in $\mathcal{A}$, or prefers p_i to at least one agent with whom he is matched in $\mathcal{A}$, or is indifferent between them. But then $\{p_i, p_j\}$ blocks $\mathcal{A}$, a contradiction. The result follows. □

We now restrict our attention to phase 1 tables which do not admit a replete agent with a list of length less than his capacity.

Lemma 5.2.5. *For an instance of SFT that admits a super-stable allocation, let x be an agent for whom $|T_x^1| \leq c_x$. Then x is committed to, and holds a commitment from, every agent on T_x^1. Further every pair of agents who are committed to each other in T^1 are matched in every super-stable allocation, so in particular x is matched with every agent on T_x^1 in every super-stable allocation.*

Proof The proof is identical to Lemma 4.2.3. □

We now demonstrate that every agent p for whom $|T_p^1| \geq c_p$ must be matched with c_p agents, in any super-stable allocation, implying that every super-stable allocation for a given instance of SFT must have the same size.

Lemma 5.2.6. *Any agent p for whom $|T_p^1| \geq c_p$ must be matched with exactly c_p agents in any super-stable allocation.*

Proof The proof is identical to Lemma 4.2.6. □

The following two corollaries are immediate from the above result.

Corollary 5.2.7. *All super-stable allocations for a given instance of SFT have the same size.*

Corollary 5.2.8. *For a given instance of SFT, if $\sum_{p \in P} \min(c_p, |T_p^1|)$ is odd then there is no super-stable allocation for the instance.*

As with Algorithms SRTI-strong and SF, there are certain circumstances under which a super-stable allocation could be identified at the end of phase 1, but this would unnecessarily complicate the algorithm, so we leave all identification of super-stable allocations to phase 2. An example of the execution of phase 1 of Algorithm SFT-super is included in the example at the end of the next section.

5.3 Phase 2 of Algorithm SFT-super

We now define a number of terms. In an extension of previous notation, let $p_{\mathcal{A}}(x)$ denote the partners of agent x in allocation $\mathcal{A}$.

A *preference table*, or *table* for brevity, is a subset of the pairs contained within the phase 1 table T^1. For a given table T and agent x, we use T_x to denote the preference list of x in T, with order induced from P_x.

Let x be an agent for whom $|T_x| < c_x$ in a preference table T. Then we define $F_T(x)$ to be T_x. Otherwise we denote by $F_T(x)$ the minimum set of agents T_x such that

- $|F_T(x)| \geq c_x$,
- there is no proper subset F of $F_T(x)$ such that $|F| \geq c_x$ and x prefers every agent in F to every agent in $F_T(x) \setminus F$ and
- there is no agent $y \in T_x$ such that $y \notin F_T(x)$ and x prefers y to any member of $F_T(x)$, or is indifferent between them.

We denote by $L_T(x)$ the set of agents $\{y : x \in F_T(y)\}$, by $l_T(x)$ the last tie on x's list, and by $s_T(x)$ the first tie on x's list, the members of whom are not in $F_T(x)$ (clearly $s_T(x)$ is only defined if $|T_x| > |F_T(x)|$). Recall that a tie can be of length 1. It is immediate that, at the end of phase 1, $F(x) = F_{T^1}(x)$, $L(x) = L_{T^1}(x)$, and $l(x) = l_{T^1}(x)$, so $l_{T^1}(x) \in L_{T^1}(x)$, for all x such that T^1_x is non-empty.

A *super-stable table* is a preference table T which satisfies the following conditions:

(i) the acceptable pair $\{x, y\}$ is absent from T if and only if x prefers $l_T(x)$ to y or y prefers $l_T(y)$ to x;

(ii) there is no agent x such that $|T_x| < \min(c_x, |T^1_x|)$;

(iii) $|F_T(x)| = \min(|T^1_x|, c_x)$ for all x;

(iv) for each agent x, $|L_T(x)| = |F_T(x)|$.

It follows from Lemmas 5.2.1 and 5.2.3 and the execution of the algorithm that the phase 1 table is super-stable. For brevity we say that an agent who violates the second defining

property of a super-stable table has a *short list*. We start to discuss some properties of super-stable tables in the following lemma.

Lemma 5.3.1. *(i) For an allocation $\mathcal{A}$ and a super-stable table T, if $\mathcal{A} \subseteq T$ no pair that is absent from T can block $\mathcal{A}$.*

(ii) If, in a super-stable table T, every agent x is such that $|T_x| = \min(|T_x^1|, c_x)$, then T is a super-stable allocation.

(iii) If T, U are super-stable tables and $F_T(x) = F_U(x)$ for all x, or equivalently $L_T(x) = L_U(x)$ for all x, then $T = U$.

Proof (i) Let $\{x, y\}$ be a blocking pair for $\mathcal{A}$. It is clear from the execution of phase 1 of the algorithm that if $|T_x^1| < c_x$ and $|T_y^1| < c_y$ then $\{x, y\}$ cannot have been deleted during phase 1. In this case, if $\{x, y\} \notin T$ then $|T_x| < |T_x^1|$, a contradiction. Otherwise, the result is a consequence of the first and second defining properties of a super-stable table.

(ii) This follows immediately from (i).

(iii) It follows from the definitions of $F_T(x)$ and $L_T(x)$ that $F_T(x) = F_U(x)$ for all x if and only if $L_T(x) = L_U(x)$ for all x, and it is an easy consequence of the first and third defining properties that, if one of these conditions is satisfied, then $T = U$. □

Before we can ascertain whether a super-stable allocation exists, we must first decide how we are going to further reduce the preference lists. To arrive at the method, we first introduce some notation and prove a number of preliminary results. For a super-stable table T, we introduce the notation $f_T(x)$ to denote the set $F_T(x) \setminus L_T(x)$. We denote by $T^{l(x)}$ the table obtained by deleting the pairs $\{x, y\}$, for all $y \in l_T(x)$, from T and then applying the main loop of phase 1. We denote by $T^{f(x)}$ the table obtained by deleting $\{z, y\}$ for each $z \in f_T(x)$ and for each y in z's preference list such that z prefers x to y, or is indifferent between them, and then applying the main loop of phase 1. The rationale behind this is simple. If we assume that T is a super-stable table containing a super-stable allocation $\mathcal{A}$, and x is an agent with $|T_x| > c_x$, then we can say the following about x and $\mathcal{A}$: if x is assigned in $\mathcal{A}$ an agent y such that $y \in l_T(x)$, then x is assigned every member of $L_T(x)$ in $\mathcal{A}$. Hence, we can make the deletions necessary to form $T^{f(x)}$ without deleting any member of $\mathcal{A}$, and we show later that $T^{f(x)}$ is a (smaller) super-stable table containing $\mathcal{A}$. If, on the other hand, x is not assigned an agent from $l_T(x)$ in $\mathcal{A}$, then we can make

the deletions necessary to form $T^{l(x)}$ without deleting any member of $\mathcal{A}$, and again we show later that $T^{l(x)}$ is a (smaller) super-stable table containing $\mathcal{A}$. The following lemma is a formal proof of this.

Lemma 5.3.2. *Let T be a super-stable table, let $\mathcal{A}$ be a super-stable allocation and let x be such that $|T_x| > c_x$. If $\mathcal{A} \subseteq T$ then either $T^{l(x)}$ is a super-stable table such that $\mathcal{A} \subseteq T^{l(x)}$ or $T^{f(x)}$ is a super-stable table such that $\mathcal{A} \subseteq T^{f(x)}$.*

Proof There are two cases to consider.

Case 1: ($T^{f(x)}$) Suppose $\{x, y\} \in \mathcal{A}$ for some $y \in l_T(x)$. Then, for each $z \in L_T(x)$, $\{x, z\} \in \mathcal{A}$. For, if not, there exists $w \in L_T(x)$ such that $\{x, w\} \notin \mathcal{A}$. But x prefers w to y, or is indifferent between them, and $x \in F_T(w)$ so $\{x, w\}$ blocks $\mathcal{A}$, a contradiction. Now we show that $f_T(x)$ is non-empty. Since $|T_x| > c_x$, we need only show that $l_T(x) \in L_T(x)$. It is an immediate corollary of Lemma 5.2.5 and the execution of phase 1 of the algorithm that $l_{T^1}(p) \in L_{T^1}(p)$ for every agent p for whom $T^1(p)$ is non-empty. Thereafter it is a consequence of the second defining property of a super-stable table and the execution of the algorithm that $l_T(p) \in L_T(p)$ for every agent p for whom $T^1(p)$ is non-empty. Thus $f_T(x)$ is non-empty. Since, for each $z \in f_T(x)$, $\{z, x\} \notin \mathcal{A}$, then neither is $\{z, y\}$ for each y such that z prefers x to y or is indifferent between them, as otherwise $\{z, x\}$ would block $\mathcal{A}$. It follows that, after deletion of these pairs from T, each pair in $\mathcal{A}$ is still contained in the preference lists. Now suppose that after application of the main loop of phase 1 some pair $\{p, q\} \in \mathcal{A}$ has been deleted. Suppose further that this was the first pair from $\mathcal{A}$ to be deleted, and it was deleted when an agent r became committed to p. For $\{p, q\}$ to have been deleted the commitment of r to p must have caused q to be dominated on P_p, or caused q to be tie-dominated on P_p. Call the set of agents p holds commitments from at this point S, excluding q in the second scenario, if necessary. Now, none of the members of S can have a set of partners in $\mathcal{A}$ who are all better than p. For, suppose some such agent u does have a set of partners in $\mathcal{A}$ who are all better than p. Then, for u to have become committed to p, some pair from $\mathcal{A}$ must have been deleted, a contradiction of the fact that $\{p, q\}$ was the first pair from $\mathcal{A}$ to be deleted. Since $\{p, q\} \in \mathcal{A}$, there is at least one member of S, v say, with whom p is not matched in $\mathcal{A}$. It follows that $\{p, v\}$ blocks $\mathcal{A}$, a contradiction. Hence $\mathcal{A} \subseteq T^{f(x)}$.

It remains to prove that $T^{f(x)}$ is a super-stable table. The first and third defining properties

of a super-stable table hold by arguments similar to those in Lemma 5.2.2 (iii) and Lemma 5.2.1 respectively. The second defining property holds by Lemmas 5.2.5 and 5.2.6. For the fourth defining property, first note that, by Lemma 5.2.5 and the second defining property, any agent p with a list of length $k < c_p$ has $F_{T^{f(x)}}(p) = L_{T^{f(x)}}(p)$, so certainly $|F_{T^{f(x)}}(p)| = |L_{T^{f(x)}}(p)|$. For the remaining agents, the second defining property implies that $\sum_{p \in P} F_{T^{f(x)}}(p) = \sum_{p \in P} F_T(p)$. By the second defining property again, it follows that $\sum_{p \in P} L_{T^{f(x)}}(p) = \sum_{p \in P} L_T(p)$, and by a simple counting argument based on the execution of the main loop of phase 1 of the algorithm, we can conclude that the fourth defining property must hold.

Case 2: ($T^{l(x)}$) Suppose $\{x, y\} \notin \mathcal{A}$ for any $y \in l_T(x)$. Then, after the deletion of $\{x, z\}$ for each $z \in l_T(x)$ no pair from $\mathcal{A}$ has been deleted from the preference lists. The remainder of the argument is as for Case 1. □

Now we must decide which of $T^{f(x)}$ and $T^{l(x)}$ to work with. It is entirely possible that we have deleted a super-stable pair when forming one or both of these tables, even though we know that there is a super-stable allocation contained in T which is also contained in at least one of $T^{f(x)}$ and $T^{l(x)}$. The next two results show that, if no agent has a short list in $T^{f(x)}$ then $T^{f(x)}$ contains a super-stable allocation.

Lemma 5.3.3. *Let T be a super-stable table in which $|T_x| > c_x$ for some agent x. Then either $T^{f(x)}$ is a super-stable table, or some agent in $T^{f(x)}$ has a short list.*

Proof The proof is similar to the argument in Lemma 5.3.2. □

Lemma 5.3.4. *Let T be a super-stable table, and let $\mathcal{A}$ be a super-stable allocation such that $\mathcal{A} \subseteq T$. Suppose that $|T_x| > c_x$ for some agent x. If no agent in $T^{f(x)}$ has a short list then there is a super-stable allocation contained in $T^{f(x)}$.*

Proof By Lemma 5.3.3, $T^{f(x)}$ is a super-stable table. We construct an allocation $\mathcal{B}$ contained in $T^{f(x)}$ as follows. Suppose $\{p, q\} \in \mathcal{A}$. We consider two cases:

- Case (i): $\{p, q\} \in T^{f(x)}$. Place $\{p, q\}$ in $\mathcal{B}$.
- Case (ii): $\{p, q\} \notin T^{f(x)}$. Since $T^{f(x)}$ is a super-stable table, then either p prefers $l_{T^{f(x)}}(p)$ to q or q prefers $l_{T^{f(x)}}(q)$ to p. Suppose, without loss of generality, that p prefers $l_{T^{f(x)}}(p)$ to q. Then add $\{p, z\}$ to $\mathcal{B}$, where z is any agent such that

$\{p, z\} \notin \mathcal{A}$, $\{p, z\} \notin \mathcal{B}$, $p \in F_{T^{f(x)}}(z)$, and z prefers at least one of his partners in $\mathcal{A}$ to any member of $F_{T^{f(x)}}(z)$.

We prove that, for every pair $\{p, q\}$ in Case (ii) above, there is an agent z satisfying the requirements.

Suppose first that $|T_p^{f(x)}| < c_p$. Then $|T_p^1| < c_p$, otherwise p violates the second defining property of a super-stable table. But then the deletion of $\{p, q\}$ when forming $T^{f(x)}$ from T must mean that $|T_p^{f(x)}| < |T_p| = |T_p^1| < c_p$, implying that p violates the second defining property of a super-stable table. Thus $|T_p^{f(x)}| \geq c_p$. But then, by the execution of the algorithm, p must be committed to, and hold a commitment from, c_p other agents. In particular, in $T^{f(x)}$ there must be an agent z, committed to p (so $p \in F_{T^{f(x)}}(z)$) such that $\{p, z\} \notin \mathcal{A}$ and p prefers z to q. Clearly $\{p, z\}$ does not block $\mathcal{A}$, so z must prefer every member of $p_{\mathcal{A}}(z)$ to p. But $\mathcal{A} \subseteq T$, so $p \notin F_T(z)$, and since $p \in F_{T^{f(x)}}(z)$, a pair $\{a, z\} \in \mathcal{A}$ must have been deleted during the execution of the algorithm, for some agent a who z prefers to p.

Further, for every agent r such that $\{p, r\} \in \mathcal{A}$, $\{p, r\} \notin T^{f(x)}$, there is a distinct agent satisfying the above properties. For, suppose not. Then $s = l_{T^{f(x)}}(p)$ is such that p prefers s to exactly k members of $L_T(p)$, and to at least $k + 1$ agents t such that $\{p, t\} \in \mathcal{A}$, for some $1 \leq k \leq c_p - 1$. Hence there exists an agent u such that $u \in L_T(p)$, $\{p, u\} \notin \mathcal{A}$, and p prefers u to at least one member of $p_{\mathcal{A}}(p)$. But, since $p \in F_T(u)$, and $\mathcal{A} \subseteq T$, u prefers p to at least one member of $p_{\mathcal{A}}(u)$, implying that $\{p, u\}$ blocks $\mathcal{A}$, a contradiction. The result follows.

We now show that each agent has the same number of partners in $\mathcal{B}$ as in $\mathcal{A}$. We start by showing that no agent p is matched with more than $min(c_p, |T_p^1|)$ partners in $\mathcal{B}$. Then, by simple counting and the preceding paragraph, every agent must have the same number of partners in $\mathcal{B}$ as in $\mathcal{A}$. Clearly every agent p with $|T_p^1| \leq c_p$ cannot be matched with more than $|T_p^1|$ partners in $\mathcal{B}$, by Lemma 5.2.5. So suppose some agent z with $|T_z^1| \geq c_z$ has $c_z + 1$ or more partners in $\mathcal{B}$. It follows that z must have had k of his partners in $\mathcal{A}$ deleted when forming $T^{f(x)}$, while gaining at least $k + 1 > 0$ new partners when playing the role of z in Case (ii). Suppose that $k = c_z$. Then all z's partners in $\mathcal{A}$ have been deleted, and since each of z's $k + 1 > c_z$ new partners appears in $F_{T^{f(x)}}(z)$, we have $|F_{T^{f(x)}}(z)| > c_z$, a contradiction. Hence $k < c_z$. It follows that z must have at least one partner in $\mathcal{A}$

such that he prefers all members of $F_{T^{f(x)}}(z)$, including each of his new partners, to this partner. But each of his new partners prefers z to at least one of their partners in $\mathcal{A}$ (namely q). It follows that z forms a blocking pair for $\mathcal{A}$ with each of these new partners, a contradiction. Hence each agent has the same number of partners in $\mathcal{B}$ as in $\mathcal{A}$, and in particular $\mathcal{B}$ is an allocation.

It remains to prove that $\mathcal{B}$ is super-stable. Suppose not. So there exists a pair $\{r, s\}$, say, which blocks $\mathcal{B}$. If both r and s have the same set of partners in $\mathcal{A}$ and $\mathcal{B}$, or if both prefer their least preferred partners in $\mathcal{B}$ to their least preferred partner in $\mathcal{A}$, or are indifferent between them, then $\{r, s\}$ blocks $\mathcal{A}$, a contradiction. Hence one of r, s must have at least one partner in $\mathcal{B}$ to whom they strictly prefer all their partners in $\mathcal{A}$. Suppose, without loss of generality, that r prefers all his partners in $\mathcal{A}$ to his least favoured partners in $\mathcal{B}$, one of whom is t, say. Then r must have been matched with t by Case (ii), and r must have played the role of z. But then $t \in F_{T^{f(x)}}(r)$, and since t is one of r's least favoured partners in $\mathcal{B}$, $F_{T^{f(x)}}(r)$ is the set of r's partners in $\mathcal{B}$. Since s is not a partner of r in $\mathcal{B}$, $s \notin F_{T^{f(x)}}(r)$, and since r must prefer s to at least one of his partners in $\mathcal{B}$ for $\{r, s\}$ to block $\mathcal{B}$, it follows that $\{r, s\} \notin T^{f(x)}$. But $T^{f(x)}$ is a super-stable table, so either s must strictly prefer $l_{T^{f(x)}}(s)$ to r, or r must strictly prefer $l_{T^{f(x)}}(r)$ to s. Either way, $\{r, s\}$ cannot block $\mathcal{B}$, a contradiction. Hence $\mathcal{B}$ is a super-stable allocation. □

We are now in a position to describe phase 2 of the algorithm. Set T to be T^1, the phase 1 table. If there is some agent x with $|T_x| > c_x$ we form $T^{f(x)}$ and $T^{l(x)}$. If no agent in $T^{f(x)}$ has a short list we set T to be $T^{f(x)}$. Otherwise, if no agent in $T^{l(x)}$ has a short list we set T to be $T^{l(x)}$. Otherwise (i.e., if both $T^{f(x)}$ and $T^{l(x)}$ contain an agent who has a short list) no super-stable allocation exists for the instance. We continue in this way until there is no agent x with $|T_x| > c_x$, at which point T specifies a super-stable allocation. Phase 2 of algorithm SFT-super is described in Figure 5.2.

The following theorem proves that Algorithm SFT-super is correct.

Theorem 5.3.5. *For a given instance of SFT, Algorithm SFT-super determines whether a super-stable allocation exists, and if so finds such an allocation.*

Proof Let I be an instance of SFT. Suppose I admits a super-stable allocation. If phase 1 terminates because $\sum_{p \in P} \min(c_p, |T_p^1|)$ is odd or because some replete agent p_i has $|T_{p_i}^1| < c_{p_i}$ then no super-stable allocation exists for the instance, by Corollary 5.2.8

$T :=$ phase 1 table;
while $|T_x| > c_x$ for some agent x {
delete from T $\{z, y\}$ from T for each $z \in f_T(x)$ and for each y in
T_z such that z prefers x to y, or is indifferent
between them, and apply main loop of phase 1 to obtain $T^{f(x)}$;
delete $\{x, y\}$ from T for all $y \in l_T(x)$ and apply main loop of
phase 1 to obtain $T^{l(x)}$;
if no agent in $T^{f(x)}$ has a short list
$T := T^{f(x)}$;
else if no agent in $T^{l(x)}$ has a short list
$T := T^{l(x)}$;
else {
no super-stable allocation exists;
halt; }}
output the super-stable allocation specified by T;

Figure 5.2: Phase 2 of Algorithm SFT-super

and Lemma 5.2.4 respectively, a contradiction. Hence we must enter phase 2. The phase 1 table, which is a super-stable table, must contain a super-stable allocation, by Lemma 5.2.3 (i). By Lemmas 5.3.2, 5.3.3 and 5.3.4, every super-stable table produced by the algorithm must contain a super-stable allocation, and so we must eventually reach a super-stable table T in which every agent p has a list of length $min(c_p, |T_p^1|)$, at which point T, which specifies a super-stable allocation by Lemma 5.3.1 (ii), is output.

Conversely, suppose I does not admit a super-stable allocation. If phase 1 does not terminate because $\sum_{p \in P} \min(c_p, |T_p^1|)$ is odd or because some replete agent p_i has $|T_{p_i}^1| < c_{p_i}$, then phase 2 will start. Since there is no super-stable allocation there cannot be a super-stable table in which every agent x has a list of length $min(c_x, |T_x^1|)$, since such would specify a super-stable allocation, by Lemma 5.3.1 (ii). It follows that every super-stable table has some agent x with a list longer than c_x. Since at least one entry is deleted from the preference lists in every loop of phase 2 there must eventually be a point at which both $T^{f(x)}$ and $T^{l(x)}$ contain an agent who has a short list, and so the algorithm will report that no super-stable allocation exists. □

Example 5.3.1. An example instance is displayed in Figure 5.3. The agents are numbered from 1 to 10, and for each agent i, i's preference list takes the form $i : (c_i)\ P_i$.

