University of Glasgow | Department of Computing Science

A Structural Approach to Matching Problems with Preferences

by

Eric McDermid

Submitted for the
Degree of
Doctor of Philosophy
University of Glasgow

May 7 2010

Abstract

This thesis is a study of a number of matching problems that seek to match together pairs or groups of agents subject to the preferences of some or all of the agents. We present a number of new algorithmic results for five specific problem domains. Each of these results is derived with the aid of some *structural* properties implicitly embedded in the problem.

We begin by describing an approximation algorithm for the problem of finding a *maximum* stable matching for an instance of the *stable marriage problem with ties and incomplete lists* (MAX-SMTI). Our polynomial time approximation algorithm provides a performance guarantee of $3/2$ for the general version of MAX-SMTI, improving upon the previous best approximation algorithm, which gave a performance guarantee of $5/3$.

Next, we study the *sex-equal stable marriage problem* (SESM). We show that SESM is $W[1]$-hard, even if the men's and women's preference lists are both of length at most three. This improves upon the previously known hardness results. We contrast this with an exact, low-order exponential time algorithm. This is the first non-trivial exponential time algorithm known for this problem, or indeed for any hard stable matching problem.

Turning our attention to the *hospitals / residents problem with couples* (HRC), we show that HRC is NP-complete, even if very severe restrictions are placed on the input. By contrast, we give a linear-time algorithm to find a stable matching with couples (or report that none exists) when stability is defined in terms of the classical Gale-Shapley concept. This result represents the most general polynomial time solvable restriction of HRC that we are aware of.

We then explore the *three dimensional stable matching problem* (3DSM), in which we seek to find stable matchings across three sets of agents, rather than two (as in the classical case). We show that under two natural definitions of stability, finding a stable matching for a 3DSM instance is NP-complete. These hardness results resolve some open questions in the literature.

Finally, we study the *popular matching problem* (POP-M) in the context of matching a set of applicants to a set of posts. We provide a characterization of the set of popular matchings for an arbitrary POP-M instance in terms of a new structure called the *switching graph*. We show that this structure can be exploited to yield efficient algorithms for a range of associated problems, extending and improving upon the previously best-known results for this problem.

Contents

List of Figures

Declaration

This thesis is submitted in accordance with the rules for the degree of Doctor of Philosophy at the University of Glasgow in the Faculty of Information and Mathematical Sciences. None of the material contained herein has been submitted for any other degree, and all the results are claimed as original.

Some of the work in this thesis is the product of collaboration with the members of my research group. Chapters 4 and 7 were undertaken jointly with Rob Irving. In particular, the in depth analysis of the profile-based algorithm described in Section 7.3.5 originated with Rob. The results of Chapter 5 and Chapter 6 represent a joint effort with David Manlove and Peter Biró, respectively. The proof of Theorem 4.6.1, in Chapter 4, is due to David Manlove.

Publications

The following publications contain some of the results presented in this thesis.

E. McDermid, A 3/2-approximation algorithm for general stable marriage. *In Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP 2009)*, pages 689–700, 2009. (This paper is based on Chapter 3).

R. Irving and E. McDermid, Sex-equal stable matchings: improved complexity results and exact algorithms. *In submission.* (This paper is based on Chapter 4).

E. McDermid and D. Manlove, Keeping partners together: Algorithmic results for the hospitals/residents problem with couples. *Journal of Combinatorial Optimization*, 46(3):359–367, 2009. (This paper is based on Chapter 5).

P. Biró and E. McDermid, Three-sided stable matchings with cyclic preferences. To appear in *Algorithmica*, 2010. (Preliminary version appeared in Proceedings of the 2nd International Workshop on Computational Social Choice (COMSOC 2008), pages 97–108, 2008) (This paper is based on Chapter 6).

E. McDermid and R. Irving, Popular matchings: structure and algorithms. To appear in *Journal of Combinatorial Optimization*, 2010. (Preliminary version appeared in Proceedings of the 15th

Annual International Conference on Computing and Combinatorics (COCOON 2009), pages 506–515, 2009.) (This paper is based on Chapter 7).

Acknowledgements

Oddly enough, the most difficult part of writing this thesis did not involve producing a proof of a difficult theorem ‘from the book’. Rather, the hardest part was determining how I can thank my colleagues, friends, and family in an appropriate way, without making this thesis twice as long as it would otherwise be.

I will try to be brief. I will not succeed.

I am truly honored to have had Rob Irving as my primary supervisor – I could not have asked for someone better. Rob read absolutely everything I wrote over these three years, providing me with invaluable feedback that greatly increased the quality of the thesis, and other papers that I have written. Collaborating with Rob felt like exploring research problems with a friend, rather than an actual supervisor. His enthusiasm for our research area is infectious, and I often benefited from his thorough knowledge of mathematics. The fact that he always took the time to patiently listen to my ideas – even the ones that were, well, stupid – gave me the courage to try to accomplish things I would not have otherwise attempted. So, thanks Rob, for playing the perfect combination of advisor, colleague, and friend.

Thanks also to David Manlove, my second supervisor, for his general enthusiasm and encouragement. I am also grateful for his impeccable proof-reading abilities, and uncanny skill to immediately understand mathematical transformations. Most importantly, without David, I would not understand the momentous significance of the phrase *Hibs vs Hearts*.

I greatly benefited from many helpful discussions and collaborations with my friend and officemate Peter Biró, who I also thank for often sharing his wife’s delicious cooking (thanks Marti!).

My studentship was funded by the Engineering and Physical Sciences Research