1:(2)	(2 10) 9 3 (6 7) 5 8	1:(2)	(2 10) 9 3 (6 7) 5
2:(2)	9 6 3 7 (1 4 5)	2:(2)	6 3 7 (1 4 5)
3:(2)	9 4 1 10 2 (5 7)	3:(2)	4 1 10 2 (5 7)
4:(2)	(5 8) 7 9 (2 3 10)	4:(2)	(5 8) 7 9 (2 3 10)
5:(2)	(1 2) 6 8 (3 4) 9	5:(2)	(1 2) 6 8 (3 4)
6:(2)	7 9 8 5 (1 2 10)	6:(2)	7 9 8 5 (1 2 10)
7:(1)	3 9 1 (4 8) 2 6	7:(1)	3 9 1 (4 8) 2 6
8:(1)	5 9 10 6 7 4 1	8:(1)	5 9 10 6 7 4
9:(1)	4 (7 10) 8 1 6 (2 3 5)	9:(1)	4 (7 10) 8 1 6
10:(1)	6 3 (4 9) 8 1	10:(1)	6 3 (4 9) 8 1
	Initial preference lists		Phase 1 table

Figure 5.3: The preference lists for an example SFT instance

During phase 1, the pair $\{1, 8\}$ is deleted when agent 4 becomes committed to agent 8, because agent 1 is then dominated in agent 8's list. The pairs $\{2, 9\}$, $\{3, 9\}$ and $\{5, 9\}$ are deleted, when agent 3 becomes committed to agent 9, because then agents 2, 3 and 5 are all tie-dominated in agent 9's list. These are the only pairs deleted during phase 1. The phase 1 table is displayed in Figure 5.3. Since, at the end of phase 1, every agent holds a number of commitments equal to his capacity and the sum of the capacities is even, phase 2 commences. Let T be the phase 1 table. Since $|T_1| > c_1 = 2$, we can form $T^{f(1)}$ and $T^{l(1)}$. For $T^{f(1)}$, we delete the pairs $\{2, 1\}$, $\{2, 4\}$, $\{2, 5\}$ and $\{10, 1\}$. It can then be verified that the application of the main loop of phase 1 produces the preference lists displayed for $T^{f(1)}$ in Figure 5.4. Since there are agents with short lists in $T^{f(1)}$ we must form $T^{l(1)}$.

For $T^{l(1)}$, we delete the pair $\{1, 5\}$. It can then be verified that the application of the main loop of phase 1 produces the preference lists displayed for $T^{l(1)}$ in Figure 5.4. Since no agent i has $|T_i^{l(1)}| > c_i$ and there are no agents with short lists in $T^{l(1)}$, $T^{l(1)}$ specifies a super-stable allocation, a fact which is easily verified.

1:(2)	3 7	1:(2)	2 3
2:(2)		2:(2)	(1 5)
3:(2)	1	3:(2)	1 10
4:(2)	9	4:(2)	7 9
5:(2)	6 8	5:(2)	2 6
6:(2)	5	6:(2)	8 5
7:(1)	1	7:(1)	4
8:(1)	5	8:(1)	6
9:(1)	4	9:(1)	4
10:(1)		10:(1)	3
$T^{f(1)}$		$T^{l(1)}$	

Figure 5.4: $T^{f(1)}$ and $T^{l(1)}$

5.4 Implementation and analysis of Algorithm SFT-super

We show that, for an instance of SFT involving a mutually acceptable pairs, Algorithm SFT-super can be implemented to run in $O(a)$ time.

In phase 1 of the algorithm, we adopt a strategy very similar to that described in Section 4.4. The only difference is ascertaining exactly which elements to delete. At the point at which an agent p_i receives a c_{p_i}th commitment it is simple to ascertain which element in his preference list represents the least preferred of the c_{p_i} agents from whom p_i has received a commitment. Say this agent is p_j. As every agent strictly succeeding p_j is deleted, the tie containing p_j will mark the end of p_i's list. When p_i next receives a commitment, there are two cases to consider. Firstly, suppose that this new commitment is from an agent tied in p_i's list with p_j. Backtracking up p_i's list from p_j to the first agent not in the tie containing p_j will identify which elements need to be deleted from the list, and p_i is again waiting to receive a c_{p_i}th commitment. Secondly, suppose that p_i prefers the agent from whom he has received this new commitment to p_j. Then backtracking up p_i's list from p_j to the first commitment encountered will identify the tie containing the c_{p_i}th commitment on the list, and it is simple to identify which elements to delete from the list. The number of backward steps is clearly bounded by a so, as with Algorithm SF (Chapter 4), phase 1

has worst-case complexity $O(a)$.

Phase 2 is somewhat more complex. We use the method of [26] to limit the amount of work done. Two possible courses of action must be considered in each loop iteration. In the worst case, we may require $\Omega(a)$ time in each iteration to calculate $T^{f(x)}$ and $T^{l(x)}$, and yet only make $O(1)$ deletions per iteration, so requiring $\Omega(a)$ iterations, giving overall complexity no better than $O(a^2)$. To solve this problem we make a copy T_b of the phase 1 table $T_a = T^1$, which takes $O(a)$ time. We then delete the pairs $\{x, y\}$, for all $y \in l_{T_b}(x)$, from T_b, and delete $\{z, w\}$ for each $z \in f_T(x)$ and for each w in T_z such that z prefers x to y, or is indifferent between them from T_a. We then apply the main loop of phase 1 to T_a and T_b, starting with the one that has had the greatest number of pairs deleted, and never allowing the number of deletions from the active table to exceed the number from the inactive table by more than 1. We maintain two stacks of deleted pairs, one for each of T_a and T_b. As soon as one of the phase 1 applications terminates we halt.

Suppose the application of phase 1 to T_a terminates first (a similar argument applies if the other application terminates first). If no agent in $T_a^{f(x)}$ has a short list, then we restore T_b from the stack, and then create another copy of $T_a^{f(x)}$ from T_b by applying the same deletions to T_b that were applied to T_a. Otherwise we restore T_a from the stack, and continue with the application of phase 1 to T_b. If this terminates with $T_b^{l(x)}$ containing an agent who has a short list then we exit the algorithm with the conclusion that no super-stable allocation exists for the instance. Otherwise we form a second copy of $T_b^{l(x)}$ from T_a by applying the same deletions to T_a that were applied to T_b.

If the algorithm has not terminated with the conclusion that no super-stable allocation exists then we have two copies of a super-stable table, and the time taken to obtain these two copies is $O(d)$, where d is the number of pairs deleted during this iteration of the main loop of phase 2. By repeating this process until phase 2 terminates we can see that the algorithm has overall time complexity $O(a)$.

5.5 Strong stability in SFT

We have extensively studied whether there is an efficient algorithm to determine if an instance of SFT admits a strongly stable allocation. In particular, we have attempted

to construct an algorithm, which we refer to as Algorithm SFT-strong, which works in a similar manner to Algorithm SFT-super. Thus far, however, we have not been successful. The problem is that, for the second phase of Algorithm SFT-strong, we have been unable to prove a result akin to Lemma 5.3.4. Indeed, it is tempting to conjecture that the problem of determining whether an instance of SFT admits a strongly stable allocation is NP-complete, given the nature of the problems encountered when attempting to prove such a result. Were this the case it would break the pattern so far established, as no variant of Stable Marriage with ties involving fewer than three sets of agents has been shown to be NP-complete for any problem not related to weak stability. We do, however, note that determining whether an instance of Stable Marriage with Partial orders admits a strongly stable matching is NP-complete [28].

Chapter 6

The structure of SMTI under super-stability

6.1 Introduction

Recall that the stable matchings for an instance of SMI are in one-to-one correspondence with the closed subsets of the rotation poset for that instance [24]. In this chapter we show how to construct, for the set of super-stable matchings in an instance of SMTI, a structure equivalent to the rotation poset for the set of stable matchings in an instance of SMI. We use this structure to solve efficiently a number of problems involving super-stable matchings. These include finding all the super-stable pairs, finding an egalitarian super-stable matching, finding a minimum regret super-stable matching, and generating all the super-stable matchings. The latter problem may have exponential time complexity in the worst case, as the number of matchings may be exponential in the size of the instance. However, we show that, after some pre-processing, we can generate successive matchings in time linear in the number of acceptable pairs.

To find the rotation poset for an instance of SMI, the rotations are identified directly from the preference lists. We studied the analogous approach for SMTI, but settled on a more amenable approach. Instead we start from the rotation poset for an instance of SMI derived from the instance of SMTI in question, and then we identify how to group the rotations together to construct so-called *meta-rotations*.

This chapter is structured as follows. In Section 6.2 we show how to construct a poset such that there is a one-to-one correspondence between the closed subsets of the poset and the super-stable matchings for an instance of SMTI. In Section 6.3 we show that we can use the elements of the poset to define a concept analogous to a rotation in an instance of SMI. In Section 6.4 we show that the poset can be constructed in time quadratic in the number of acceptable pairs in the instance, and in Sections 6.5, 6.6, 6.7, and 6.8, we show that that we can efficiently: find all the super-stable pairs; generate all the super-stable matchings; find an egalitarian super-stable matching; and find a minimum regret super-stable matching, respectively.

Note that in this chapter, where the context does not make clear whether a blocking pair is with respect to stability in an instance of SMI, or with respect to super-stability in an instance of SMTI, we refer to a *blocking pair* in the former case, and a *super-stable blocking pair* in the latter case.

6.2 A one-to-one correspondence for super-stability

Before we describe the construction of the poset we need to recall, and expand on, some of the results stated in Chapter 1. As noted previously, a given instance of SMTI may not admit a super-stable matching, but there is an $O(a)$ algorithm which determines if such a matching exists, and if so finds one [34]. The algorithm, Algorithm SUPER2, is displayed in Figure 1.3 of Section 1.7. After each proposal a number of pairs may be deleted from the preference lists. The following results are from [34].

Lemma 6.2.1. *No super-stable pair is ever deleted during an execution of Algorithm SUPER2.*

Theorem 6.2.2. *For a given instance of SMTI, Algorithm SUPER2 determines whether or not a super-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 every super-stable matching.*

As with SM, the matching obtained by Algorithm SUPER2 is *man-optimal*, and if the roles of the men and women are switched we obtain the *woman-optimal* super-stable matching.

Recall that the set of super-stable matchings for an instance I of SMTI forms a finite distributive lattice [51], and if a person p is indifferent between partners q and q' in super-stable matchings M and M' ($M \neq M'$), then $q = q'$ [38]. This latter result is important for part of what follows. We also state in full a result from [24] pertaining to SMI, which we use in Lemma 6.2.4:

Theorem 6.2.3. *If man m and woman w are partners in some stable matching S then*

1. *there is no stable matching S' in which both m and w have worse partners;*
2. *there is no stable matching S' in which both m and w have better partners.*

Here we seek a similar structure to the rotation poset for an instance of SMI to allow us to exploit the lattice property of the set of super-stable matchings for I. We refer to such a structure as the *meta-rotation poset*.

Let I be an instance of SMTI which admits a super-stable matching, and let M_0 and M_z be the man-optimal and woman-optimal super-stable matchings respectively (we assume $M_0 \neq M_z$, as otherwise there is only one super-stable matching for I).

Let $\mathcal{B}_1$ be the reduced preference lists obtained by applying Algorithm SUPER2 to I to obtain M_0, and let $\mathcal{B}_2$ be the reduced preference lists obtained by applying Algorithm SUPER2 to I to obtain M_z. We then define $\mathcal{B}$ to be the intersection of $\mathcal{B}_1$ with $\mathcal{B}_2$. Let $\mathcal{B}'$ denote the preference lists for an instance of SMI obtained from $\mathcal{B}$ by breaking the ties in some arbitrary but fixed way. We call $\mathcal{B}$ and $\mathcal{B}'$ the *base lists* and the *resolved base lists*, respectively. For purposes of reference we regard the members of each tie in $\mathcal{B}$ (and I) as being listed according to the way in which that tie was broken to form $\mathcal{B}'$. Note that in $\mathcal{B}$, and hence in $\mathcal{B}'$, every man (resp. woman) has a single woman (resp. man) at the head of his (resp. her) list, and no two men (resp. women) can have the same woman (resp. man) at the head of their lists. Note that $\mathcal{B}'$ can be considered to be an instance of SMI, which we shall also refer to as $\mathcal{B}'$ (and we also use $\mathcal{B}$ and I interchangeably).

By Lemma 6.2.1 every super-stable matching M for I is super-stable in $\mathcal{B}$, and is stable in $\mathcal{B}'$ (by the alternative definition of super-stability given in Section 1.6). Thus there is a stable set (see Section 1.4) corresponding to M. We call a stable set which has a derived matching that is super-stable in $\mathcal{B}$ a *super-stable set*. In particular, every super-stable matching for I must correspond to a closed subset of R', the rotation poset for $\mathcal{B}'$. Here

we show how to derive from R' a new poset such that the closed subsets of this new poset are in one-to-one correspondence with the super-stable matchings for I.

Algorithm POSET2 utilises a similar strategy to that of Algorithm POSET (Figure 1.1) as a first step to constructing a representation for the meta-rotation poset for I. The starting point is a directed acyclic graph representing R', which we shall also refer to as R'. The lists referred to in the algorithm are those in $\mathcal{B}$. A *proper* tie is a tie of length at least 2. The algorithm is broken down into two parts, displayed in Figures 6.1 and 6.2. The algorithm identifies pairs of rotations (ρ_1, ρ_2) such that, were ρ_1 eliminated but ρ_2 not eliminated, then a resulting matching would admit a super-stable blocking pair. Thus we must ensure that ρ_2 is eliminated before ρ_1 by adding the appropriate edge to the digraph.

A *stable man* (resp. *stable woman*) on a woman's (resp. man's) list is a man (resp. woman) who forms a stable pair with that woman (resp. man). Alternatively a stable man m on a woman w's list is such that $(m, w) \in \rho_{(m,w)}$ or $(m, w) \in M_z$, and similarly for a stable woman on a man's list (recall that $\rho_{(m,w)}$ denotes the rotation that deletes the pair (m, w)). In the algorithm the ties t are proper, whereas the ties t' may not be.

In the algorithm in Figure 6.1 the relationship between the various women on the preference list of man m is as follows:

$$m : ...w_1... \underbrace{(...w'...w_2...w...w_3)}_{t} ...$$

where

- w is the last stable woman in tie t;
- w_1 is the last stable woman before t;
- w_2 is the penultimate stable woman in t;
- w_3 is the last woman in t, and
- w' ranges across all women in t not equal to w.

Note that woman w_1 must exist by the super-stability of the man-optimal stable matching for $\mathcal{B}'$ and the fact that man m has a single woman at the head of his list. Also note that w_3 may coincide with w, and w_2 may not exist.

```
    let R' be a representation of the rotation poset for B';
    for each man m
        for each proper tie t on m's list containing at least one stable woman {
            let w be the last stable woman in t;
            let w_1 be the last stable woman preceding t;
            for each w' ≠ w in t {
                let t' be the tie on the list of w' containing m;
                let m_1 be the first stable man that w' does not prefer to m;
                if ρ_(m,w) precedes ρ_(m_1,w') in R'
10                  add edge (ρ_(m,w), ρ_(m,w_1)) to R';
                else {
                    if ρ_(m_1,w') does not precede ρ_(m,w_1) in R'
13                      add edge (ρ_(m_1,w'), ρ_(m,w_1)) to R';
                    if there are at least two stable women in t {
                        let w_2 be the penultimate stable woman in t;
16                      add edge (ρ_(m,w_2), ρ_(m,w_1)) to R'; }
                    let w_3 be the last woman in t;
                        if w_3 ≠ w
19                          add edge (ρ_(m,w_3), ρ_(m,w_1)) to R'; }}}
```

Figure 6.1: Algorithm POSET2: Constructing the meta-rotation poset for an instance of SMTI: first step: men's side

On the list of w', m_1 is the first stable man in tie t', or the first stable man after t' if there is no stable man in t'. Note that man m_1 must exist by the super-stability of the man-optimal stable matching for $\mathcal{B}'$.

In the algorithm in Figure 6.2 the relationship between the various men on the preference list of woman w is as follows:

$$w : ... \underbrace{(...m_2...m'...m...)}_{t} ...m_1...$$

where

- m is the last stable man in tie t;

for each woman w
  for each proper tie t on w's list containing at least one stable man {
    let m be the last stable man in t;
    let m_1 be the first stable man after t;
    for each $m' \neq m$ in t {
      let t' be the tie on the list of m' containing w;
      let w_1 be the last stable woman preceding t';
      let m_2 be the first stable man in t;
      if $\rho_{(m',w_1)}$ precedes $\rho_{(m_1,w)}$ in R'
30   add edge $(\rho_{(m_2,w)}, \rho_{(m_1,w)})$ to R';
      else {
        if $\rho_{(m,w)}$ does not precede $\rho_{(m',w_1)}$ in R'
33   add edge $(\rho_{(m,w)}, \rho_{(m',w_1)})$ to R';
        if there are at least two stable men in t
35   add edge $\rho_{(m_2,w)}, \rho_{(m,w)})$ to R'; }}}

Figure 6.2: Algorithm POSET2 cont.: Constructing the meta-rotation poset for an instance of SMTI: first step: women's side

- m_1 is the first stable man after t;
- m_2 is the first stable man in t, and
- m' ranges across all men in t not equal to m.

Note that m_2 may coincide with m.

The relationship between the various women on the list of m' is as follows:

$$m' : ...w_1... \underbrace{(...w...)}_{t'} ...$$

where w_1 is the last stable woman preceding tie t'.

When Algorithm POSET2 has terminated, the amended digraph R' will consist of a number of strongly connected components. We construct a new digraph R from R' as follows: create a node n_C for each strongly connected component C in R'; if there is an edge

from u to v in R', where u and v are in different strongly connected components C and D respectively, we add an edge from n_C to n_D in R if such an edge does not already exist. We show, via a number of lemmas, that the closed subsets of R are in one-to-one correspondence with the super-stable matchings in $\mathcal{B}$.

Lemma 6.2.4. *Let t be a tie on the list of a person y, let $x_1, x_2, ..., x_k$ be the stable partners of y in $\mathcal{B}'$ who are in t, and suppose that t is broken in $\mathcal{B}'$ so that $x_1, x_2, ..., x_k$ appear in that order. Then none of $x_1, x_2, ..., x_{k-1}$ can be a super-stable partner of y in $\mathcal{B}$.*

Proof A super-stable matching in $\mathcal{B}$ is a stable matching in $\mathcal{B}'$. Consider a stable matching M in $\mathcal{B}'$ containing the pair $\{y, x_i\}$ for some i $(1 \leq i \leq k-1)$. In $\mathcal{B}'$, x_k prefers y to his partner in M, by Theorem 6.2.3. Then, in $\mathcal{B}$, x_k either prefers y to that partner or is indifferent between them. Since y is indifferent between x_k and x_i in $\mathcal{B}$ it follows that $\{y, x_k\}$ is a super-stable blocking pair for M. □

Lemma 6.2.5. *Each super-stable matching in I corresponds to a closed subset of R.*

Proof Suppose, for a contradiction, that some super-stable matching in I does not correspond to a closed subset of R. Then there must be a super-stable set $\mathcal{U}$ which is obtainable by eliminating a rotation u but not eliminating a rotation v, where an edge from v to u was added during the execution of the algorithm. There are a number of cases to consider. The first four relate to Figure 6.1, and the last three relate to Figure 6.2.

Case 1 (line 10): The edge (v, u) was added as $(\rho_{(m,w)}, \rho_{(m,w_1)})$. Recall that w_1 is the last stable woman preceding the tie t containing w on the list of m. In $\mathcal{U}$, $\rho_{(m,w_1)}$ has been eliminated, but $\rho_{(m,w)}$ has not, so m is matched in $M_{\mathcal{U}}$ with some woman w'' such that he is indifferent between w and w''. By Lemma 6.2.4 $w = w''$. For the edge in question to have been added there must be a woman $w'(\neq w) \in t$ such that w' has a tie t' on her list containing m, and there is a rotation $\rho_{(m_1,w')}$ which, at this point in the algorithm, is preceded by $\rho_{(m,w)}$, where m_1 is the first stable man in t' or, if such does not exist, the first stable man after t'. Since $\rho_{(m,w)}$ has not been eliminated in $\mathcal{U}$ neither has $\rho_{(m_1,w')}$, so w' must be partnered in $M_{\mathcal{U}}$ by a man m' such that w' prefers m to m' or is indifferent between them, and $m \neq m'$ (since $(m, w) \in M_{\mathcal{U}}$). Then (m, w') is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

Case 2 (line 13): The edge (v, u) was added as $(\rho_{(m_1,w')}, \rho_{(m,w_1)})$, where the agents are as defined in Case 1. In $\mathcal{U}$, $\rho_{(m,w_1)}$ has been eliminated, but $\rho_{(m_1,w')}$ has not. Then w'

is matched in $M_{\mathcal{U}}$ with a man m' such that w' prefers m to m', or is indifferent between them. Further, m is matched in $M_{\mathcal{U}}$ with a woman w such that m prefers w' to w, or is indifferent between them. Finally $(m, w') \notin M_{\mathcal{U}}$, by Lemma 6.2.4. Hence (m, w') is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

Case 3 (line 16): The edge (v, u) was added as $(\rho_{(m,w_2)}, \rho_{(m,w_1)})$, where w_2 is the penultimate stable woman in the tie on the list of m containing w. Since $\rho_{(m,w_1)}$ has been eliminated from $\mathcal{U}$ and $\rho_{(m,w_2)}$ has not, m is matched in $M_{\mathcal{U}}$ with some woman w_m such that m is indifferent between w and w_m, and $w \neq w_m$. Further $\rho_{(m,w_2)}$ must precede $\rho_{(m,w)}$, so (m, w) has not been deleted. Hence w prefers m to $l_{\mathcal{U}}(w)$, her partner in $M_{\mathcal{U}}$, or is indifferent between them (by the second defining property of a stable set). Thus (m, w) is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

Case 4 (line 20): The edge (v, u) was added as $(\rho_{(m,w_3)}, \rho_{(m,w_1)})$, where w_3 is the final woman in the tie containing w on the list of m, and is non-stable. By a similar argument to that in Case 3, (m, w_3) is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

Case 5 (line 30): The edge (v, u) was added as $(\rho_{(m_2,w)}, \rho_{(m_1,w)})$. So there is a woman w with a tie t on her list in which m_2 is the first stable man and m is the last stable man, and m_1 is the first stable man after t. Since $\rho_{(m_1,w)}$ has been eliminated from $\mathcal{U}$ but $\rho_{(m_2,w)}$ has not, w is matched in $M_{\mathcal{U}}$ with some man m'' from t. By Lemma 6.2.4 $m = m''$. For the edge in question to have been added there must be a man $m'(\neq m) \in t$ with a tie t' on his list containing w, and with w_1 the last stable woman preceding t'. Further the algorithm requires that $\rho_{(m',w_1)}$ precedes $\rho_{(m_1,w)}$. Since $\rho_{(m_1,w)}$ has been eliminated from $\mathcal{U}$, so has $\rho_{(m',w_1)}$, so m' is matched in $M_{\mathcal{U}}$ with a woman w' such that m' prefers w to w' or is indifferent between them (and $w \neq w'$). But then (m', w) is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

Case 6 (line 33): The edge (v, u) was added as $(\rho_{(m,w)}, \rho_{(m',w_1)})$, where the agents are as defined in Case 5. Since $\rho_{(m',w_1)}$ has been eliminated from $\mathcal{U}$, m' is matched in $M_{\mathcal{U}}$ with a woman w' such that m' prefers w to w' or is indifferent between them. Since $\rho_{(m,w)}$ has not been eliminated, w is matched in $M_{\mathcal{U}}$ with a man m'' such that w prefers m' to m'' or is indifferent between them. Finally $(m', w) \notin M_{\mathcal{U}}$, by the preceding case. Hence (m', w) is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

Case 7 (line 35): The edge (v, u) was added as $(\rho_{(m_2,w)}, \rho_{(m,w)})$, where the agents are as

defined in Case 5. Since $\rho_{(m,w)}$ has been eliminated and $\rho_{(m_2,w)}$ has not, w is matched in $M_{\mathcal{U}}$ with a man m_w such that w is indifferent between m and m_w and $m_w \neq m$. Since $\rho_{(m,w)}$ has not been eliminated, m must be matched in $M_{\mathcal{U}}$ to a woman w_m such that m prefers w to w_m or is indifferent between them. But then (m, w) is a super-stable blocking pair for $M_{\mathcal{U}}$, a contradiction.

It follows that there is no super-stable set in which u has been eliminated but v has not, and hence that each super-stable matching in I corresponds to a closed subset of R. □

Lemma 6.2.6. *Each closed subset of R corresponds to a unique super-stable matching in I.*

Proof Let S be a closed subset of R. Let $\mathcal{S}$ be the super-stable set corresponding to M_0. Let $\mathcal{T}$ be the stable set produced when the set of rotations equivalent to S is eliminated from $\mathcal{S}$, and suppose $\mathcal{T}$ is not super-stable. Then there is a super-stable blocking pair (m', w) for $M_{\mathcal{T}}$. Hence m' prefers w to his partner in $M_{\mathcal{T}}$, w' say, or is indifferent between them.

Case 1: Suppose m' prefers w to w'. Since (m', w) is not a blocking pair for $M_{\mathcal{T}}$ in $\mathcal{B}'$ it follows that w prefers her partner in $M_{\mathcal{T}}$, m say, to m' in $\mathcal{B}'$, and hence w prefers m to m' in $\mathcal{B}$, or is indifferent between them. Since the former case contradicts the fact that (m', w) is a super-stable blocking pair for $M_{\mathcal{T}}$, it follows that w is indifferent between m' and m in $\mathcal{B}$.

By the construction of R, m must be the last stable man in the tie t containing m and m' on the list of w (see line 35 of Algorithm POSET2). Consider the point in the algorithm when m' plays the role of m' (and m and w play the roles of m and w respectively), m_1 is the first stable man after t, and w_1 is the last stable woman preceding the tie containing w on the list of m'.

If $\rho_{(m',w_1)}$ precedes $\rho_{(m_1,w)}$, then $\rho_{(m_2,w)}$ is made to precede $\rho_{(m_1,w)}$, where m_2 is the first stable man in t. Thus, when $\rho_{(m_1,w)}$ is eliminated so is $\rho_{(m_2,w)}$, and, by the structure of the rotation poset for $\mathcal{B}'$, which R is built upon, so is every pair (m_s, w), where m_s is a stable man in t. Thus, (m, w) can not possibly be in $M_{\mathcal{T}}$, a contradiction. So $\rho_{(m',w_1)}$ does not precede $\rho_{(m_1,w)}$.

Since $\rho_{(m',w_1)}$ does not precede $\rho_{(m_1,w)}$, by the construction of R $\rho_{(m,w)}$ precedes $\rho_{(m',w_1)}$.

Since $(m, w) \in M_{\mathcal{T}}$, (m', w_1) has not been eliminated from $\mathcal{T}$, so m' must prefer w', his partner in $M_{\mathcal{T}}$, to w_1 or be indifferent between them. But m' prefers w_1 to w, so (m', w) cannot block $M_{\mathcal{T}}$. This contradiction establishes the super-stability of $M_{\mathcal{T}}$ in this case.

Case 2: Suppose m' is indifferent between w and w'. By the construction of R, w' must be the last stable woman in the tie t containing w and w' on the list of m'. Consider the point in the algorithm when w plays the role of w' (and m' and w' play the roles of m and w respectively), m_1 is the first stable man in, or, if such does not exist, after the tie on the list of w containing m' and w_1 is the last stable woman preceding t (note that we refer to the agents as they appear in this proof, not as they appear in the algorithm).

If $\rho_{(m',w')}$ precedes $\rho_{(m_1,w)}$ then, by the algorithm, $\rho_{(m',w')}$ precedes $\rho_{(m',w_1)}$. Thus, (m', w') must be deleted before the rotation $\rho_{(m',w_1)}$ is eliminated. Since m' prefers w_1 to w' it follows that (m', w') can not possibly be in $M_{\mathcal{T}}$, a contradiction. So $\rho_{(m',w')}$ does not precede $\rho_{(m_1,w)}$.

Since $\rho_{(m',w')}$ does not precede $\rho_{(m_1,w)}$, by the construction of R $\rho_{(m_1,w)}$ precedes $\rho_{(m',w_1)}$. Since $(m, w) \in M_{\mathcal{T}}$ and m' prefers w_1 to w, it follows that (m_1, w) has been deleted. But then w must be matched in $M_{\mathcal{T}}$ with a man she prefers to m', contradicting the fact that (m', w) is a super-stable blocking pair for $M_{\mathcal{T}}$. Thus $M_{\mathcal{T}}$ is super-stable.

Finally note that S corresponds to a unique closed subset of R', which corresponds to a unique stable matching for $\mathcal{B}'$. It follows that each closed subset of R corresponds to a unique super-stable matching in I. □

The last two lemmas lead to the following key theorem :

Theorem 6.2.7. *The closed subsets of R are in one-to-one correspondence with the super-stable matchings of I.*

Before we give an example to illustrate the execution of Algorithm POSET2 we make a formal definition of a meta-rotation.

6.3 Meta-rotations

Each node in R corresponds to one or more of the rotations in $\mathcal{B}'$, and is a minimal difference between super-stable matchings. We call each collection of rotations a *meta-*

rotation, and hence R is a representation of the *meta-rotation poset*. To *eliminate* a meta-rotation is to eliminate each of its constituent rotations in turn. A meta-rotation is *exposed* in a super-stable set if all the preceding meta-rotations in R have been eliminated. A pair (m, w) is *in* a meta-rotation Φ if it is in one of the rotations, ρ say, in Φ, and no rotation in Φ that precedes ρ has a pair involving m in it.

Recall that the rotations in question are the rotations for $\mathcal{B}'$, and $\mathcal{B}'$ is dependent on the order in which the ties were broken. Once the precedences between the meta-rotations have been established, it is possible to abstract away from the underlying set of rotations by representing a meta-rotation as a set of triples or cycles which show the change in partner for any man who changes partner when the meta-rotation is eliminated.

Representing these changes as a set of triples is similar to the transformation construct employed in [5]. Let M be a stable matching and let S be a set of rotations that can be eliminated in succession from the stable set from which M is derived. Let M' be the stable matching derived from the resulting stable set. Then a *transformation* is a set Π of triples (m, w, w'), one for each man m who has different partners in M and M', such that $(m, w) \in M$, $(m, w') \in M'$. Letting M be the matching derived from a super-stable set $\mathcal{S}$, and letting Φ be a meta-rotation exposed in $\mathcal{S}$ we can see that Π is a representation of Φ which is independent of the particular underlying set of rotations. Using this representation, eliminating a meta-rotation $\Pi = (m_i, w_i^1, w_i^2)$ $(1 \leq i \leq k)$ from a super-stable set $\mathcal{S}$ means deleting all pairs (m, w_i^2) such that w_i^2 prefers m_i to m $(1 \leq i \leq k)$. Any pair (m, w) such that $(m, w, w') \in \Pi$ for any w' is *in* Φ.

Representing these changes as a set of cycles is similar, and more compact in terms of the amount of information to be written down. Using the symbols defined above, we represent the meta-rotation Φ as a set Π of cycles $C_j = (m_0^j, w_0^j), ..., (m_{s_j-1}^j, w_{s_j-1}^j)$ such that $(m_i^j, w_i^j) \in M$, $(m_i^j, w_{i+1}^j) \in M'$, where $i+1$ is taken modulo s_j, and s_j is the length of the jth cycle. Using this representation, let $\Pi = \bigcup_{j=1}^{t} (m_0^j, w_0^j), ..., (m_{s_j-1}^j, w_{s_j-1}^j)$ be a meta-rotation consisting of t cycles. Eliminating Π from a super-stable set $\mathcal{S}$ means deleting all pairs (m, w_{i+1}^j) such that w_{i+1}^j prefers m_i^j to m, where $i+1$ is taken modulo s_j $(0 \leq i \leq s_j - 1)$ $(1 \leq j \leq t)$. Any pair (m, w) such that $(m, w) \in C_j$ for any $1 \leq j \leq t$ is *in* Φ.

We now give an example to illustrate the application of Algorithm POSET2.

Example 6.3.1. In Figure 6.4 we present an instance I of SMT, and in Figure 6.5 we present an instance I' of SM which can be considered to be the resolved base lists for I. In I' there are 10 rotations, which we denote by $\rho_1, ..., \rho_{10}$. For each of the first 7 entries w_i or w^i on a man m's list, ρ_i is the rotation the elimination of which causes the pair (m, w) to be deleted, and similarly for the last seven entries on a woman's list. Those entries with subscripts denote stable men or women, those with superscripts denote non-stable men or women. For example the stable pairs $(1, 1)$ and $(2, 2)$ and the non-stable pairs $(6, 1)$ and $(8, 1)$ are deleted when $\rho_1 = \{(1, 1)(2, 2)\}$ is eliminated. The rotation poset for I' is displayed in Figure 6.3. It can be verified that there are 14 closed subsets of this poset, and so there are 14 stable matchings for I'.

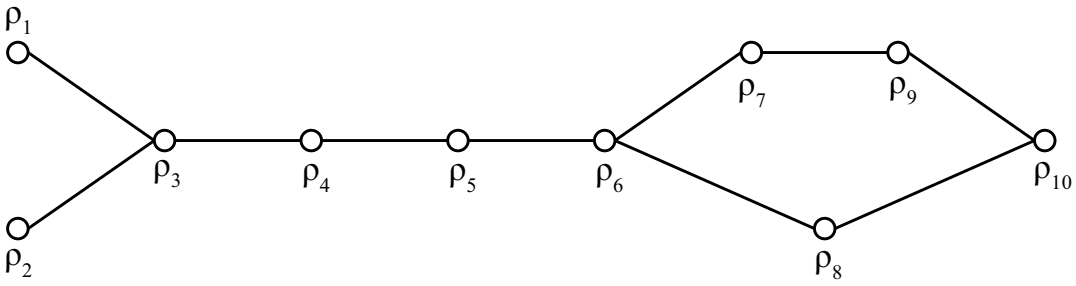

Figure 6.3: The rotation poset for I'

1:	1_1	$(2_4$	7^3	8_6	3^3	$4^4)$	6_{10}	5
2:	2_1	1_5	8^4	4_8	5_{10}	7^7	6^5	3
3:	3_2	4_4	$(6^3$	$5^2)$	$(2_6$	$7^3)$	8_9	1
4:	4_2	5_5	6_6	7_7	2^4	3^2	1_9	8
5:	5_2	3_3	4^2	6_5	8^4	1_7	2^4	7
6:	6_3	1_1	7_6	$(2_8$	8^6	3_{10}	$5^8)$	4
7:	7_3	3_8	2^6	8^6	4_{10}	1^9	5^8	6
8:	8_4	1^1	4_5	6^3	7^3	5_8	3^3	2

The men's preference lists

1:	3	7^9	4_9	5_7	2_5	6^1	8^1	1_1
2:	8	6_8	7^6	3_6	5^4	4^4	1_4	2_1
3:	2	6_{10}	7_8	8^3	1^3	5_3	4^2	3_2
4:	6	7_{10}	$(2_8$	8_5	$1^4)$	3_4	5^2	4_2
5:	1	2_{10}	6^8	7^8	$(8_8$	4_5	$3^2)$	5_2
6:	7	1_{10}	4_6	2^5	5_5	8^3	3^3	6_3
7:	5	2^7	$(4_7$	6_6	$3^3)$	1^3	8^3	7_3
8:	4	$(3_9$	$6^6)$	7^6	1_6	5^4	2^4	8_4

The women's preference lists

Figure 6.4: An instance I of SMT

We show that the derivation of the meta-rotation poset for I from the rotation poset for I' requires the addition of at least one edge of each of the various types, and also explain why these additions are necessary.

1:	1_1	2_4	7^3	8_6	3^3	4^4	6_{10}	5
2:	2_1	1_5	8^4	4_8	5_{10}	7^7	6^5	3
3:	3_2	4_4	6^3	5^2	2_6	7^3	8_9	1
4:	4_2	5_5	6_6	7_7	2^4	3^2	1_9	8
5:	5_2	3_3	4^2	6_5	8^4	1_7	2^4	7
6:	6_3	1_1	7_6	2_8	8^6	3_{10}	5^8	4
7:	7_3	3_8	2^6	8^6	4_{10}	1^9	5^8	6
8:	8_4	1^1	4_5	6^3	7^3	5_8	3^3	2

The men's preference lists

1:	3	7^9	4_9	5_7	2_5	6^1	8^1	1_1
2:	8	6_8	7^6	3_6	5^4	4^4	1_4	2_1
3:	2	6_{10}	7_8	8^3	1^3	5_3	4^2	3_2
4:	6	7_{10}	2_8	8_5	1^4	3_4	5^2	4_2
5:	1	2_{10}	6^8	7^8	8_8	4_5	3^2	5_2
6:	7	1_{10}	4_6	2^5	5_5	8^3	3^3	6_3
7:	5	2^7	4_7	6_6	3^3	1^3	8^3	7_3
8:	4	3_9	6^6	7^6	1_6	5^4	2^4	8_4

The women's preference lists

Figure 6.5: I', the resolved base lists for I

First consider the second tie on the list of man 3, and in particular consider the effect of the first part of Algorithm POSET2, in Figure 6.1, on this tie, when woman 7 is w'. Then man 3 is m, woman 2 is w, woman 4 is w_1, and man 4 is m_1. We can see from the lists that $\rho_6 = \rho_{(m,w)}$, $\rho_7 = \rho_{(m_1,w')}$, and $\rho_4 = \rho_{(m,w_1)}$. So $\rho_{(m,w)}$ precedes $\rho_{(m_1,w')}$, and the algorithm adds the edge (ρ_6, ρ_4) to the digraph. Suppose that this edge was not added to the digraph. Then ρ_4 could be eliminated before ρ_6, and so we could have a matching containing both $(3, 2)$ and $(6, 7)$. However, $(3, 7)$ would block this matching. Thus we need the edge (ρ_6, ρ_4) in the digraph.

Now consider the tie on the list of man 6, and in particular consider the effect of the first part of Algorithm POSET2, in Figure 6.1, on this tie, when woman 8 is w'. Then man 6 is m, woman 3 is w, woman 7 is w_1, man 3 is m_1, woman 2 is w_2, and woman 5 is w_3. Further, $\rho_{10} = \rho_{(m,w)}$, $\rho_9 = \rho_{(m_1,w')}$, $\rho_6 = \rho_{(m,w_1)}$, $\rho_8 = \rho_{(m,w_2)} = \rho_{(m,w_3)}$. Since $\rho_{(m,w)}$ does not precede $\rho_{(m_1,w')}$, the algorithm enters the else statement. Then $\rho_{(m_1,w')}$ does not precede $\rho_{(m,w_1)}$, so the algorithm adds the edge (ρ_9, ρ_6) to the digraph. Since there are at least two stable women, namely women 2 and 3, in the tie on man 6's list, the algorithm adds the edge (ρ_8, ρ_6), to the digraph. Further, w is not the last woman in the tie, so the algorithm again adds (ρ_8, ρ_6) to the digraph. We consider the consequence if these edges had not been added. Suppose the first edge, (ρ_9, ρ_6) was not added to the digraph. Then ρ_6 could be eliminated before ρ_9, and so we could have a matching containing $(6, 2)$ and $(3, 8)$. However, $(6, 8)$ would block this matching, a contradiction. Thus we need the edge (ρ_9, ρ_6) in the digraph. Now suppose the second edge, (ρ_8, ρ_6), was not added to the digraph. Then ρ_6 could be eliminated before ρ_8, and so we could have a matching, M say, containing $(6, 2)$. But the rotation which deletes $(6, 3)$ must be preceded by that which deletes $(6, 2)$. For, women 2 and 3 are both stable women in the same tie on man 6's list, but man 6 prefers woman 2 to woman

3 in the resolved base lists. Thus woman 3 must be matched in M with a man she does not prefer to man 6, and so $(6, 3)$ blocks M. Thus we need the edge (ρ_8, ρ_6) in the digraph. Finally suppose the third edge, (ρ_8, ρ_6), was not added to the digraph (so this edge is added twice, but we show that it needs adding for two different reasons). Then ρ_6 could be eliminated before ρ_8, and so we could conceivably have a matching, M say, containing $(6, 2)$ in which $\rho_8 = \rho_{(6,5)}$ has not been eliminated, so woman 5 has a partner she does not prefer to man 6. But then $(6, 5)$ would block this matching. Thus we again need the edge (ρ_8, ρ_6) in the digraph.

Now we switch our attention to the women's lists. First consider the tie on woman 4's list, and in particular consider the effect of the second part of Algorithm POSET2, in Figure 6.2, on this tie, when man 1 is m'. Then woman 4 is w, man 8 is m, man 3 is m_1, woman 1 is w_1, and man 2 is m_2. Further, $\rho_1 = \rho_{(m',w_1)}$, $\rho_4 = \rho_{(m_1,w)}$, and $\rho_8 = \rho_{(m_2,w)}$. Hence $\rho_{(m',w_1)}$ precedes $\rho_{(m_1,w)}$, so the algorithm adds the edge (ρ_8, ρ_4) to the digraph. Suppose that this edge was not added to the digraph. Then ρ_4 could be eliminated before ρ_8, and ρ_1 must have been eliminated before ρ_4. Thus we could have a matching containing either $(8, 4)$ or $(2, 4)$. But in any such matching, because ρ_1 has been eliminated, man 1 cannot prefer his partner to woman 4, and so $(1, 4)$ blocks the matching. Thus we need the edge (ρ_8, ρ_4) in the digraph.

Finally consider the tie on the list of woman 5, and in particular consider the effect of the second part of Algorithm POSET2, in Figure 6.2, on this tie, when man 3 is m'. Then woman 5 is w, man 4 is m, man 5 is m_1, woman 4 is w_1, and man 8 is m_2. Further, $\rho_4 = \rho_{(m',w_1)}$, $\rho_2 = \rho_{(m_1,w)}$, $\rho_5 = \rho_{(m,w)}$ and $\rho_8 = \rho_{(m_2,w)}$. Hence $\rho_{(m',w_1)}$ does not precede $\rho_{(m_1,w)}$, so the algorithm enters the else statement. Then $\rho_{(m,w)}$ does not precede $\rho_{(m',w_1)}$, so the algorithm adds the edge (ρ_5, ρ_4) to the digraph. Further, there are two stable men, namely man 8 and man 4, in the tie on woman 5's list, so the algorithm adds the edge (ρ_8, ρ_5) to the digraph. We consider the consequence if these edges had not been added. Suppose the first edge, (ρ_5, ρ_4), was not added to the digraph. Then ρ_4 could be eliminated before ρ_5, and so we could have a matching containing both $(4, 5)$ and $(3, 2)$. However, $(3, 5)$ would block this matching. Thus we need the edge (ρ_5, ρ_4) in the digraph. Now suppose the second edge, (ρ_8, ρ_5), was not added to the digraph. Then ρ_5 could be eliminated before ρ_8, and so we could have a matching, M say, containing $(8, 5)$. But man 4 cannot then have a partner better than woman 5 in M, and so $(4, 5)$ blocks M. Thus we need the edge (ρ_8, ρ_5) in the digraph.

It can be verified that, when the 6 edges above are added to the digraph R' representing the rotation poset for I', there is one strongly connected component consisting of more than one rotation, and it includes the rotations ρ_4 to ρ_9. Thus the meta-rotation poset consists of 5 meta-rotations, $\Phi_1, ..., \Phi_5$, such that $\Phi_1 = \rho_1$, $\Phi_2 = \rho_2$, $\Phi_3 = \rho_3$, $\Phi_5 = \rho_{10}$, and Φ_4 consists of the remaining 6 rotations. The meta-rotation poset for I is displayed in Figure 6.6. It can be verified that there are 7 closed subsets of the poset, and so there are 7 super-stable matchings for I, 7 less than the

number of stable matchings for I'.

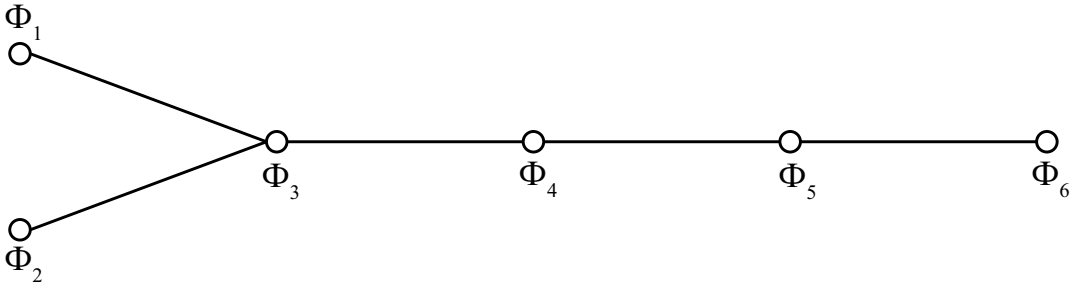

Figure 6.6: The meta-rotation poset for I

6.4 Implementation and analysis of Algorithm POSET2

Let n be the number of men matched in a super-stable matching in an instance I of SMTI, and hence the number of men in $\mathcal{B}'$, and let a be the number of acceptable pairs in I. Then $\mathcal{B}'$ can be constructed from I by two applications of Algorithm SUPER2, taking $O(a)$ time [34], and then breaking the ties arbitrarily, again taking $O(a)$ time. The rotation poset for $\mathcal{B}'$ can be constructed by Algorithm POSET in $O(a)$ time [15], and in so doing it is an easy matter to note which rotation each pair is deleted by, and annotate the preference lists accordingly. Since a rotation must contain at least two men, and a man can appear in at most a rotations, there are at most $a/2$ rotations in the rotation poset for $\mathcal{B}'$. Further, it can be seen from the algorithm in Figure 1.1 that the digraph representing the rotation poset cannot have more than a edges. We can therefore initiate a depth first search from each rotation ρ in the rotation poset for $\mathcal{B}'$ to create an adjacency matrix representing the transitive closure of R'. The total time for this is $O(a^2)$, since each search takes $O(a)$ time, and there are $O(a)$ searches. We can then verify if a rotation ρ_1 precedes a rotation ρ_2 in constant time.

Now we consider Algorithm POSET2, starting with the first part of the algorithm, displayed in Figure 6.1. It can be verified that the total number of iterations of the inner for loop is bounded by the number of acceptable pairs, and identifying each of the people who play the roles required in the algorithm can be achieved in constant time. We can determine in constant time which rotation deletes a given pair, because of the work

done when constructing the rotation poset for $\mathcal{B}'$. We can also determine in constant time whether a given rotation precedes a second given rotation, by looking at the adjacency matrix. Clearly updating the digraph when a new edge is added can also be achieved in constant time. Thus the overall time complexity of the first part of the algorithm is $O(a)$.

The case for the second part of Algorithm POSET2, displayed in Figure 6.2, is similar, and it can be verified that the time complexity of it is also $O(a)$.

Finally, we need to consider the complexity of taking the graph output by the algorithm and forming the new poset. We use as our starting point a representation R' of the rotation poset constructed by an algorithm similar to Algorithm POSET, and with identical running time, where R' has $O(a)$ edges, but additionally has the property that the outdegree of each node in R' is $O(n)$ (see [15]). Adding an edge to R' takes constant time, so, since the time complexity of each part of the algorithm is $O(a)$, it follows that we add $O(a)$ edges to R' during the execution of Algorithm POSET2, so the meta-rotation poset R has $O(a)$ edges. Finding all the strongly connected components in a directed graph $G = (V, E)$ takes $O(|V| + |E|)$ time (see, for example, [4, pps. 488-494]). For an SMI instance consisting of b acceptable pairs there are at most $\frac{b}{2}$ rotations, since each rotation consists of at least 2 pairs. Thus $|V|$ is $O(a)$, and we have previously established that R' has $O(a)$ edges. Hence finding all of the strongly connected components in R' takes $O(a)$ time. Since there are $O(a)$ edges in R', ensuring that the precedences between the strongly connected components are correct also takes $O(a)$ time in the worst case. Thus the overall time complexity of producing the new poset is $O(a^2)$. However, the only stage of the algorithm that is not $O(a)$ is finding the transitive closure of the rotation poset for $\mathcal{B}'$. If we could identify in $O(a)$ time, without requiring to find the transitive closure of the rotation poset for $\mathcal{B}'$, whether a given rotation precedes a second given rotation, then the algorithm could be implemented to run in $O(a)$ time.

6.5 The super-stable pairs

In this section we show that we can find all the super-stable pairs for an instance I of SMTI in $O(a^2)$ time.

Theorem 6.5.1. *A pair (m, w) is a super-stable pair if and only if it is in some meta-rotation, or it is in M_z.*

Proof Clearly a pair is super-stable if it is in M_z. Suppose (m, w) belongs to a meta-rotation ρ. Then there is a super-stable set, $\mathcal{S}$ say, in which ρ is exposed, and hence (m, w) is in $M_{\mathcal{S}}$, and so is super-stable.

Now suppose (m, w) is a super-stable pair. If (m, w) appears in M_z there is nothing to prove. Suppose (m, w) does not appear in M_z. Then $(m, w) \in M$ for some super-stable matching $M \neq M_z$, and m must have a partner $w' \neq w$ in M_z. Since M must dominate M_z in $\mathcal{B}'$, and an agent p can have at most one super-stable partner from any given tie on their list, m must prefer w to w' in I. But then there must be a meta-rotation that changes m's partner from w to w' or some other woman w'' such that m prefers w to w'' and w'' to w', and hence there must be a meta-rotation ρ such that $(m, w) \in \rho$. □

Theorem 6.5.2. *All the super-stable pairs can be found in $O(a^2)$ time.*

Proof This follows immediately from Theorem 6.5.1, and the fact that the meta-rotation poset for I, and hence the meta-rotations for I, can be found in $O(a^2)$ time. □

6.6 Generating all super-stable matchings

In this section we show how to generate the super-stable matchings for an instance of SMTI in a manner analogous to that for generating the stable matchings for an instance of SM(I) presented in [13].

For a given instance I of SMTI, let R be the representation of the meta-rotation poset for I found by Algorithm POSET2. We use R to construct a tree T with root node r corresponding to M_0, and edges corresponding to the meta-rotations for I. The tree T is constructed so that any path in T from r to a node v corresponds to a unique closed subset of R, and hence, by Theorem 6.2.7, to a super-stable matching in I. Further, the order of the meta-rotations on a path will be such that every predecessor of a meta-rotation Φ which is part of the path will precede Φ on the path, so that the super-stable matching corresponding to v can be generated by eliminating the meta-rotations on the path to v from r in order.

To construct T we must first label the meta-rotations in R according to a topological order on R. This can be achieved in $O(a)$ time, since R has $O(a)$ edges. We then construct T in the following manner.

We start at r, and expand T from each unexpanded node in turn. Let v be an unexpanded node. We expand v as follows. Let $p(v)$ be the set of meta-rotations on the path from r to v, and let w be the label on the final meta-rotation on the path. Let $q(v)$ be the set of meta-rotations in R which have indegree zero when the meta-rotations in $p(v)$ and all incident edges are removed from R, and which have a label larger than w. We add $|q(v)|$ nodes to T, with an edge from v to each new node labeled with one of the meta-rotations in $q(v)$.

Lemma 6.6.1. *Given R we can construct T in $O(ka)$ time, where k is the number of nodes in T.*

Proof For each node v of T, we require a graph G_v derived from R by deleting those nodes that correspond to meta-rotations on the path $p(v)$, and all edges incident on them. Further, we need to know the indegree of each node in G_v. Then, if ρ is the meta-rotation on the edge from v to h, $q(h)$ is the union of the set of all neighbours of ρ in G_v with indegree 1 (these all have label larger than ρ since all the meta-rotations on $p(v)$ have been removed from G_v), and the set of meta-rotations in $q(v)$ with label larger than ρ. Since there are only $O(a)$ edges in R, no node in R, and hence in G_v, can have outdegree greater than $O(a)$, so we can find the first part in time $O(a)$. Now consider $q(v)$. Suppose $|q(v)| > n$. Then some man must appear in at least two meta-rotations in $q(v)$. But the meta-rotations in $q(v)$ are all exposed in the super-stable set obtained by eliminating the meta-rotations in $p(v)$, a contradiction. Hence $|q(v)| \leq n$, so we can find the second part in time $O(n)$.

To construct these graphs efficiently we must construct T in a depth first manner. Thus, for a node v, we find all the elements in G_v which have indegree 0 (the edges out of v) and store them. We then choose the edge with the lowest label larger than that on the last edge on $p(v)$ which has not been visited, $< v, w >$ say, with meta-rotation Φ, and construct G_w by deleting Φ and all incident edges from G_v. Since there are only $O(a)$ edges in R this can be done in time $O(a)$. Further, the indegree of each node in G_w is the same as its indegree in G_v except for the neighbours of Φ which have indegree 1 less, thus the indegrees can be calculated in $O(a)$ time. When we come back from node w to node v, we need to know the meta-rotation on edge $< v, w >$ to reconstruct G_v. We then repeat this process for the edge with the lowest label which has not been visited. In this manner we can construct T in $O(a)$ time per node. □

It remains to prove that the nodes of T are in one-to-one correspondence with the closed subsets of R.

Lemma 6.6.2. *Every node of T corresponds to a unique closed subset of R.*

Proof Clearly r corresponds to a closed subset of R, namely the empty set. Consider each of the edges emanating from r. By the construction of T each corresponds to a meta-rotation initially exposed, and hence the nodes u for each edge (r, u) correspond to closed subsets of R. Now let v be an arbitrary node of T, and assume that every node added to T before v corresponds to a closed subset of R. Let u be the immediate predecessor of v in T. By the construction of T, the edge (u, v) represents a meta-rotation for which every predecessor is represented by an edge on the path from r to u. It follows that v corresponds to a closed subset of R.

Now suppose that two nodes, u and v say, of T correspond to the same closed subset of R. Let $\Phi_1, ..., \Phi_s$ and $\Pi_1, ..., \Pi_s$ be the edge labels on the paths from r to u and v respectively. Let j be such that $\Phi_i = \Pi_i$ for every $1 \leq i < j$, but $\Phi_j \neq \Pi_j$. Suppose, without loss of generality, that the label on Φ_j is greater than that on Π_j. Then Π_k is Φ_j for some $k > j$. It follows that an edge must have been added to the path from r to u despite the meta-rotation with which that edge is labelled having topological label smaller than the topological label on the meta-rotation preceding that edge on the path from r to u, a contradiction. The result follows. □

Lemma 6.6.3. *Every closed subset of R is represented by at least one node of T.*

Proof Let $\Phi = \Phi_{i_1}, ..., \Phi_{i_s}$ be a closed subset of R, ordered according to the topological order on R. Suppose Φ is not represented by any node in T. Let j be such that $\Phi_{i_1}, ..., \Phi_{i_j}$ are the edge labels on the path from r to a node u in T, but no node in T has edge labels $\Phi_{i_1}, ..., \Phi_{i_{j+1}}$. Consider the expansion of u. Since $\Phi = \Phi_{i_1}, ..., \Phi_{i_j}$ include all the predecessors of $\Phi_{i_{j+1}}$ in R, $\Phi_{i_{j+1}}$ certainly has indegree 0 when the meta-rotation in $p(u)$ and all incident edges are removed from R. Further, since $\Phi_{i_1}, ..., \Phi_{i_s}$ is a topological ordering of the meta-rotations in Φ, $\Phi_{i_{j+1}}$ also has label larger than Φ_{i_j}. Thus there is an edge added from u to a new node v, with the edge labelled $\Phi_{i_{j+1}}$. This contradiction proves the result. □

Lemmas 6.6.1, 6.6.2 and 6.6.3 together give rise to the following theorem.

Theorem 6.6.4. *We can generate the set of all super-stable matchings in $O(a^2 + ka)$ time, where k is the number of matchings.*

6.7 Finding an egalitarian super-stable matching

Algorithm SUPER2 produces a matching which is simultaneously man-optimal and woman-pessimal. Now we consider how to find a super-stable matching which does not treat the sets so inequitably. There are two ways in which we will approach this problem. In Section 6.8 we show how to find a super-stable matching which minimises the regret of the worst-off participant, while in this section we show how to find a matching which minimises the sum of the regrets of all the participants.

In this section the meta-rotation poset is exploited in a manner that is directly analogous to the exploitation of the rotation poset in the paper of Irving et al. [25]. The results presented here are, in the most part, identical to those in [25].

For a pair (m, w), $mr(m, w)$ (resp. $wr(m, w)$) is equal to 1 plus the number of women (resp. men) whom m (resp. w) prefers to w (resp. m). An egalitarian super-stable matching is one in which the weight of a matching M, $w(M)$, where

$$w(M) = \sum_{(m,w)\in M} (mr(m, w) + wr(m, w))$$

is minimised.

We define the *weight* of a meta-rotation Φ to be

$$w(\Phi) = \sum_{i=1}^{l}(\sum_{j=0}^{r_i-1}(mr(m_j^i, w_j^i) - mr(m_j^i, w_{j+1}^i)) + \sum_{j=0}^{r_i-1}(wr(m_j^i, w_j^i) - wr(m_{j-1}^i, w_j^i)))$$

where l is the number of cycles in Φ of the form $\{(m_0, w_0), ..., (m_{r_i-1}, w_{r_i-1})\}$, and $j + 1$ and $j - 1$ are taken modulo r_i.

It can be shown that the weight of a meta-rotation is equal to the change in the weight of a matching when the meta-rotation is eliminated from the super-stable set from which the matching is derived. More generally, for any matching M,

$$w(M) = w(M_0) - \sum_{i=1}^{k} w(\Phi_i)$$

where M can be obtained from M_0 by eliminating the meta-rotations $\Phi_1, ..., \Phi_k$ from the relevant super-stable sets.

Recall that, for an instance of SMTI, there is a one-to-one correspondence between the closed subsets of the meta-rotation poset for the instance and the super-stable matchings for the instance (Theorem 6.2.7). To facilitate finding an egalitarian super-stable matching we form the *weighted meta-rotation poset* by adding the weight of each meta-rotation to the corresponding node of the poset. For each closed subset of the poset, we refer to the total weight of the meta-rotations in that subset as the *weight* of the subset. We can then find an egalitarian super-stable matching by finding a closed subset of the weighted meta-rotation poset with maximum weight.

Let I be an instance of SMTI. Our starting point is R, the representation of the meta-rotation poset for I found by Algorithm POSET2, which we can obtain in $O(a^2)$ time. Recall that R has $O(a)$ edges. We form a network N from R by adding to R nodes s and t, a source and a sink respectively, and by adding the following edges:

- if meta-rotation ρ has weight $w < 0$, an edge is added from s to ρ, with capacity $|w|$;
- if meta-rotation ρ has weight $w > 0$, an edge is added from ρ to t, with capacity w;

In addition, every edge in R is included in N, with infinite capacity. A *positive node* is a node representing a meta-rotation with weight greater than 0, while a *negative node* is a node representing a meta-rotation with weight less than 0.

Lemma 6.7.1. *Let S be the set of positive nodes which do not have their edge into t cut by a minimum cut C in N. Let T be the set of predecessors of the nodes in S. Then $S \cup T$ is a maximum weight closed subset of R.*

Proof Let W be a maximum weight closed subset of R. Let v be a negative node in W. Then v must precede at least one positive node in W. Thus finding a maximum weight closed subset of R is equivalent to finding a subset U of the positive nodes in R which maximizes, over all possible subsets, $w(U) - |w(P(U))|$, where $P(U)$ is the set of negative

nodes that are predecessors of any positive node in U. This, in turn, is equivalent to finding a subset U of the positive nodes in R which minimizes, over all possible subsets, $w(V) - w(U) + |w(P(U))|$, where V is the set of positive nodes in R.

Let U be an arbitrary subset of V. In N, suppose every edge from s to a node in $P(U)$ is cut, and suppose that every edge from a node in $V-U$ is cut. Then this set of edges defines a cut in N. Hence the capacity of C, $cap(C)$, is such that $cap(C) \leq w(V-U)+|w(P(U))|$. Now consider S. By definition of S, C must cut all the edges from a node in $V - S$ to t. Further, C must cut all the edges from s to a node in $P(S)$, since C must have finite capacity and all the edges in N from R have infinite capacity. Thus $cap(C) = w(V-S)+|w(P(S))| \leq w(V-U)+|w(P(U))|$, where U is an arbitrary subset of V. The result follows. □

Lemma 6.7.2. *Given S, as defined in Lemma 6.7.1, an egalitarian super-stable matching can be found in $O(a)$ time.*

Proof To find the predecessors of S, we can track backwards from each node in S, marking a node when it is encountered for the first time, and starting from the next node in S when we reach s or a marked node. Since R has $O(a)$ edges this can be achieved in $O(a)$ time. Then we simply start from M_0 and eliminate the marked meta-rotations in topological order. □

As noted above, it takes $O(a^2)$ time to find R. It is therefore desirable to bound all the other work we need to do to find an egalitarian super-stable matching by $O(a^2)$. To show that we can find a minimum cut in N in $O(a^2)$ time, we require the following lemma.

Lemma 6.7.3. *A minimum cut in N has capacity bounded by $O(a)$.*

Proof It is clear that a minimum cut cannot have capacity greater than the sum of the capacities of the edges from the nodes in R to t. Recall that each such edge has a capacity equal to the weight of the meta-rotation at the originating node. We show that the sum of these weights is bounded by $O(a)$.

The weight of a meta-rotation consists of the sum of the change of the men's partners added to the sum of the change of the women's partners. The former is negative, so the second must be greater than the weight of the meta-rotation. But this second part,

summed over all the meta-rotations, clearly cannot exceed the number of pairs in the preference lists, so is bounded by $O(a)$. □

Theorem 6.7.4. *A minimum cut C in N can be found in $O(a^2)$ time.*

Proof If all capacities in N are integral, then Ford-Fulkerson runs in $O(E|f^*|)$ time, where E is the number of edges in N, and f^* is a maximum flow in N [4]. Certainly E is $O(a)$, and, by Lemma 6.7.3, $|f^*|$ is bounded by $O(a)$. □

Combining the foregoing results, we get the following:

Theorem 6.7.5. *For an instance I of SMTI, an egalitarian super-stable matching can be found in $O(a^2)$ time.*

Feder [6] presented an algorithm to find an egalitarian stable matching in $O(n^{2.5} \log n)$ time for an instance of SM of size n, but as there is a $O(a^2)$ pre-processing cost for super-stability, extending his algorithm would not give any improvement. Note that the construction given here was first used by Picard [45].

6.8 Finding a minimum regret super-stable matching

Recall that, for a super-stable matching M, the regret of a person p is the position in p's preference list of the partner of p in M. The regret of a super-stable matching M, denoted $r(M)$, is the maximum regret of any person in M. In other words the regret of a super-stable matching is measured by the person who is worst off in that matching. A *minimum regret super-stable matching* is a super-stable matching with smallest regret amongst all the super-stable matchings for a given instance of SMTI. In this section we present an algorithm that finds a minimum regret super-stable matching in $O(a)$ time.

Recall that, for an instance I of SMTI that admits a super-stable matching, the base lists are the intersection of the preference lists obtained after applying Algorithm SUPER2 to find the man-optimal super-stable matching with those obtained after applying Algorithm SUPER2 to find the woman-optimal super-stable matching.

Let $\mathcal{B}$ be the base lists for an instance I of SMTI that admits a super-stable matching. Again we assume that $M_0 \neq M_z$, for otherwise the single super-stable matching for I must

trivially be the minimum regret super-stable matching. Recall that to *delete* a pair (m, w) we break any engagement between m and w, and remove each of m and w from the other's list.

Algorithm MinReg starts from $\mathcal{B}$, and from two matchings $M = M_0$ and $M' = M_z$. The algorithm finds a person x with regret $max(r(M), r(M'))$ in M or M'. If that person has the same partner in M and M' then the algorithm outputs M, which we show later must be a minimum regret super-stable matching. Otherwise, if x is a man the algorithm deletes $(x, p_{M'}(x))$, reactivates Algorithm SUPER2 with the women proposing, and sets M' to be the super-stable matching output by Algorithm SUPER2. Thus, in M', x now has a better partner. Alternatively, x must be a woman. The algorithm then deletes $(p_M(x), x)$, reactivates Algorithm SUPER2 with the men proposing, and sets M to be the super-stable matching output by Algorithm SUPER2. Thus, in M, x now has a better partner. We show that, regardless of which conditional statement is executed, there must still be a minimum regret super-stable matching contained in the preference lists. The algorithm is displayed in Figure 6.7.

$M := M_0$;
$M' := M_z$;
repeat {
  let x be a person with regret $max(r(M), r(M'))$ in M or M';
  if $p_M(x) = p_{M'}(x)$ {
    output M;
    halt; }
  else if x is a man{
    delete $(x, p_{M'}(x))$;
    reactivate Algorithm SUPER2 with the women proposing;
    M' := matching output by Algorithm SUPER2;}
  else{
    delete $(p_M(x), x)$;
    reactivate Algorithm SUPER2 with the men proposing;
    M := matching output by Algorithm SUPER2;}}

Figure 6.7: Algorithm MinReg - finding the minimum regret super-stable matching

We prove the correctness of Algorithm MinReg via a sequence of lemmas, but first we make some definitions. A matching M *man-dominates* a matching M' if and only if every man who has different partners in M and M' prefers his partner in M to his partner in M'. Woman-dominance is defined similarly. A matching M'' is *between* M and M' if and only if $M \neq M''$, $M' \neq M''$, and either M man-dominates M'' and M'' man-dominates M', or M woman-dominates M'' and M'' woman-dominates M'.

Lemma 6.8.1. *Suppose that, for one particular execution of Algorithm MinReg, M' changed from M_1 to M_2 because some man m was x, and $p_M(m) \neq p_{M'}(m)$. Then every super-stable matching between M_2 and M_1 contains $(m, p_{M'}(m))$.*

Proof Suppose that there exists a super-stable matching M_3 such that M_3 is between M_2 and M_1, and $(m, w) \notin M_3$, where $w = p_{M_1}(m)$. Then M_2 man-dominates M_3, M_3 man-dominates M_1, $M_2 \neq M_3$ and $M_1 \neq M_3$, and M_3 must be contained in the preference lists after (m, w) has been deleted. It follows that at least one pair from M_3 must be deleted by the algorithm after splitting m and w. Suppose (m', w') was the first such pair deleted during the execution of the algorithm. There are two cases to consider.

Case 1: Suppose (m', w') was deleted because m' received a proposal from a woman w'' whom he prefers to w'. Now w'' must be matched in M_3 to a man m'' such that w'' prefers m' to m'', or is indifferent between them, since (m', w') is the first pair from M_3 to be deleted. But then (m', w'') is a super-stable blocking pair for M_3, a contradiction.

Case 2: Suppose (m', w') was deleted because m' is multiply engaged, to women in the tie in his list containing w'. Let w'' be an arbitrary woman engaged to m' at this point. In M_3, w'' cannot be matched with a man she prefers to m', since (m', w') is the first pair from M_3 to be deleted. Since $(m', w') \in M_3$, (m', w'') is a super-stable blocking pair for M_3, a contradiction.

It follows that (m, w) is in every super-stable matching between M_2 and M_1. □

Lemma 6.8.2. *Suppose that, for one particular execution of Algorithm MinReg, M changed from M_1 to M_2 because some woman w was x, and $p_M(w) \neq p_{M'}(w)$. Then every super-stable matching between M_2 and M_1 contains $(p_M(w), w)$.*

Proof The proof is similar to that for Lemma 6.8.1. □

In the following theorem, $c_M(x)$ denotes the regret of person x in super-stable matching M.

Theorem 6.8.3. *Algorithm MinReg produces a minimum regret super-stable matching.*

Proof Suppose that, at the start of some iteration of the loop, the preference lists contain a minimum regret super-stable matching. We show that the preference lists must contain a minimum regret super-stable matching at the end of the loop iteration. Let x be a person with regret $max(r(M), r(M'))$ in M or M'. Without loss of generality, suppose that x is selected by the algorithm, and suppose that $p_M(x) \neq p_{M'}(x)$. Suppose further that x is a man (the case that x is a woman is analogous, and uses Lemma 6.8.2). As $c_M(x) < c_{M'}(x)$ it follows that $c_{M'}(x) = \max(r(M), r(M'))$. Let M_1 be M' at the start of the loop iteration, and let M_2 be M' at the end of the loop iteration. By Lemma 6.8.1, every super-stable matching between M_2 and M_1 contains $(x, p_{M'}(x))$. Thus, for any super-stable matching M_d contained in the preference lists for which at least one pair of M_d was deleted during the loop iteration in question, it follows that $r(M_d) \geq c_{M_d}(x) = c_{M'}(x) = \max(r(M), r(M'))$. Hence there must be a minimum regret super-stable matching contained in the preference lists at the end of the loop iteration. Since there is clearly a minimum regret super-stable matching contained in the preference lists at the start of the first loop iteration, the result follows by induction.

Let M_{min} be the matching output by the algorithm. If $M = M'(= M_{min})$ at this point then, by the foregoing, M_{min} is a minimum regret super-stable matching, as it is the only matching contained in the preference lists. So suppose $M \neq M'$. Consider x. Since $p_M(x) = p_{M'}(x)$, $p_M(x)$ must be the only person on x's list. Hence $(x, p_M(x))$ is in every super-stable matching between M and M', and so every super-stable matching between M and M' has regret $r(M)$, and so $M_{min} = M$ is a minimum regret super-stable matching. □

6.8.1 Implementation and analysis of Algorithm MinReg

Initially we require M_0 and M_z, each of which takes $O(a)$ time to find. The algorithm starts at M_0 and M_z, and outputs a matching between M_0 and M_z. It follows that the total number of proposals/deletions is $O(a)$. We also need to be able to find a person of regret $max(r(M), r(M'))$. We show how to find a person of maximum regret in M' in $O(1)$

time below, where n is the number of men in the instance. The case for M is analogous, and then it is simply a matter of looking at a set of pointers of size $O(n)$ and determining which points to the highest value.

We maintain two arrays of n doubly-linked lists, one for the men and one for the women. For the men (resp. women), there is a list for each possible regret r of a man (resp. woman), with a pointer p_m (resp. p_w) to the list representing the highest value of r which is not empty. In addition we require a pointer for each person, pointing to that person in the relevant linked list. Finally, we require a counter for each list, to maintain a total of the number of people in that list. Clearly we can find a man of maximum regret by following the pointer p_m. After a reactivation of Algorithm SUPER2, each person with a new mate is removed from their current linked list, in constant time using the set of pointers, and added to the head of the linked list corresponding to their new regret r, with the corresponding increments and decrements made to the list counters. If the counter for the list pointed to by p_m holds 0, we set p_m to point to the list for regret one less, until we find a list with a non-zero counter. Since a man's regret can only improve as the algorithm executes, this takes $O(n)$ time in total. For the women, if any list counter changes from 0, then p_w is set to point to that list if it represents a regret greater than that in the list currently pointed to by p_w. Overall these updates take time $O(n)$, as does initialisation of the list structures. Hence Algorithm MinReg takes $O(a)$ time in total.

6.9 Conclusions and subsequent work in the literature

In this chapter we have shown that, for an instance I of SMTI, the set of all super-stable matchings can be found in $O(a^2 + ka)$ time, where k is the number of super-stable matchings; all the super-stable pairs can be found in $O(a^2)$ time; all the meta-rotations can be found in $O(a^2)$ time; an egalitarian super-stable matching can be found in $O(a^2)$ time; and a minimum regret super-stable matching can be found in $O(a)$ time. Subsequent to the completion of this work, Dias et al. [5] gave algorithms for finding all stable pairs and all stable matchings for an instance of Stable Marriage with Forbidden pairs (SMF), with time complexities $O(a)$ and $O(a + kn)$ respectively, where k is the number of stable matchings. Manlove and Fleiner [39] has shown that we can reduce an instance I of SMT to an instance I' of SMF so that the set of super-stable matchings in I is exactly the set

of stable matchings in I'. This reduction takes $O(a)$ time, and so this method provides a speed up for the problems of finding all super-stable pairs and all super-stable matchings for an instance of SMTI, since we can reduce an instance of SMTI to an instance of SMT in time linear in the number of acceptable pairs, such that the super-stable matchings of the two instances are in one-to-one correspondence. Additionally, it would seem likely that this method could lead to an algorithm to find an egalitarian super-stable matching more efficiently than the method of Section 6.7.

Chapter 7

On weak stability in SMTI

7.1 Introduction

In this chapter we present a number of results relating to SMTI under weak stability. Recall that weakly stable matchings may have different cardinality, though the largest is at most twice the size of the smallest [34]. Furthermore, Maximum Cardinality SMTI, the problem of finding a maximum cardinality weakly stable matching in an instance of SMTI, is NP-hard, even if the ties are at the tails of lists and on one side only, and each tie has length 2 [40]. Thus we are interested in approximating a maximum cardinality weakly stable matching. As noted in Chapter 1, Maximum Cardinality SMTI is not approximable within δ, unless P=NP, for some $\delta > 1$, even if the preference lists are of constant length, there is at most one tie per list, and the ties occur on one side only [16]. However δ is close to 1, so it is still feasible that there will be some improvement on the best known approximation algorithms, listed in Section 1.6. Additionally, there is an instance of SMT which admits neither a man-optimal weakly stable matching nor a woman-optimal weakly stable matching [47], thus apparently precluding the existence of a lattice structure similar to that discussed in Chapter 6. This lattice, of course, allows us to solve a number of problems, including listing all the super-stable matchings for an instance of SMTI, in time polynomial in the number of such matchings. We are certainly interested in being able to do the same for the weakly stable matchings in an instance of SMTI, but to date little progress has been made in producing multiple weakly stable matchings for an instance, let alone all the weakly stable matchings. It is also known that the problems of finding an

egalitarian weakly stable matching and finding a minimum regret weakly stable matching are both NP-hard, even if the ties are on one side only, there is at most one tie per list and each tie has length 2 [40]. It has also been shown that both problems are also not approximable within $\Omega(n)$ unless P=NP [16]. Additionally, determining whether a given man-woman pair is weakly stable is NP-complete, even if the ties are at the tails of lists and on one side only, and each tie has length 2 [40].

In Chapter 8 we look at weak stability in instances of SMTI in which the preferences of at least one set are sublists of a fixed complete ordering of the other set, but first we study weak stability in a more general setting. In Section 7.2 we show a relationship between the size of a maximum cardinality weakly stable matching and the size of all strongly stable matchings in an instance of SMTI or SRTI which admits a strongly stable matching. We also show that this relationship is potentially stronger in HRT, i.e., when we introduce capacitated agents. Thus, for a restricted set of instances of SMTI, we have an algorithm for approximating a maximum cardinality weakly stable matching with guarantee better than any previously known. In Section 7.3 we give an improvement on the best known approximation algorithm for both maximum and minimum cardinality weakly stable matchings for instances of SMTI with sparse ties. In Section 7.4 we show that we can efficiently determine whether an instance of SMTI admits a unique weakly stable matching. Finally, in Section 7.5 we show that the problem of finding a weakly stable matching in an instance of SMTIF (Stable Marriage with Ties, Incomplete lists and Forbidden pairs) is NP-complete, though we can find a super-stable matching in such an instance efficiently. The former problem, had it been solvable in polynomial-time, could have led to an efficient algorithm for generating all the weakly stable matchings in time linear in the number of such matchings.

Note that in this chapter we must occasionally draw a distinction between the three types of blocking pair. Therefore we say a pair is a *weakly stable blocking pair* for a matching M if it blocks M when considered as a weakly stable matching, we say a pair is a *strongly stable blocking pair* for a matching M if it blocks M when considered as a strongly stable matching, and we say a pair is a *super-stable blocking pair* for a matching M if it blocks M when considered as a super-stable matching.

7.2 Comparing strongly stable matchings to weakly stable matchings

In this section we show that, in the one-to-one matching problems SMTI and SRTI, a strongly stable matching is at least $\frac{2}{3}$ the size of a maximum cardinality weakly stable matching, while there is a potentially stronger result for HRT.

7.2.1 The comparison in SMTI

We start by showing that, if an instance of SMTI admits a strongly stable matching, that matching must be at least $\frac{2}{3}$ the size of a maximum cardinality weakly stable matching.

Theorem 7.2.1. *Let I be an instance of SMTI, let M be a strongly stable matching for I, and let M' be a maximum cardinality weakly stable matching for I. Then $|M| \geq \frac{2}{3}|M'|$.*

Proof Suppose some person p is matched in M' but not in M. Then p's partner in M' must be matched in M. For otherwise there is a pair $(m, w) \in M'$ such that m, w are unmatched in M. But m, w clearly find each other acceptable, and hence (m, w) is a strongly stable blocking pair for M. We show that, to include in M' a person who was unmatched in M, we must include in M' three people who were matched in M. The role of these people depends on the whether we are adding a man or a woman to M', so each person can play two roles. In the following paragraph each p_i is from one set, and each q_i is from the other.

Consider a person p_1 who is unmatched in M. Suppose p_1 is matched in M', to q_1. By the above, q_1 must be matched in M. Consider $p_2 = p_M(q_1)$. Clearly $p_2 \neq p_1$. Clearly q_1 prefers p_2 to p_1, for otherwise (p_1, q_1) is a strongly stable blocking pair for M. Suppose p_2 is unmatched in M', or prefers q_1 to $p_{M'}(p_2)$. Then (p_2, q_1) is a weakly stable blocking pair for M'. Hence p_2 must be matched in M', and must prefer $q_2 = p_{M'}(p_2)$ to q_1, or be indifferent between them. Clearly $q_2 \neq q_1$, since $(p_1, q_1) \in M'$ and $p_2 \neq p_1$. Now suppose q_2 is unmatched in M. Then (p_2, q_2) is a strongly stable blocking pair for M, so q_2 must be matched in M. So, for every man, say, who is matched in M' but not in M there are two women who are matched in M, and each such man generates two such distinct women. Hence $|M'| - |M| \leq x \leq \frac{|M|}{2}$, where x is the number of men who are matched in M' but not in M. Clearly $|M'| \leq \frac{3}{2}|M|$. □

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

Figure 7.1: An instance of SMTI

The instance of SMTI in Figure 7.1 demonstrates that the lower bound of Theorem 7.2.1 is tight, as it admits a strongly stable matching of size 2, namely $(2, 1), (3, 2)$, and a weakly stable matching of size 3, namely $(1, 1), (2, 2), (3, 3)$. Note that this example can easily be extended to instances of arbitrarily large size for which the bound is tight.

1: (1 2)	1: 1
2: 2	2: (1 2)

Figure 7.2: A second instance of SMTI

By contrast, the instance of SMTI in Figure 7.2 admits a strongly stable matching of size 2 (hence also a maximum cardinality weakly stable matching), namely $(1, 1), (2, 2)$, and a weakly stable matching of size 1 (hence also a minimum cardinality weakly stable matching), namely $(1, 2)$. Thus, even if a strongly stable matching is of equal size to a maximum cardinality weakly stable matching, it is still possible for a minimum cardinality weakly stable matching to be half the size of a maximum cardinality weakly stable matching. Note again that this example can easily be extended to instances of arbitrarily large size.

7.2.2 The comparison in HRT

We now extend the result of Theorem 7.2.1 to HRT. This extension gives a potentially stronger result than Theorem 7.2.1.

Theorem 7.2.2. *Let I be an instance of HRT, let M be a strongly stable matching for I, and let M' be a maximum cardinality weakly stable matching for I. Then $|M'| \leq \frac{3}{2}|M| - \frac{1}{2}u_M$, where $u_M = \Sigma_{h \in H^*} f_h$, H^* is the set of hospitals that are not full in M, and f_h is the number of posts that a hospital $h \in H^*$ fills in M.*

Proof Consider a resident r_1 who is unmatched in M. Suppose r_1 is assigned in M' to hospital h_1. Clearly h_1 is full in M, as otherwise (r_1, h_1) is a strongly stable blocking pair for M. Hence there is some resident $r_2(\neq r_1)$ who is assigned to h_1 in M, but is not assigned to h_1 in M'. Clearly h_1 prefers r_2 to r_1, for otherwise (r_1, h_1) is a strongly stable blocking pair for M. Suppose r_2 is unassigned in M', or prefers h_1 to $p_{M'}(r_2)$. Then (r_2, h_1) is a weakly stable blocking pair for M'. Hence r_2 must be assigned to some hospital in M', and must prefer $h_2 = p_{M'}(r_2)$ to h_1, or be indifferent between them. Now suppose h_2 is not full in M. Then (r_2, h_2) is a strongly stable blocking pair for M. Hence h_2 must be full in M. Note that $(r_2, h_1) \notin M'$, by the choice of r_2, so $h_1 \neq h_2$.

So, for every resident who is matched in M' but not in M there are two hospital posts which are filled in M at hospitals that are full in M, and each such resident generates two such distinct hospital posts. Let g_M be the number of hospital posts filled in M at hospitals that are full in M, and let u_M be the number of posts filled in M at hospitals that are not full (i.e., undersubscribed) in M. Then $|M'| - |M| \leq \frac{1}{2}g_M$. Now $|M| = g_M + u_M$, so $|M'| - |M| \leq \frac{1}{2}(|M| - u_M)$, or $|M'| \leq \frac{3}{2}|M| - \frac{1}{2}u_M$. □

7.2.3 The comparison in SRTI

Now we consider the extension of Theorem 7.2.1 to SRTI. The proof is similar to that of Theorem 7.2.1.

Theorem 7.2.3. *Let I be an instance of SRTI, let M be a strongly stable matching for I, and let M' be a maximum cardinality weakly stable matching for I. Then $|M| \geq \frac{2}{3}|M'|$.*

Proof Suppose some agent p is matched in M' but not in M. Then p's partner in M' must be matched in M. For otherwise there is a pair $\{p, q\} \in M'$ such that p, q are unmatched in M. But p and q find each other acceptable, and hence $\{p, q\}$ is a strongly stable blocking pair for M.

Consider an agent p who is unmatched in M. Suppose p is matched in M', to q. By the above, q must be matched in M. Consider $r = p_M(q)$. Then $r \neq p$. Clearly q prefers r to p, for otherwise $\{p, q\}$ is a strongly stable blocking pair for M. Suppose r is unmatched in M', or prefers q to $p_{M'}(r)$. Then $\{q, r\}$ is a weakly stable blocking pair for M'. Hence r must be matched in M', and must prefer $s = p_{M'}(r)$ to q, or be indifferent between them.

Clearly $s \neq q$, since $\{p, q\} \in M'$ and $r \neq p$. Now suppose s is unmatched in M. Then $\{r, s\}$ is a strongly stable blocking pair for M, so s must be matched in M. Clearly $p \neq s$. Potentially, $p_M(s)$ could be matched in M' with an agent who is unmatched in M, and so four agents who are matched in M are required and suffice to allow two agents who are unmatched in M to become matched in M'.

So, for every two agents who are matched in M' but not in M there are four agents who are matched in M. Hence $|M'| - |M| \leq x \leq \frac{1}{2}|M|$, where x is the number of agents who are matched in M' but not in M. Clearly $|M'| \leq \frac{3}{2}|M|$. □

Since SMTI can be viewed as a special case of each of both HRT and SRT, the examples in Figures 7.1 and 7.2 demonstrate the same properties for both of these problems as they do for SMTI.

7.3 An approximation guarantee

In this section we consider the problem of approximating a maximum cardinality weakly stable matching in an instance of SMTI. We denote by $s^+(I)$ and $s^-(I)$ the size of a maximum and minimum cardinality weakly stable matching in I respectively. Ideally, in the case of Maximum Cardinality SMTI, one might hope for a bound of the form $s^+(I)/|M| \leq f(p)$ given an instance I of SMTI, where M is a stable matching found by some approximation algorithm, p is the proportion of preference lists that contain ties, and $f(p)$ is a function that decreases to 1 as p decreases to 0.

However, it is not hard to see that a bound of this form is infeasible. Were such an algorithm to exist, a 'gap' argument could be used to show that it could solve instances of Maximum Cardinality SMTI exactly. Given an arbitrary such instance, it could be 'expanded' by the addition of new persons, none of whom has a tie in his or her list, and none of whom can be matched in any stable matching. With an appropriate expansion factor, application of the supposed approximation algorithm to this derived instance would solve the original instance exactly.

Instead we derive a bound on the *difference* in size between a maximum (or minimum) cardinality stable matching and an arbitrary stable matching, expressed in terms of the number of preference lists that contain ties. So the trivial approximation algorithm - break

all ties arbitrarily and apply the Gale/Shapley algorithm - has a performance guarantee, for both Maximum Cardinality SMTI and Minimum Cardinality SMTI, expressible as a difference rather than a ratio. As observed by Garey and Johnson [11, pp.137-138], this form of performance guarantee can reasonably be viewed as being stronger than the more familiar performance ratio form, and there are relatively few NP-hard problems for which approximation algorithms with performance guarantees of this kind are known.

Some additional definitions are necessary before presenting the main results of this section. Let M and M' be stable matchings for an instance I of SMTI. If a person p strictly prefers his partner in M to his partner in M', or is matched in M but not in M', then we say that p *strictly prefers* M *to* M'. If p is indifferent between his partners in M and M', or has the same partner in M as in M', or is matched in neither M nor M', then we say that p is *indifferent between* M *and* M'.

Define a *tied pair* to be a pair (m, w) such that m is in a tie in w's list, or w is in a tie in m's list (or both). In what follows, t_M denotes the number of tied pairs in M, and u_I denotes the number of preference lists in I that contain ties. In general t_M depends on the matching M, whilst u_I is invariant for the given instance I; clearly $t_M \leq u_I$.

Lemma 7.3.1. *Let T be a maximum cardinality stable matching for a given instance I of SMTI. Then if M is an arbitrary stable matching in I, $|T| \leq |M| + t_M$.*

Proof We construct an undirected edge-coloured graph $G = G(M, T)$ as follows: G has a vertex for each person in I, and two vertices form a blue (respectively red) edge if the corresponding persons are matched in T but not in M (respectively in M but not in T). It is clear that the connected components of G are paths and cycles with edges of alternating colour. Furthermore, $|T| - |M|$ is at most equal to the number of *blue augmenting paths* in G, i.e., the number of paths of odd length in which the first and last edges are blue. Further, every such path has at least three edges, since a component that is a path of length one would provide a blocking pair for one of the supposed stable matchings.

We claim that, in every blue augmenting path, at least one of the intermediate vertices represents a person who is indifferent between T and M, and is therefore in a tied pair in both T and M. This claim, together with the preceding observation, suffices to establish the lemma.

To establish the claim, let $p_1, q_1, \ldots, p_r, q_r$ form a blue augmenting path in G. Since p_1

and q_r are both matched in T but not in M, they both prefer T to M. Suppose that no person in the path is indifferent between T and M. A simple inductive proof starting from p_1 then reveals that q_i $(i = 1, 2, \ldots, r-1)$ strictly prefers M to T, otherwise (p_i, q_i) would block M, and p_i $(i = 2, 3, \ldots, r)$ strictly prefers T to M, otherwise (p_i, q_{i-1}) would block T. Thus (p_r, q_r) blocks M, a contradiction. Hence at least one of the p_i $(2 \leq i \leq r)$ or q_i $(1 \leq i \leq r-1)$ must be indifferent between T and M, as claimed. □

Since $t_M \leq |M|$, it follows immediately by Lemma 7.3.1 that there exists an approximation algorithm for Maximum Cardinality SMTI with performance ratio 2. Using a similar argument to the one employed in the proof of Lemma 7.3.1, we may deduce that $|M| \leq |S| + t_S$, where S is a stable matching of minimum cardinality. Since $t_S \leq |S|$, it follows immediately that there exists an approximation algorithm for Minimum Cardinality SMTI, also with performance ratio 2. The inequality established by Lemma 7.3.1 also leads to the following result:

Theorem 7.3.2. *There is an approximation algorithm A such that, given any instance I of either Maximum Cardinality SMTI or Minimum Cardinality SMTI, A finds a stable matching M in I satisfying the following inequality:*

$$s^+(I) - u_I \leq |M| \leq s^-(I) + u_I.$$

Proof Let M be defined as in Lemma 7.3.1. Since $t_M \leq u_I$, Lemma 7.3.1 implies that $s^+(I) - u_I \leq |M| \leq s^+(I)$. Also by Lemma 7.3.1, $s^+(I) \leq s^-(I) + u_I$, and hence the result follows. □

We remark that, when the ties in a given instance I of SMTI are sparse, i.e., u_I is small compared to the numbers of men and women in I, the performance guarantee indicated by Theorem 7.3.2 is an improvement on the best-known previous results (see Section 1.6).

The following instance is an illustration of the worst case for the above theorem. For each $n \geq 1$, we define an SMTI instance I with $2n$ men, namely $\{p_1, \ldots, p_n, q_1, \ldots, q_n\}$, and $2n$ women, namely $\{r_1, \ldots, r_n, s_1, \ldots, s_n\}$. For each i $(1 \leq i \leq n)$, define preference lists for p_i, q_i, r_i, s_i as follows:

$$\begin{array}{llll} p_i: & s_i \quad r_i & r_i: & p_i \\ q_i: & s_i & s_i: & (p_i \quad q_i) \end{array}$$

There is a stable matching of size n (namely $M_1 = \{(p_i, s_i) : 1 \leq i \leq n\}$) and one of size $2n$ (namely $M_2 = \{(p_i, r_i), (q_i, s_i) : 1 \leq i \leq n\}$). Clearly $s^+(I) = 2n$, and also $s^-(I) = n$

since $|M_2| = 2|M_1|$. Since the difference between $s^+(I)$ and $s^-(I)$ is the number of lists with ties, the bounds given by Theorem 7.3.2 are tight.

7.4 Finding a second weakly stable matching

Finding a way to generate all the weakly stable matchings for an instance of SMTI is an open question of great interest. In this section we take a first tentative step toward being able to generate all the weakly stable matchings. For a given instance I of SMTI, and a matching M which is weakly stable in I, we show that we can find, in time linear in the input size, a matching $M' \neq M$ which is also weakly stable in I, if such a matching exists, thereby determining whether I admits a unique weakly stable matching. Note that we again use Algorithm SUPER2, which is displayed in Figure 1.3 in Section 1.7.

So let I be an instance of SMTI, and let M be a weakly stable matching for I. We consider three cases. The first case is trivial. Suppose I admits at least two super-stable matchings. Then, by finding the man-optimal and woman-optimal super-stable matchings, by applying the man-oriented and woman-oriented versions of Algorithm SUPER2 [34] respectively, we have found two distinct weakly stable matchings, and so at least one of these is a weakly stable matching $M' \neq M$.

There are two cases left to consider, where I admits a unique super-stable matching, and where I does not admit a super-stable matching. We consider these two cases in turn.

First, suppose I does not admit a super-stable matching. The following lemma shows that we can find two distinct weakly stable matchings, if they exist, in $O(a)$ time, where n is the size of I.

Lemma 7.4.1. *Let I be an instance of SMTI which does not admit a super-stable matching, and let M be the weakly stable matching obtained from the man-oriented GS algorithm applied to an instance I' of SMI derived from I. Then we can find a second weakly stable matching different from M in $O(a)$ time.*

Proof Since M cannot be super-stable in I there must exist a pair $(m, w) \notin M$ which is a super-stable blocking pair for M. There are three forms this pair can take:

1. m prefers w to his partner in M, and w is indifferent between m and her partner in M.

2. w prefers m to her partner in M, and m is indifferent between w and his partner in M.

3. m and w are both indifferent between the other and their partner in M.

Where we have indifference in the first two cases the relevant tie will have been resolved in I' so that m or w prefers their partner in M to w or m respectively, and in the third case this must have happened in at least one of the ties, in order that M should be stable in I'. We change the way these ties are resolved so that m and w always prefer the other to their partner in M. Denote by I'' the resulting instance of SMI. Clearly (m, w) now blocks M in I'', so application of the man-oriented GS algorithm to I'' must produce a matching $M' \neq M$ which is weakly stable in I.

Finding a super-stable blocking pair for M takes $O(a)$ time, as does application of the man-oriented GS algorithm, while all other operations, with the necessary data structures in place, take $O(n)$ time in the worst case. As the pre-processing of the data structures can be accomplished in $O(a)$ time (see for example [15]), the process overall is $O(a)$. □

Now we consider the case where I admits a unique super-stable matching M. Let $\mathcal{B}'$ be the lists after the application of Algorithm SUPER2 with the men proposing. Let $\mathcal{B}$ be the original preference lists for the instance with (m, w') deleted for each woman w' such that m prefers w' to $p_M(m)$, and all people with empty lists in $\mathcal{B}'$ removed (M is stable in every instance of SMI obtainable from I, and so the set of people matched in M is exactly the set of people matched in every stable matching in each such instance). For each pair $(m, w) \in M$, if w is in a tie in m's list we partially resolve the tie so that w precedes the remainder of the tie, and if m is in a tie in w's list we partially resolve the tie so that the remainder of the tie precedes m. We call these lists the *man-oriented base lists*. It is clear that each man has a single, unique woman at the head of his list in $\mathcal{B}$. Let $s(m)$ be the set of women belonging to the tie (recalling that ties can be of length 1) on m's list which follows $p_M(m)$, if such a tie exists. Otherwise let $s(m)$ be the empty set. We construct a directed graph G_0 as follows:

Create a node in G_0 for each woman in M. For each man m, create an edge of G_0 from

$p_M(m)$ to w, for each $w \in s(m)$. Consider the graph G_0. If there is a cycle $\mathcal{C}$ in G_0, then we can construct a second weakly stable matching M' as follows:

For each $w \in \mathcal{C}$, set $p_{M'}(w)$ to be $p_M(w')$ where w' precedes w in $\mathcal{C}$. All the other woman in I are paired with the man they are paired with in M.

If there is no cycle then we repeat the above method, starting from the construction of the lists $\mathcal{B}$ from the application of Algorithm SUPER2, this time with the women proposing. If there is a cycle in the new graph G_z then we construct a new matching M'' in the same manner. If there is again no cycle then M is the unique weakly stable matching for I.

The following lemmas prove the correctness of this method.

Lemma 7.4.2. *The matching M' formed by the process described above is weakly stable.*

Proof Suppose that (m, w) is a weakly stable blocking pair for M'. Then m prefers w to $p_{M'}(m)$ and w prefers m to $p_{M'}(w)$. Any woman who has different partners in M and M' prefers her partner in M' to her partner in M, or is indifferent between them. So w prefers m to $p_M(w)$. Suppose m has the same partner in M and M', or is indifferent between his partners in M and M'. Then m prefers w to $p_M(m)$, and (m, w) is a super-stable blocking pair for M, a contradiction. So m prefers $p_M(m)$ to $p_{M'}(m)$. But $p_{M'}(m)$ is a member of $s(m)$, so m must prefer w to every member of $s(m)$, implying that $w = p_M(m)$, or m prefers w to $p_M(m)$. In the latter case (m, w) blocks M, a contradiction, so $w = p_M(m)$, and hence $m = p_M(w)$. But then w prefers m to $m = p_M(w)$, a contradiction. This contradiction establishes the result. □

This proof is also valid for the second iteration of the method, in which M'' is constructed. We need only reverse the roles of the men and women.

Lemma 7.4.3. *If there is no cycle in G_0 then M is the woman-optimal stable matching for every instance of SMI obtainable from I.*

Proof Suppose that there is no cycle in G_0, but there is an instance I' of SMI obtainable from I in which $M'(\neq M)$ is the woman-optimal stable matching. Then there must be a rotation exposed in M. Suppose $\rho = (m_0, w_0), ..., (m_{r-1}, w_{r-1})$ is that rotation. Then w_{i+1} is the second woman on m_i's list in I', and hence $w_{i+1} \in s(m_i)$ in $\mathcal{B}$, by the construction of $\mathcal{B}$, where the subscripts are taken modulo r. But then there must be a cycle $\mathcal{C} =< w_0, w_1, ..., w_{r-1} >$ in G_0. This contradiction establishes the result. □

Reversing the roles of the men and women in the above proof shows that, if there is no cycle in G_z then M is the man-optimal stable matching for every instance of SMI obtainable from I. Combining these two results shows that, if there is a cycle in neither G_0 nor G_z then M is both the man-optimal and the woman-optimal stable matching for every instance of SMI obtainable from I. It follows that M is the unique weakly stable matching for I. Thus the method either produces a second weakly stable matching for I, or else M is the unique weakly stable matching for I. Finally, the application of Algorithm SUPER2 takes $O(a)$ time, while it is certainly the case that every other operation in this process takes $O(a)$ time. It follows that, given an instance I of SMTI and a matching M that is weakly stable in I, we can find a weakly stable matching $M' \neq M$ in $O(a)$ time, if one exists, where a is the number of acceptable pairs in I.

7.5 An aside on SMTIF

As noted earlier, Dias et al. [5] gave an algorithm for finding a stable matching in an instance of SMF. We show that this result can be extended to find a super-stable matching in an instance of SMTIF, but the problem of determining whether an instance of SMTIF admits a weakly stable matching is NP-complete. To see why we are interested in the latter problem, suppose it had been polynomial-time solvable, and that we could extend the results in Section 7.4 to SMTIF. Then we could have generated all the weakly stable matchings for an instance I of SMTI with polynomial-time between the generation of successive matchings by the following method. Find a weakly stable matching M_1 for I, and then use the results in the preceding section to find a different weakly stable matching, M_2 say. Find a pair, (m, w) say, which is in M_1 but not in M_2. Create an instance I_1 of SMTIF in which (m, w) is forbidden, and create a second instance I_2 of SMTIF in which (m, w) is a forced pair i.e., (m, w) must appear in every weakly stable matching. Then the set of weakly stable matchings in I is the union of the set of weakly stable matchings in I_1 and I_2. Repeating this process until every instance of SMTIF does not admit a weakly stable matching will produce all the weakly stable matchings for I.

7.5.1 Finding a super-stable matching in SMTIF

Let I be an instance of SMTIF. Then I comprises a unique instance I' of SMTI together with a set F of forbidden pairs. We assume that I' admits a super-stable matching, otherwise I clearly does not admit a super-stable matching. An algorithm for finding a super-stable matching in I is displayed in Figure 7.3, again using Algorithm SUPER2 (see Section 1.7). The algorithm and Theorem 7.5.2 are extensions of material in [5].

M := man-optimal super-stable matching for I';
while there exists $(m, w) \in M \cap F$ {
    delete (m, w);
    reactivate Algorithm SUPER2 with the men proposing;
    if Algorithm SUPER2 reports that no super-stable matching exists
        no super-stable matching exists for I;
    else
        M := matching output by Algorithm SUPER2; }
output M, a super-stable matching for I;

Figure 7.3: Algorithm SUPER-FORBIDDEN

Lemma 7.5.1. *Let I be an instance of SMTIF. No pair which is contained in a super-stable matching for I is deleted during the execution of Algorithm SUPER-FORBIDDEN.*

Proof Suppose, for a contradiction, that (m, w) is deleted during the execution of the algorithm, but there is a super-stable matching M for I which contains (m, w). Suppose further that (m, w) is the first such pair deleted. Clearly (m, w) can only be deleted by the reactivation of Algorithm SUPER2. Suppose that the deletion takes place because some man m' proposes, and becomes engaged to, w, and w prefers m' to m, or is indifferent between them. Now m' cannot be matched in M with a partner he prefers to w, since (m, w) is the first pair from M to be deleted. But then m' prefers w to $p_M(m')$, or is indifferent between them, and so (m', w) blocks M, a contradiction. □

Theorem 7.5.2. *Let I be an instance of SMTIF. Then Algorithm SUPER-FORBIDDEN determines whether a super-stable matching exists for I, and if so outputs the man-optimal such matching, all in $O(a)$ time.*

Proof Firstly, suppose there is no super-stable matching for I. Then, for every matching M generated by Algorithm SUPER-FORBIDDEN, there is some pair $(m, w) \in M$ such that $(m, w) \in F$. Thus no iteration of the main loop of the algorithm can produce a super-stable matching for I, and at least one pair is deleted from the preference lists during each iteration. It follows that the algorithm must eventually delete a pair from the woman-optimal super-stable matching for I', the unique instance of SMTI from which I can be obtained. The output of Algorithm SUPER2 during the iteration when this pair is deleted cannot be a super-stable matching, and so Algorithm SUPER-FORBIDDEN will terminate with the conclusion that there is no super-stable matching for I.

Now suppose I admits a super-stable matching, M_0. By Lemma 7.5.1 no pair which appears in any super-stable matching for I is deleted by Algorithm SUPER-FORBIDDEN, so the algorithm must eventually produce a matching which is super-stable in I. Let M_1,...,M_k be the sequence of matchings produced by Algorithm SUPER-FORBIDDEN. Then each of M_1,...,M_k must dominate M_0, or be equal to it. Thus M_k must be super-stable in I, and must be man-optimal.

Finally, we construct a boolean matrix containing to record whether a given pair is forbidden or not, taking $O(a)$ time. We can then check in constant time whether a given pair (m, w) is forbidden at the point at which w first appears at the head of m's list, and mark it as such if relevant. Since it takes $O(a)$ time for the combined iterations of Algorithm SUPER2, the overall running time of Algorithm SUPER-FORBIDDEN is $O(a)$. □

7.5.2 Finding a weakly stable matching in SMTF

We now show that the problem of finding a weakly stable matching in an instance of SMTF is NP-complete.

Theorem 7.5.3. *Determining whether an instance of SMTF admits a weakly stable matching is NP-complete.*

Proof Clearly the problem of determining whether an instance of SMTF admits a weakly stable matching is in NP. We transform from the special case of Maximum Cardinality SMTI where the target is a complete matching [40]. Let I be an instance of Maximum Cardinality SMTI, in which the set of men is $X_1 = \{m_1, ..., m_n\}$ and the set of women is

$Y_1 = \{w_1, ..., w_n\}$. We create an instance I' of SMTF as follows: let $X = X_1 \cup \{m_{n+1}\}$ be the set of men in I' and let $Y = Y_1 \cup \{w_{n+1}\}$ be the set of women in I'. The preference lists in I' are as follows (where P_i and Q_i denote the preference lists of m_i and w_i in I respectively, and "$-$" in agent p's list denotes all the agents on the opposite side who do not appear explicitly elsewhere on p's list, listed in arbitrary order):

$$\begin{aligned} m_i: &\quad P_i\ w_{n+1}\ - && (1 \le i \le n) \\ m_{n+1}: &\quad w_{n+1}\ - \\ w_i: &\quad Q_i\ m_{n+1}\ - && (1 \le i \le n) \\ w_{n+1}: &\quad m_1\ m_2\ ...\ m_{n+1} \end{aligned}$$

Finally, $F = \{(m_i, w_{n+1}) : 1 \le i \le n\}$, where F is the set of forbidden pairs. We show that I admits a complete weakly stable matching if and only if I' admits a weakly stable matching.

Suppose I admits a complete weakly stable matching M. Then $M' = M \cup \{(m_{n+1}, w_{n+1})\}$ is clearly weakly stable in I', and does not contain a forbidden pair.

Conversely, suppose I' admits a weakly stable matching M'. Then (m_{n+1}, w_{n+1}) is in M'. Suppose m_i is matched in M' with a woman w_j such that $w_j \notin P_i$ for some $1 \le i \le n$ and $1 \le j \le n$. Then (m_i, w_{n+1}) blocks M', a contradiction. It follows that m_i must be matched in M' with a partner from P_i. Hence $M' \setminus \{(m_{n+1}, w_{n+1}\}$ is a complete weakly stable matching in I. The result follows. □

Chapter 8

Master Lists

8.1 Introduction

In this chapter we still focus on weak stability, but we now address a special case of SMTI, where every person in at least one set has a preference list which is a sublist of a fixed complete ordering of the people in the other set. We call this complete ordering a *master list*. We consider two cases: where there is a master list for both sides - Master List SMTI or ML-SMTI; and where there is a master list for at least one side - Single Master List SMTI or SML-SMTI. Such a situation could occur where, for example, students are ranked by schools based on exam results, and the schools may or may not be ordered according to national rankings. Thus ML-SMTI and SML-SMTI are versions of SMTI which could occur in real-life situations. Additionally we study ML-SMT and SML-SMT, where every agent has a complete preference list.

As well as different categories of instance, we study a number of different problems. We start with a definition. A *man minimum regret matching* is a matching in which the regret of the worst-off man is minimised. We denote by Egalitarian ML-SMT, (Man) Minimum Regret ML-SMT, and Stable Pair of ML-SMT the problems of finding an egalitarian weakly stable matching, finding a (man) minimum regret weakly stable matching and the problem of determining whether a given pair is weakly stable in an instance of ML-SMT respectively. Additionally the first two problems could also be framed in the context of SML-SMT, while the third could also be framed in any of ML-SMTI, SML-SMT or SML-SMTI. Finally we also study Maximum Cardinality ML-SMTI and Minimum Cardinality ML-

SMTI, the problems of finding a maximum cardinality and minimum cardinality weakly stable matching respectively in an instance of ML-SMTI, and the same problems in SML-SMTI.

We show that, even with this additional restriction, the relevant problems in ML-SMTI and SML-SMTI still remain hard, as they do in SML-SMT, except in the special cases of Minimum Regret SML-SMT, where one side effectively becomes irrelevant anyway, and the maximum and minimum cardinality problems. In the context of ML-SMT, on the other hand, we show that all the above problems are solvable in polynomial-time, and further we show that the set of weakly stable matchings for an instance of ML-SMT can be generated with sublinear time between each matching. This is the first such algorithm for any version of SMTI under weak stability, and is all the more interesting as it not based on any of the usual techniques for solving stable marriage problems.

The NP-completeness, NP-hardness, and inapproximability results produced for SMT and SMTI characterise the restrictions on the instances in terms of the preference lists of individual agents (see, for example, the first paragraph of Chapter 7). For instances of SML-SMT(I), we continue to characterise the restrictions on the agents on the side for whom the preference lists are not sublists of a master list in these terms, but for the other side, and for instances of ML-SMT(I), we shall characterise the restrictions in terms of the master list. We do this because, in any practical setting, it is unlikely that agents whose preference lists are sublists of a master list will be required to select an acceptable subset of the other agents in a way that conforms to a particular pattern. We do, however, highlight cases where these agents have lists of constant length.

In Sections 8.2 and 8.3 we present a number of NP-completeness and inapproximability results for Maximum Cardinality ML-SMTI and SML-SMTI. These results build on results from [40] and [16]. In Section 8.4, we show that Egalitarian ML-SMT can be solved in $O(n)$ time, for an instance of size n, while in contrast Egalitarian SML-SMT is NP-complete, even if the master list is strict. In Section 8.5 we show that Minimum Regret ML-SMT can be solved in $O(n)$ time, for an instance of size n (i.e., with n men), and Minimum Regret SML-SMT can be solved in $O(n^2)$ time, for an instance of size n, if there is no tie at the tail of the master list. By contrast, we show that Minimum Regret SML-SMT is NP-complete if there is a tie at the tail of the master list, even if there are no ties on the other side, and Man Minimum Regret SML-SMT is NP-complete if the master list is of

men, even if it is strictly ordered. In Section 8.6, we show that Stable Pair of ML-SMTI is NP-complete, even if there are ties in only one of the master lists, and that SML-SMT is NP-complete, even if the master list is strict. By contrast, we show that Stable Pair of ML-SMT can be solved in $O(n)$ time for an instance of size n. Additionally, we show that we can find all s stable pairs in an instance of ML-SMT of size n in $O(n+s)$ time. In the final section, Section 8.7, we show that we can generate all k weakly stable matchings for an instance I of ML-SMT in $O(\log(n)k)$ time, where I is of size n.

Note that the NP-completeness and NP-hardness results hold by restriction for the equivalent master list versions of HRT, SRT(I) and SFT.

Throughout this chapter, $[A]$ denotes the agents in set A listed in arbitrary strict order, and "$-$" in agent p's list denotes all the agents on the opposite side who do not appear explicitly elsewhere on p's list, listed in arbitrary strict order. As ever, ties are represented by parentheses and, whenever ties are involved, stable means weakly stable.

Note that a number of the NP-completeness proofs in this chapter extend historical results. In most instances the proofs of these results are long, and so we do not include in full the established details, except where necessary for a complete understanding of the extension.

8.2 Maximum Cardinality ML-SMTI

We start by demonstrating the NP-completeness of Maximum Cardinality ML-SMTI, even if the ties occur on one side only. To do this, we show that the transformation of Lemma 1 of [40] can have master lists imposed directly. A *subdivision graph* of a graph G^* is obtained from G^* by replacing every edge in G^* with a path of length 2. Exact Maximal Matching is the problem of finding a maximal matching of a specified size in a given graph. Note that the following result is used, in Theorem 8.2.2, to obtain an even stronger result.

Lemma 8.2.1. *Maximum Cardinality ML-SMTI is NP-complete, even if the ties occur in one master list only.*

Proof Clearly Maximum Cardinality ML-SMTI is in NP. We transform from Exact Maximal Matching for subdivision graphs, which is NP-complete [19][1]. Let $G = (V, E)$ be

[1]In fact Horton and Kilakos proved that Minimum Edge Dominating set, which is known to be polynomially equivalent to Minimum Maximal Matching, is NP-complete. Minimum Edge Dominating set is

the graph for an instance of Exact Maximal Matching for subdivision graphs. Then G is a bipartite graph with vertex sets $V_1 \subseteq V$ and $V_2 \subseteq V$, and every vertex in V_1 has degree 2. We may assume, without loss of generality, that $|V_1| = |V_2| = n$ and that $K \leq n$, where K is the size of the target matching (see Lemma 1 of [40] for details). Let $V_1 = \{m_1, ..., m_n\}$ and let $V_2 = \{w_1, ..., w_n\}$. We construct an instance I of Maximum Cardinality ML-SMTI as follows: let $V_1 \cup V_1' \cup X$ be the set of men in I, and let $V_2 \cup Y \cup Z$ be the set of women in I, where $V_1' = \{m_1', ..., m_n'\}$, $X = \{x_1, ..., x_{n-K}\}$, $Y = \{y_1, ..., y_n\}$ and $Z = \{z_1, ..., z_{n-K}\}$. For each vertex $m_i \in V_1$, there are two edges incident on m_i. Let $\{m_i, w_{j_i}\}$ and $\{m_i, w_{k_i}\}$ be those two edges, and assume $j_i < k_i$ $(1 \leq i \leq n)$. For each woman $w_j \in V_2$, let $M_j = \{m_i : \{m_i, w_j\} \in E\}$, and let $M_j' = \{m_i : \{m_i, w_j\} \in E \wedge j = k_i\}$. The preference lists of the agents in I are as follows:

$$
\begin{array}{rll}
m_i: & y_i\ w_{j_i}\ w_{k_i}\ z_1\ z_2...z_{n-K} & (1 \leq i \leq n) \\
m_i': & y_i\ w_{k_i} & (1 \leq i \leq n) \\
x_i: & w_1\ w_2...w_n & (1 \leq i \leq n-K) \\
w_j: & (M_j \cup M_j')\ (x_1...x_{n-K}) & (1 \leq j \leq n) \\
y_j: & (m_j\ m_j') & (1 \leq j \leq n) \\
z_j: & (m_1...m_n) & (1 \leq j \leq n-K)
\end{array}
$$

It is straightforward to verify that

$$(m_1\ m_2...m_n\ m_1'\ m_2'...m_n')\ (x_1\ x_2...x_{n-K})$$

is a master list of the men in I, and

$$y_1\ y_2...y_n\ w_1\ w_2...w_n\ z_1\ z_2...z_{n-K}$$

is a master list of the women in I. Clearly there are ties in only one master list. We set a target value of $K' = 3n - K$. Lemma 1 of [40] establishes that I has a stable matching of size K' if and only if G has a maximal matching of size K. The result for Maximum Cardinality ML-SMTI follows. □

the problem of determining, for a given graph $G = (V, E)$ and a positive integer K, whether there is a set S of edges of size at most K such that every edge in $E \setminus S$ is adjacent to some edge in S. There is a straightforward polynomial-time reduction from Minimum Maximal Matching to Exact Maximal Matching.

Note that the above result proves that the special case of Maximum Cardinality ML-SMTI in which the target is a complete stable matching is NP-complete, even if the ties occur in one master list only. We shall call this problem Complete ML-SMTI. For a number of transformations in succeeding sections, we wish to use as our starting point Complete SML-SMTI, the problem of finding a complete stable matching in an instance of SML-SMTI. This problem is a generalisation of Complete ML-SMTI, and so must be NP-complete, by Lemma 8.2.1. In addition, since Complete ML-SMTI is NP-complete even if the ties occur in one master list only, Complete SML-SMTI is NP-complete both when there are ties in the master list only, and when the master list is strictly ordered, but there are ties in the lists on the other side. We will make use of both of these cases in later results.

Let I be the instance of SMTI from Lemma 8.2.1. If we add a new man a_j to the end of the list of every woman $e_j \in Y \cup Z$ in I, and in addition create a new man b_j, and women c_j and d_j with lists as follows:

$$\begin{aligned} a_j &: \quad c_j \ d_j \ e_j \\ b_j &: \quad c_j \ d_j \\ c_j &: \quad a_j \ b_j \\ d_j &: \quad a_j \ b_j \end{aligned}$$

then it can be verified that this new instance has ties on the women's side only, and each woman has exactly two ties on her list (where a tie may be of length 1). Further, by adding all the c_j and then all the d_j to the head of the master list of women, and appending all the a_j and then all the b_j to the end of the master list of men, it can be shown that all the preference lists in this new instance are sublists of these extended master lists. Finally, consider the proof of Lemma 1 in [40]. Clearly $(a_j, c_j) \in M$, and $(b_j, d_j) \in M$ for any stable matching M in this new instance, hence this restriction of Maximum Cardinality SMTI is NP-complete. We use this restriction for the transformation in the next theorem. Note that the transformation is similar to that of Theorem 2 of [40].

Theorem 8.2.2. *Maximum Cardinality ML-SMTI is NP-complete, even if the ties occur in one master list only, and are of length 2.*

Proof Clearly Maximum Cardinality ML-SMTI is in NP. We transform from the version of Maximum Cardinality ML-SMTI described above, where the target value is the number

of men in the instance. Let I be an instance of the aforementioned problem, let $U = \{m_1, ..., m_n\}$ be the set of men in I, let $W = \{w_1, ..., w_n\}$ be the set of women in I, with master list $w_1\ w_2\ ...\ w_n$. Recall that every woman has exactly two ties on her list, and a tie can be of length 1. For each woman $w_j \in W$, $M_j^h = \{m_i \in U : m_i \in$ first tie on list of $w_j\}$ and $M_j^t = \{m_i \in U : m_i \in$ second tie on list of $w_j\}$. Let $M_j^h = \{m_{k_j,1}, ..., m_{k_j,h_j}\}$ and let $M_j^t = \{m_{l_j,1}, ..., m_{l_j,t_j}\}$ for some $h_j, t_j > 0$. We form an instance I' of Maximum Cardinality ML-SMTI as follows: let $U \cup \bigcup_{i=1}^n X_i$ be the set of men in I', and let $\bigcup_{j=1}^n W_j \cup \bigcup_{j=1}^n Y_j$ be the set of women in I', where

$$W_j = \{w_{j,r} : 1 \leq r \leq h_j + t_j\}\ (1 \leq j \leq n)$$

$$X_i = \{x_{i,r} : 1 \leq r \leq h_i + t_i\}\ (1 \leq j \leq n)$$

$$Y_j = \{y_{j,r} : 1 \leq r \leq h_j + t_j\}\ (1 \leq j \leq n).$$

Finally let $W_j^t = \bigcup_{r=h_j+1}^{h_j+t_j} \{w_{j,r}\}$. The preference lists in I' are formed as follows: each man $m_i \in U$ starts with his preference list from I. For each woman w_j on m_i's list in I, if $m_i \in M_j^h$ then $m_i = m_{k_j,a}$ for some a $(1 \leq a \leq h_j)$, while if $m_i \in M_j^t$ then $m_i = m_{l_j,b}$ for some b $(1 \leq b \leq t_j)$. In the former case we replace w_j by the women in $W_j^t \cup \{w_{j,a}\}$ in strict subscript order, while in the latter case we replace w_j by $w_{j,b+h_j}$. The remaining preference lists are as follows:

$$\begin{array}{rlll}
x_{i,r}: & (w_{i,r}\ y_{i,r}) & (1 \leq i \leq n) & (1 \leq r \leq h_i + t_i) \\
w_{j,r}: & x_{j,r}\ m_{k_j,r} & (1 \leq j \leq n) & (1 \leq r \leq h_j) \\
w_{j,r+h_j}: & x_{j,r+h_j}\ m_{k_j,1}\ m_{k_j,2}...m_{k_j,h_j}\ m_{l_j,r} & (1 \leq j \leq n) & (1 \leq r \leq t_j) \\
y_{j,r}: & x_{j,r} & (1 \leq j \leq n) & (1 \leq r \leq h_j + t_j)
\end{array}$$

Assume $\bigcup_{j=1}^n M_j^h = \{m_{a_1}, m_{a_2}, , m_{a_c}\}$ and $\bigcup_{j=1}^n M_j^t = \{m_{b_1}, m_{b_2}, , m_{b_d}\}$. Let the men in M_j^h be ordered such that if $m_{k_j,p} = m_{a_r}$ and $m_{k_j,q} = m_{a_s}$ then $p < q$ if and only if $r < s$. Then it is straightforward to verify that

$$x_{1,1}\ x_{1,2}...x_{1,h_1+t_1}\ x_{2,1}...x_{n,h_n+t_n}$$
$$m_{a_1}\ m_{a_2}...m_{a_c}\ m_{b_1}\ m_{b_2}...m_{b_d}$$

is a master list of the men in I', and

$$(w_{1,1}\ y_{1,1})\ (w_{1,2}\ y_{1,2})...(w_{1,h_1+t_1}\ y_{1,h_1+t_1})\ (w_{2,1}\ y_{2,1})...(w_{n,h_n+t_n}\ y_{n,h_n+t_n})$$

is a master list of the women in I'. Clearly there are ties in only one master list, they are of length 2, and any individual list has at most one tie and it occurs at the end of the list.

We show that I' has a stable matching in which all the men are matched if and only if I does.

Suppose I has such a matching M. We construct a matching M' exactly as in Theorem 2 of [40], except that, where the pair $(x_{j,a}, y_j)$ is added to M' in that theorem, we add $(x_{j,a}, y_{j,a})$. It is clear that all the men in I' are matched in M', hence any blocking pair for M' must be of the form $(m_i, w_{j,a})$. The rest of the proof is exactly as in Theorem 2 of [40]. □

We now give an inapproximability result. We show it is NP-hard to approximate Maximum Cardinality ML-SMTI within δ, for some $\delta < 1$, even if the preference lists in the given instance are of constant length and there is only one tie in each master list. The transformation is similar to that of Theorem 6 of [16].

Theorem 8.2.3. *It is NP-hard to approximate Maximum Cardinality ML-SMTI within δ, for some $\delta < 1$. The result holds even if the preference lists in the given instance are of constant length and there is only one tie in each master list.*

Proof We transform from Minimum Maximal Matching for subdivision graphs of cubic graphs, which is NP-complete [19]. Let $G = (V, E)$ be the subdivision graph of some cubic graph for an instance of Minimum Maximal Matching. Then G is a bipartite graph with vertex sets $V_1 \subseteq V$ and $V_2 \subseteq V$, every vertex in V_1 has degree 3 and every vertex in V_2 has degree 2. Let $V_1 = \{m_1, ..., m_s\}$ and let $V_2 = \{w_1, ..., w_t\}$. For each vertex $m_i \in V_1$, there are three edges incident on m_i. Let $\{m_i, w_{k_{3i-2}}\}$, $\{m_i, w_{k_{3i-1}}\}$ and $\{m_i, w_{k_{3i}}\}$ be those edges, and let $W_i = \{w_{k_{3i-2}}, w_{k_{3i-1}}, w_{k_{3i}}\}$. For each vertex $w_j \in V_2$, there are two

edges incident on w_j. Let $\{m_{p_j}, w_j\}$ and $\{m_{q_j}, w_j\}$ be those two edges, and assume $p_j < q_j$ $(1 \leq j \leq t)$. We construct an instance I of Maximum Cardinality SML-SMTI as follows: let $V_1 \cup X \cup Z$ be the set of men in I, and let $V_2 \cup V_2' \cup Y$ be the set of women, where $X = \{x_1, ..., x_t\}$, $Z = \{z_1, ..., z_t\}$, $V_2' = \{w_1', ..., w_t'\}$, and $Y = \{y_1, ..., y_s\}$. Further, let $W_i' = \{w'_{k_{3i-2}}, w'_{k_{3i-1}}, w'_{k_{3i}}\}$ $(1 \leq i \leq s)$. The preference lists in I are as follows:

$$\begin{array}{rll}
m_i: & (W_i \cup W_i')\ y_i & (1 \leq i \leq s) \\
x_i: & w_i & (1 \leq i \leq t) \\
z_i: & (w_i\ w_i') & (1 \leq i \leq t) \\
w_j: & z_j\ (m_{p_j}\ m_{q_j})\ x_j & (1 \leq j \leq t) \\
w_j': & z_j\ (m_{p_j}\ m_{q_j}) & (1 \leq j \leq t) \\
y_j: & m_j & (1 \leq j \leq s)
\end{array}$$

It is straightforward to verify that

$$z_1\ z_2...z_t\ (m_1...m_s)\ x_1\ x_2...x_t$$

is a master list of the men, and

$$(w_1\ w_2...w_t\ w_1'\ w_2'...w_t')\ y_1\ y_2...y_s$$

is a master list of the women in I. Theorem 6 of [16] shows that if G has a maximal matching of size $\beta_1^-(G)$[2], a maximum cardinality stable matching in I has size $s + 2t - \beta_1^-(G)$. For the proof that a maximum cardinality stable matching has size at most $s + 2t + \beta_1^-(G)$, it can be verified that the changes to the lists of w_j and w_j' do not invalidate the proof, because no strict preferences have been added to the instance. For the reverse proof, the blocking pairs that are constructed involve either w_j or w_j' being unmatched. In each case the changes to the lists of w_j and w_j' do not invalidate the proof. The result then follows from the final paragraphs of Theorem 6, and from Theorem 1 and Proposition 4 of [16]. □

[2]Here we use the notation of Theorem 6 of [16] for ease of reference.

8.3 Minimum Cardinality ML-SMTI

In this section we switch our attention to Minimum Cardinality ML-SMTI.

Theorem 8.3.1. *It is NP-hard to approximate Minimum Cardinality SML-SMTI within δ, for some $\delta > 1$. The result holds even if the preference lists in the given instance are of constant length and the only tie is at the tail of the master list.*

Proof The construction is as for Theorem 8.2.3, except that the people in $X \cup Y$ are not included. Theorem 7 of [16] shows that if G has a maximal matching of size $\beta_1^-(G)$, a minimum cardinality stable matching in I has size $t + \beta_1^-(G)$. For the proof that a maximum cardinality stable matching has size at most $t + \beta_1^-(G)$, it can be verified that the changes to the lists of w_j and w'_j do not invalidate the proof, because no strict preferences have been added to the instance. For the reverse proof, the blocking pairs that are constructed involve either w_j or w'_j being unmatched. In each case the changes to the lists of w_j and w'_j clearly do not invalidate the proof. □

It remains open as to whether Minimum Cardinality ML-SMTI is NP-complete if the ties occur in one master list only, and are of length 2. Minimum Cardinality SMTI is NP-complete, even if the ties occur at the tails of the lists and are on one side only, there is at most one tie per list, and each tie is of length 2 [40], but the proof of that result cannot be directly extended to ML-SMTI.

8.4 Egalitarian ML-SMT

We now show that Egalitarian ML-SMT can be solved in linear time, but Egalitarian SML-SMT is NP-complete, even if the master list is strict.

Theorem 8.4.1. *Egalitarian ML-SMT can be solved in $O(n)$ time, for an instance I of size n.*

Proof For every position i on one master list, some person from the other master list must be matched with the person at position i. Thus every stable matching for I must have the same weight.

Break the ties on the master lists arbitrarily, and match the man at position i on the master list of men with the woman at position i on the master list of women. Clearly this matching must be stable, and can be found in $O(n)$ time. By the foregoing it is also an egalitarian stable matching, and the result follows. □

Theorem 8.4.2. *Egalitarian SML-SMT is NP-complete.*

Proof Clearly Egalitarian SML-SMT is in NP. We transform from Complete SML-SMTI. Let I be an instance of Complete SML-SMTI where there is a master L_m list of the men. Let $U = \{m_1, ..., m_n\}$ and $V = \{w_1, ..., w_n\}$ be the sets of men and women in I respectively, and let P_i be the preference list of m_i in I. We construct an instance I' of Egalitarian SML-SMT as follows: the set of men in I' is $U \cup X$ where $X = \{m'_1, ..., m'_{n^2}\}$; the set of women in I' is $V \cup Y$, where $Y = \{w'_1, ..., w'_{n^2}\}$. The preference lists for the men are as follows:

$$\begin{aligned} m_i: &\quad P_j\ [Y] - &\quad (1 \leq i \leq n) \\ m'_i: &\quad w'_i - &\quad (1 \leq i \leq n^2) \end{aligned}$$

while the master list of men becomes:

$$L'_m : m'_1 ... m'_{n^2}\ L_m$$

We give all women the same list, namely L'_m, so all lists are complete. Note also that there are the same number of men and women in I', so every person must be matched in every stable matching in I'. We show that I admits a complete stable matching if and only if I' has a stable matching M' where the weight of M', $w(M')$, is at most $\frac{(n^2+n)(n^2+n+1)}{2} + 2n^2$.

Suppose I admits a complete stable matching M. We create a matching $M' = M \cup \bigcup_{i=1}^{j}(m'_i, w'_i)$ in I'. Clearly no man in X can be involved in a blocking pair for M'. Suppose (m_j, w_l) blocks M'. Since every woman w_k on P_j who strictly precedes $p_M(m_j)$ is on the list of m_j in I, m_j is on the list of w_k in I. It follows that (m_j, w_l) blocks M, a contradiction. Hence M' is stable. Each of the n^2 men in X contributes 1 to the weight of M', while each of the n men in U contributes at most n to the weight of M'. Finally, there are $n^2 + n$ women in I', so in the worst case, where there are no

ties in L'_m, the women contribute $\frac{(n^2+n)(n^2+n+1)}{2}$ to the weight of M'. It follows that $w(M') \leq \frac{(n^2+n)(n^2+n+1)}{2} + 2n^2$.

Now suppose I does not admit a complete stable matching. Let M' be an arbitrary stable matching in I'. Then there is some man m_j who is not matched with a woman from P_j. Suppose m_j is matched with a woman $w'_i \in Y$. Then (m'_i, w'_i) blocks M', a contradiction. Hence m_j must contribute at least $n^2 + 2$ to the weight of M'. Each of the remaining $n^2 + n - 1$ men must contribute at least 1 to the weight of M'. Thus $w(M') \geq \frac{(n^2+n)(n^2+n+1)}{2} + 2n^2 + n + 1$. Since $2n^2 + n + 1 > 2n^2$ for all n, the result follows. □

Recall that Complete ML-SMTI is NP-complete even if the ties are on one side only. Complete SML-SMTI is a generalisation of this problem regardless of which side the ties are on. It is therefore clear that the above proof holds if there are ties in the master list only, or if there are ties only in the lists on the other side, with only minor adaptations in the former case.

It remains open as to whether Egalitarian SML-SMT is NP-hard to approximate. Egalitarian SMT is NP-hard to approximate within δn, for some $\delta > 0$, where n is the number of men in the given instance [16], but the proof of that result cannot be readily extended to SML-SMT. Egalitarian SMT is not approximable within $N^{1-\epsilon}$, unless P=NP, for any $\epsilon > 0$, where N is the number of men in a given instance of the problem, even if the ties are on one side only and of length 2 [40], but again the proof of that result cannot be readily extended to SML-SMT. Indeed, in an instance of Egalitarian SML-SMT of size n in which the master list is strict it is the case that, for any stable matching M,

$$\frac{1}{2}n^2 + \frac{3}{2}n = n + \frac{n(n+1)}{2} \leq w(M) \leq n^2 + \frac{n(n+1)}{2} = \frac{3}{2}n^2 + \frac{1}{2}n.$$

Hence Egalitarian SML-SMT is approximable within a factor of 3 when the master list is strict.

8.5 Minimum Regret problems

First, we document two solvable variants of Minimum Regret (S)ML-SMT.

Theorem 8.5.1. *Minimum Regret ML-SMT can be solved in $O(n)$ time, for an instance of size n.*

Proof Since every person must be matched in every stable matching, it follows that the members of the final ties on the two master lists must be matched in every stable matching. Thus every stable matching must have the same regret. Break the ties on the master lists arbitrarily, and match the man at position i on the resolved master list of men with the woman at position i on the resolved master list of women. Clearly this matching must be stable, can be found in $O(n)$ time, and is of minimum regret, by the foregoing argument. □

Theorem 8.5.2. *Minimum Regret SML-SMT can be solved in $O(n^2)$ time, if there is no tie at the tail of the master list, for an instance of size n.*

Proof Every person must be matched in every stable matching. Since there is a unique person at the end of the master list, their partner must have regret n in every stable matching. Break all the ties arbitrarily and find a stable matching for the derived instance of SM. This matching must be stable in the initial instance, and can be found in $O(n^2)$ time. □

So, when the master list has no tie at the tail, Minimum Regret SML-SMT can be solved in time linear in the input size. However, if there is a tie at the tail of the master list, we show that this problem is NP-complete, even if there are no ties on the other side.

Theorem 8.5.3. *Minimum Regret SML-SMT is NP-complete if there is a tie t the tail of the master list, even if there are no ties in the lists on the other side.*

Proof Clearly Minimum Regret SML-SMT is in NP. To show the problem is NP-hard, we transform from Complete SML-SMTI. Recall that the latter problem is NP-complete even when the ties are on one side only. Here we assume that there is a master list of the men, L_m say, and there are no ties in the lists on the other side. Let I be an instance of Complete SML-SMTI, and let L_m be the master list of men. Let $U = \{m_1, ..., m_n\}$ and $V = \{w_1, ..., w_n\}$ be the sets of men and women in I respectively, and let P_i be the preference list of m_i in I. We construct an instance I' of Minimum Regret SML-SMT as follows: let $M^j = \{m_1^j, ..., m_n^j\}$ $(1 \leq j \leq n+1)$, and let $W^j = \{w_1^j, ..., w_n^j\}$ $(1 \leq j \leq n+1)$. Then the set of men in I' is $U \cup \bigcup_{j=1}^{n+1} M^j$, and the set of women in I' is $V \cup \bigcup_{j=1}^{n+1} W^j$.

The preference lists for the men are as follows (where P_i^j is the list obtained by replacing $w \in P_i$ with w^j):

$$\begin{array}{lll} m_i: & P_i - & (1 \leq i \leq n) \\ m_i^j: & P_i^j \ w_1 ... w_n - & (1 \leq i \leq n) \quad (1 \leq j \leq n+1) \end{array}$$

while the master list of men becomes:

$$L'_m : L_m(\bigcup_{j=1}^{n+1} M^j)$$

We give all women the same list, namely L'_m, so all lists are complete. Further, there are no ties in the men's lists. Note also that there are the same number of men and women in I', so every person must be matched in every stable matching in I'. We show that I admits a complete stable matching if and only if I' has a stable matching of regret at most $n+1$.

Suppose I admits a complete stable matching M. We construct a matching M' in I' as follows: suppose, without loss of generality, that $w_i = p_M(m_i)$. For each man m_i we add (m_i, w_i) to M' $(1 \leq i \leq n)$. For each man m_i^j, we add (m_i^j, w_i^j) to M'. Clearly any blocking pair for M' is of the form (m_i, w_k). Since every woman on P_i who strictly precedes $p_M(m_i)$ is on the list of m_i in I, m_i is on the list of w_k in I. It follows that (m_i, w_k) blocks M, a contradiction. Hence M' is stable. Finally, every man has a partner in M from among the first n women in his list, and so has regret at most n. It follows that the regret of M' is $n+1$.

Now suppose I does not admit a complete stable matching. Let M' be an arbitrary stable matching in I'. Then, for every j $(1 \leq j \leq n+1)$, there is some m_i^j who is not matched with a woman from P_i^j. Thus there are at least $n+1$ men m_i^j who are not matched with a woman from P_i^j. At best n of these men are matched in M' with one of the next n women on their list, since w_1,...,w_n are the next n women on the lists of every one of these men. Thus there is some man $m_i^j \in M^j$ for some $1 \leq i \leq n$ and $1 \leq j \leq n+1$ who is matched in M' with a woman who appears after w_n in his list. But then, since P_i^j has length at least 1, m_i has regret at least $n+2$, and the result follows. □

We are left with one case to consider. If the master list is of men, then can we efficiently

find a man minimum regret stable matching? The following result shows that this problem is NP-hard, even if the master list is strictly ordered.

Theorem 8.5.4. *Man Minimum Regret SML-SMT is NP-complete if the master list is of men, even if it is strictly ordered.*

Proof Clearly Man Minimum Regret SML-SMT is in NP. To show the problem is NP-hard, we transform from Complete SML-SMTI. Recall that the latter problem is NP-complete even when the ties are on one side only. Here we assume that there is a master list of the men, L_m say, and the master list is strictly ordered. Let I be an instance of Complete SML-SMTI, and let L_m be the master list of men. Let $U = \{m_1, ..., m_n\}$ and $V = \{w_1, ..., w_n\}$ be the sets of men and women in I respectively, and let P_i be the preference list of m_i in I. We construct an instance I' of Man Minimum Regret SML-SMT as follows: the set of men in I' is $U \cup U'$, and the set of women in I' is $V \cup V'$, where $U' = \{m'_1, ..., m'_n\}$ and $V' = \{w'_1, ..., w'_n\}$. The preference lists for the men are as follows:

$$\begin{aligned} m_i: &\quad P_i\ w'_1 ... w'_n - &\quad (1 \leq i \leq n) \\ m'_i: &\quad w'_i - &\quad (1 \leq i \leq n) \end{aligned}$$

while the master list of men becomes:

$$L'_m : m'_1\ m'_2 ... m'_n\ L_m$$

We give all women the same list, namely L'_m, so all lists are complete. Further, there are no ties in the master list. Note also that there are the same number of men and women in I', so every person must be matched in every stable matching in I'. We show that I admits a complete stable matching if and only if I' has a stable matching in which no man has regret greater than n.

Suppose I admits a complete stable matching M. Let $M' = M \cup \bigcup_{i=1}^{n} (m'_i, w'_i)$. Clearly no man from U' can be in a blocking pair. Suppose, without loss of generality, that (m_i, w_j) $(1 \leq i, j \leq n)$ blocks M'. Then clearly (m_i, w_j) blocks M, a contradiction. Finally, it is clear that no man has regret greater than n.

Now suppose I does not admit a complete stable matching. Let M' be an arbitrary stable matching in I'. Then some man m_i $(1 \leq i \leq n)$ must be matched with a woman who is not

in P_i. Suppose m_i is matched with a woman from V', w'_j say. Then (m'_j, w'_j) blocks M', a contradiction. Thus m_i must be matched with a woman from V who does not appear in P_i, and so m_i has regret at least $n + 2$, since P_i must have length at least 1. The result follows. □

It remains open as to whether Minimum Regret SML-SMT is NP-hard to approximate. Minimum Regret SMT is NP-hard to approximate within δn, for some $\delta > 0$, where n is the number of men in the given instance [16], but the proof of that result cannot be readily extended to SML-SMT. Minimum Regret SMT is not approximable within $N^{1-\epsilon}$, unless P=NP, for any $\epsilon > 0$, where N is the number of men in a given instance of the problem, even if the ties are on one side only and of length 2 [40], but again the proof of that result cannot be readily extended to SML-SMT.

8.6 Stable Pair problems

We start with two NP-completeness proofs, before showing that some versions of the Stable Pair problem are solvable. First we consider Stable Pair of ML-SMTI, then we consider Stable Pair of SML-SMT.

Theorem 8.6.1. *Stable Pair of ML-SMTI is NP-complete, even if there are ties in only one of the master lists.*

Proof Clearly Stable Pair of ML-SMTI is in NP. To show the problem is NP-hard, we transform from Complete ML-SMTI. Let I be an instance of Complete ML-SMTI, and let L_m and L_w be the master lists of the men and women respectively. Recall that Complete ML-SMTI is NP-complete even if the ties are on one side only, so we additionally assume there are no ties on the men's side. Let $U = \{m_1, ..., m_n\}$ and $V = \{w_1, ..., w_n\}$ be the sets of men and women in I respectively, and let P_i and Q_i be the preference lists of m_i and w_i in I respectively. We construct an instance I' of Stable Pair of ML-SMTI as follows: the set of men in I' is $\{m_0\} \cup U$, and the set of women in I' is $\{w_0\} \cup V$. The preference lists for each person are as follows:

$$\begin{aligned} m_0 &: \quad L_w\ w_0 \\ m_i &: \quad P_i\ w_0 \qquad (1 \leq i \leq n) \\ w_0 &: \quad L_m\ m_0 \\ w_i &: \quad Q_i\ m_0 \qquad (1 \leq i \leq n) \end{aligned}$$

and the master lists are

$$L'_m : L_m\ m_0$$

$$L'_w : L_w\ w_0$$

Clearly there are no ties in the men's lists. We show that I admits a complete stable matching if and only if I' has a stable matching M' containing (m_0, w_0).

Suppose I admits a complete stable matching M. We define a matching $M' = M \cup \{(m_0, w_0)\}$ in I'. Any pair (m_i, w_j) $(i, j \neq 0)$ blocking M' must also block M, a contradiction. Finally, every man $m_i \neq m_0$ prefers his partner in M' to w_0, and every woman $w_i \neq w_0$ prefers her partner in M' to m_0, so M' is stable.

Now suppose I does not admit a complete stable matching. Let M' be an arbitrary stable matching in I' containing (m_0, w_0). Then there is at least one man m_i in M' who is not matched to a woman from P_i. Since m_i is not matched with w_0, he is unmatched in M'. Then w_0 prefers m_i to m_0, her partner in M', and (m_i, w_0) blocks M'. The result follows. □

Theorem 8.6.2. *Stable Pair of SML-SMT is NP-complete.*

Proof Clearly Stable Pair of SML-SMT is in NP. To show the problem is NP-hard, we transform from Complete SML-SMTI. Let I be an instance of Complete SML-SMTI where there is a master list L_m of the men. Let $U = \{m_1, ..., m_n\}$ and $V = \{w_1, ..., w_n\}$ be the sets of men and women in I respectively, and let P_i be the preference list of m_i in I. We construct an instance I' of Stable Pair of SML-SMT as follows: the set of men in I' is $\{m_0\} \cup U$, and the set of women in I' is $\{w_0\} \cup V$. The preference lists for the men are as follows:

$$m_0 : \quad - \; w_0$$
$$m_i : \quad P_i \; w_0 - \quad (1 \leq i \leq n)$$

while the master list of men becomes:

$$L'_m : L_m \; m_0$$

We give every woman a complete list, namely L'_m, so all lists are complete. Note also that there are the same number of men and women in I', so every person must be matched in every stable matching in I'. We show that I admits a complete stable matching if and only if I' has a stable matching M' containing (m_0, w_0).

Suppose I admits a complete stable matching M. We define a matching $M' = M \cup \{(m_0, w_0)\}$ in I'. Suppose (m_i, w_j) $(i, j \neq 0)$ blocks M'. Then m_i prefers w_j to $p_{M'}(m_i) = p_M(m_i)$, hence w_j appears on P_i and m_i appears on the list of w_j in I. Since w_j prefers m_i to $p_{M'}(w_j) = p_M(w_j)$, it follows that (m_i, w_j) blocks M, a contradiction. Finally, every man $m_i \neq m_0$ prefers his partner in M' to w_0, and every woman $w_i \neq w_0$ prefers her partner in M' to m_0, so M' is stable.

Now suppose I does not admit a complete stable matching. Let M' be an arbitrary stable matching in I' containing (m_0, w_0). Then there is at least one man m_i in M' who is matched to a woman who is not in P_i. Since m_i is not matched with w_0, he prefers w_0 to his partner in M'. Clearly w_0 prefers m_i to her partner in M', so (m_i, w_0) blocks M', a contradiction. The result follows. □

Recall that Complete ML-SMTI is NP-complete even if the ties are on one side only. It is therefore clear that the above proof holds if there are ties in the master list only, or if there are only ties in the lists on the other side.

We now show that Stable Pair of ML-SMT can be solved in linear time, and that we can find all the stable pairs in time linear in the number of such pairs and the size of the instance. Let L_m and L_w be the master lists for a given instance of ML-SMT. Henceforth we assume, without loss of generality, that the men on L_m are indexed so that man m_i appears before man m_j if and only if $i < j$, and similarly for the women on L_w. Note that this includes agents within the same tie. We say that a tie $t_m \in L_m$ *overlaps* a tie $t_w \in L_w$

if and only if there is some i such that $m_i \in t_m$ and $w_i \in t_w$, and for every such i we say that m_i and w_i are *in the overlap* between t_m and t_w.

Lemma 8.6.3. *Let I be an instance of ML-SMT. The pair (m, w) is a stable pair in I if and only if the master lists can be resolved so that m and w occupy the same position in their respective master lists.*

Proof Suppose the master lists can be resolved so that m and w occupy the same position in their respective master lists. Then we match the man at position i with the woman at position i $(1 \leq i \leq n)$, where n is the size of the instance. Clearly this matching is stable.

Conversely, suppose the master lists cannot be resolved so that m and w occupy the same position in their respective master lists. Suppose M is a stable matching containing (m, w), and suppose, without loss of generality, that w occupies a lower position in m's list than m does in w's, regardless of how the ties are resolved. Resolve the ties in the master list of women so that w is preferred to every woman with whom she is tied, and resolve the tie in the master list of men so that every man with whom m is tied is preferred to m. Let W be the set of women who are strictly better than w. Then, if w occupies position i and m occupies position j, we have $i > j$ and $|W| = i - 1$. Hence there is at least one women $w' \in W$ such that w' is matched in M with a man who is strictly worse than m. It follows that w' prefers m to $p_M(w')$. But, by definition of W, m prefers w' to $w = p_M(m)$, and so (m, w') blocks M, a contradiction. □

Theorem 8.6.4. *Stable Pair of ML-SMT can be solved in $O(n)$ time.*

Proof By Lemma 8.6.3, a given pair (m, w) is stable if and only if the master lists can be resolved so that m and w occupy the same position in their respective master lists. We can find m on the master list of men and ascertain the extent of the tie in which m appears in $O(n)$ time, and similarly for w. Then, if there is an overlap between these ties, which can be ascertained in constant time, (m, w) is a stable pair, otherwise it is not. □

Theorem 8.6.5. *For an instance I of ML-SMT we can find all the stable pairs in $O(n+s)$ time, where s is the number of stable pairs.*

Proof Let L_m and L_w be the master lists for I. By Lemma 8.6.3, a given pair (m, w) is stable if and only if the master lists can be resolved so that m and w occupy the same

position in their respective master lists. Consider a tie $t \in L_m$. We can find the stable partners of each man in t as follows: let $j_1 = min\{l : m_l \in t\}$, and let $k_1 = max\{l : m_l \in t\}$. Let $t_1 \in L_w$ be such that $w_{j_1} \in t_1$, and let $t_2 \in L_w$ be such that $w_{k_1} \in t_2$. Let $j_2 = min\{l : w_l \in t_1\}$, and let $k_2 = max\{l : w_l \in t_2\}$. Then the stable partners of the men in t are $w_{j_2}, ..., w_{k_2}$. Finding j_1 and k_1 takes $O(|t|)$ time, while finding j_2 and k_2 takes $O(s_1)$ time, where s_1 is the number of stable partners of the men in t. If we repeat this process for every tie in L_m then clearly we can find all the stable pairs in $O(n + s)$ time. □

8.7 Generation of all stable matchings for an instance of ML-SMT

We complete this chapter by showing that we can find all the weakly stable matchings for an instance I of ML-SMT, with sublinear time between the generation of successive matchings. As noted in Section 8.1, ML-SMT is the first version of SMTI for which we can generate the stable matchings in time polynomial in the number of such matchings.

Let I be an instance of ML-SMT of size n, and let $U = \{m_1, ..., m_n\}$ and $V = \{w_1, ..., w_n\}$ be the set of men and women in I respectively. By Theorem 8.6.5, we can list the stable pairs for I in $O(n+s)$ time, where s is the number of stable pairs. We construct a bipartite graph G_I, the *matching graph*, as follows. The set of vertices in G_I is $U \cup V$, and there is an edge from $m_i \in U$ to $w_j \in V$ if and only if w_j is a stable partner of m_i. This construction takes $O(n + s)$ time. We show that there is a one-to-one correspondence between the perfect matchings in G_I and the stable matchings for I.

Lemma 8.7.1. *Let I be an instance of ML-SMT. Then there is a one-to-one correspondence between the perfect matchings in the matching graph G_I and the stable matchings for I.*

Proof Let $M = \{(m_1, w_{k_1}), ..., (m_n, w_{k_n})\}$ be a stable matching for I. Then there is an edge in G_I from m_i to w_{k_i} $(1 \leq i \leq n)$, and, by the definition of a matching, $w_{k_i} \neq w_{k_j}$ $(i \neq j)$. Hence there is a perfect matching in G corresponding to M.

Conversely, let $M = \{(m_1, w_{k_1}), ..., (m_n, w_{k_n})\}$ be a perfect matching in G_I, and suppose M is not stable. Let (m, w) be a blocking pair for M. Then w prefers m to $p_M(w)$, so

m appears before $p_M(w)$ on the master list of men. Since $(m, p_M(m)) \in M$, the tie on the master list of men containing m must overlap with that on the master list of women containing $p_M(m)$, by Lemma 8.6.3. Similarly, m prefers w to $p_M(m)$, so w appears before $p_M(m)$ on the master list of women. Since $(p_M(w), w) \in M$, the tie on the master list of women containing w must overlap with that on the master list of men containing $p_M(w)$, by Lemma 8.6.3. It is clear that at most three of these four conditions can be satisfied, a contradiction. The result follows. □

Uno [53] gives an algorithm which, given an initial perfect matching in a bipartite graph G, can generate all k perfect matchings for G in $O(k \log |V|)$ time. By Lemma 8.6.3, the matching produced by breaking the ties on the two master lists arbitrarily, and then matching the man at position i with the woman at position i, for each $1 \leq i \leq n$, is weakly stable, and can be produced in $O(n)$ time. Then, using Uno's algorithm, the rest of the perfect matchings in G_I can be generated in $O(\log n)$ time per matching, giving overall complexity $O(n + s + k \log n)$ to generate the k prefect matchings in G_I which, by Lemma 8.7.1 are exactly the weakly stable matchings for I. Thus we get the following theorem.

Theorem 8.7.2. *Let I be an instance of ML-SMT of size n. Then we can generate all the weakly stable matchings for I in $O(n + s + \log(n)k)$ time, where k is the number of such matchings, and s is the number of stable pairs in I.*

Chapter 9

Open Problems

9.1 Introduction

The original work in this thesis can be divided into two broad strands. In Chapters 2, 3, 4 and 5 we have presented algorithms for determining whether there is a matching which satisfies one of the four stability criteria, and if there is the algorithms output one. On the other hand, in Chapters 6, 7 and 8 we have presented results relating to the structure of particular types of instance or matchings satisfying both a particular stability criterion and some additional property. We discuss the issues relating to these two areas in Sections 9.2 and 9.3 respectively.

9.2 Algorithms

The table in Figure 9.2 gives the complexities of the best known algorithms for determining whether an instance of each of the main variants of Stable Marriage admits a stable matching. In every solvable case the algorithm also finds such a matching. In the table, a is the number of acceptable pairs in an instance, k is the total number of men and women in an SM(T)I instance, and $l = |R| + \sum_{h \in H} p_h$, where R and H are the sets of residents and hospitals in an instance of HR(T), and p_h is the quota of hospital h.

As can be seen, the only one of the main variants which remains open is determining whether an instance of SFT admits a strongly stable matching. In Chapter 5 we conjec-

Problem	Stability	Super-Stability	Strong Stability	Weak Stability
SM(T)I	$O(a)$ [8]	$O(a)$ [22, 34]	$O(ka)$ [30]	$O(a)$ [10]
HR(T)	$O(a)$ [8, 15]	$O(a)$ [27]	$O(la)$ [30]	$O(a)$ [10]
SR(T)I	$O(a)$ [20]	$O(a)$ [26]	$O(a^2)$ Ch.3	NP-complete [46]
SF(T)	$O(a)$ Ch.4	$O(a)$ Ch.5	Open	NP-complete [46]

Figure 9.1: Stable Marriage complexities

tured that this problem is NP-complete, which would break the pattern established thus far.

9.2.1 Approximation algorithms

We have noted that weakly stable matchings may vary in size for instances of each of SMTI, SRTI, HRT, and SFT, but in each case the largest is at most twice the size of the smallest. In Section 7.3 we presented an approximation algorithm for finding a maximum cardinality weakly stable matching in an instance of SMTI, and there are the results of Halldórsson et al. ([18], [17], [16], see Section 1.6), and of Iwama et al. [29]. There is, however, still a significant gap between the best approximation algorithm, which is in any case for a restricted set of instances, and the bound in the inapproximability result of [16], so there is scope for approximation algorithms with improved guarantees. Additionally, the problem of approximating a maximum cardinality weakly stable matching in HRT with guarantee better than the trivial value of 2 remains open. In SRTI and SFT, of course, determining whether a weakly stable matching exists is NP-complete.

9.2.2 Master Lists

A number of the results in Chapter 8 were positive. It remains open as to whether any of these results can be extended to HRT. We also showed that we can generate all the weakly stable matchings for an instance of ML-SMT. The next step is to ask if we can generate all the weakly stable matchings for an instance of SML-SMT or ML-SMTI. At this time, though, a reasonable algorithm to solve either of the problems is not in sight.

9.3 Structure

There are many more open questions of interest when we consider structural results for the main variants of Stable Marriage. Here we discuss a subset of these.

9.3.1 The lattice structure

It is known that the set of strongly stable matchings in an instance I of SMTI forms a finite distributive lattice, when the matchings are partitioned into suitable equivalence classes [38]. It is certainly possible to formulate an algorithm in the manner of Algorithm POSET2 to find all the strongly stable matchings (or possibly equivalence classes of matchings) contained within the lattice for an instance I' of SMI obtained from I by breaking the ties. However, unlike super-stability, it is not the case that a strongly stable matching is stable in every instance of SMI obtainable from I. The next question to ask is whether there exists an instance I' of SMI obtainable from I in which at least one matching from each equivalence class of strongly stable matchings is stable? We conjecture that this is not necessarily the case, though even if it is, there would still be the question of finding the correct way of breaking the ties to find the appropriate instance. Also, little work has been done on the equivalent problems in HR(T).

In contrast to the bipartite problems, the set of stable matchings, for an instance of SR which admits a stable matching, forms a semi-lattice. It remains open to determine whether the set of stable allocations, for an instance of SF which admits a stable allocation, forms a semi-lattice. The same question is open for the set of super-stable (resp. strongly stable) matchings, for an instance of SRTI which admits a super-stable (resp. strongly stable) matching. A related open question is if we can determine in polynomial-time whether a given pair of agents is either super-stable or strongly stable?

Finally there is SMTI under weak stability, which does not appear to admit any lattice structure. We have shown that it is possible to find two distinct weakly stable matchings for an instance of SMTI, if such exist. However, short of generating every possible matching for the instance and verifying whether it is weakly stable, we do not know how to generate all the weakly stable matchings. Is there an algorithm for doing this which has polynomial-time between the generation of successive matchings? We touch on this problem again in

the next paragraph.

9.3.2 Interpolating invariants

We noted in Section 1.6 that weak stability is an interpolating invariant in SMTI, i.e., for an instance I of SMTI, there is a matching of size k, weakly stable with respect to I, for each $p \leq k \leq q$, where p is the size of a minimum cardinality weakly stable matching in I and q is the size of a maximum cardinality weakly stable matching in I. We could ask the same question of weakly stable matchings in instances of SRTI, HRT and SFT. We conjecture that the result for SMTI will extend to all of the more general cases, but at this time the question is open for all three.

9.3.3 Some miscellaneous problems

Here we list a few open problems of interest which do not fit easily into any of the foregoing sections.

- For an instance of SF in which the preference lists are complete, what is the smallest possible size of a stable allocation (expressed in terms of the size of the instance, n, and the capacities)?
- There is an algorithm for finding, in an instance of SR, a maximum cardinality matching such that the matched agents are stable within themselves [52]. Can this be extended to SF? Indeed we can consider these questions for SMTI under strong stability or super-stability, or for any other variant of Stable Marriage which may not admit a certain type of stable matching.

Bibliography

[1] M. Baïou and M. Balinski. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics*, 101:1–12, 2000.

[2] Canadian Resident Matching Service. How the matching algorithm works. Web document available at `http://www.carms.ca/matching/algorith.htm`.

[3] Katarina Cechlarova and Tamas Fleiner. On a generalization of the stable roommates problem. Technical Report 2003-03, Egervary Group for Research on Combibatorial Optimization, May 2003.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction To Algorithms*. MIT Press, 1990.

[5] V.M.F. Dias, G.D. da Fonseca, C.M.H. de Figueiredo, and J.L. Szwarcfiter. The stable marriage problem with restricted pairs. *Theoretical Computer Science*, 306:391–405, 2003.

[6] T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45:233–284, 1992.

[7] H.N. Gabow. An efficient reducction technique for degree-constrained subgraph and bidirected network flow problems. *Proceedings, 15th Annual ACM Symposium on Theory of Computing*, pages 448–456, 1983.

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

[9] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

[10] P. Gardenfors. Match making: assignments based on bilateral preferences. *Behavioural Sciences*, 20:166–173, 1975.

[11] M.R. Garey and D.S. Johnson. *Computers and Intractibility.* Freeman, 1979.

[12] D. Gusfield. The structure of the stable roommate problem. Technical Report DCS/TR 482, Yale University, 1986.

[13] D. Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987.

[14] 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.

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

[16] M. Halldórsson, R.W. Irving, K. Iwama, D.F. Manlove, S. Miyazaki, Y. Morita, and S. Scott. Approximability results for stable marriage problems with ties. *Theoretical Computer Science*, 306:431–447, 2003.

[17] M Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved Approximation of the Stable Marriage Problem. In *Proceedings of ESA '03: the 11th Annual European Symposium on Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 266–277. Springer-Verlag, 2003.

[18] M Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Randomized Approximation of the Stable Marriage Problem. In *Proceedings of COCOON '03: the 9th Annual International Conference on Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 339–350. Springer-Verlag, 2003.

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

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

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

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

[23] R.W. Irving. Matching medical students to pairs of hospitals: a new variation on a well-known theme. In *Proceedings of ESA '98: the Sixth European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 381–392. Springer-Verlag, 1998.

[24] R.W. Irving and P. Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

[25] R.W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the Association for Computing Machinery*, 34(3):532–543, 1987.

[26] R.W. Irving and D.F. Manlove. The stable roommates problem with ties. *Journal of Algorithms*, 43(1):85–105, 2002.

[27] 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.

[28] R.W. Irving, D.F. Manlove, and S. Scott. Strong Stability in the Hospitals/Residents Problem. In *Proceedings of STACS 2003: the 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 439–450. Springer-Verlag, 2003.

[29] K. Iwama, S. Miyazaki, and K. Okamoto. A $(2 - (clog(n))/n)$?approximation algorithm for the stable marriage problem. In *Proceedings of SWAT 2004: 9th Scandinavian Workshop on Algorithm Theory*, volume 3111 of *Lecture Notes in Computer Science*, pages 349–361. Springer-Verlag, 2004.

[30] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Strongly Stable Matching in Time $O(nm)$ and Extension to the Hospitals-Residents Problem. In *Proceedings of STACS 2004: the 21st Annual Symposium on Theoretical Aspects of Computer Science*, volume 2996 of *Lecture Notes in Computer Science*, pages 222–233. Springer-Verlag, 2004.

[31] 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.

[32] G.M. Low. Matching medical students to hospital posts in the West of Scotland. Master's thesis, University of Glasgow, Department of Computing Science, 1997.

[33] V.S. Malhotra. On the stability of multiple partner stable marriages with ties. To appear in *Proceedings of the 12th Annual European Symposium on Algorithms, volume 3221 of* Lecture Notes in Computer Science, 2004.

[34] D.F. Manlove. Stable marriage with ties and unacceptable partners. Technical Report TR-1999-29, University of Glasgow, Department of Computing Science, January 1999.

[35] D.F. Manlove. A new characterisation of super-stable and strongly stable matchings. Unpublished manuscript, 2000.

[36] D.F. Manlove. "Cloning" hospitals. Unpublished manuscript, 2000.

[37] D.F. Manlove. Strongly stable matchings in HRT. Unpublished manuscript, 2001.

[38] D.F. Manlove. The structure of stable marriage with indifference. *Discrete Applied Mathematics*, 122:167–181, 2002.

[39] D.F. Manlove. Personal communication, 2004.

[40] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002.

[41] D. McVitie and L.B. Wilson. Stable marriage assignment for unequal sets. *BIT*, 10:295–309, 1970.

[42] D. McVitie and L.B. Wilson. The stable marriage problem. *Communications of the A.C.M.*, 14:486–490, 1971.

[43] M. Mukerjee. Medical mismatch. *Scientific American*, 276(6):40–41, 1997.

[44] National Resident Matching Program. How the matching algorithm works. Web document available at `http://www.nrmp.org/res_match/about_res/algorithms.html`.

[45] J.C. Picard. Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 11:1268–1272, 1976.

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

[47] 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.

[48] A.E. Roth. On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica*, 54:425–427, 1986.

[49] A.E. Roth and M. Sotomayor. The college admissions problem revisited. *Econometrica*, 57:559–570, 1989.

[50] Scottish PRHO Allocations Scheme. The Scottish PRHO Allocation Scheme. Web document available at `http://www.nes.scot.nhs.uk/spa`.

[51] B. Spieker. The set of super-stable marriages forms a distributive lattice. *Discrete Applied Mathematics*, 58:79–84, 1995.

[52] J.J.M. Tan. A maximum stable matching for the roommates problem. *BIT*, 29:631–640, 1990.

[53] T. Uno. A fast algorithm for enumerating bipartite perfect matchings. In *Proceedings of ISAAC 2001: the 12th Annual International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, pages 367–379. Springer-Verlag, 2001.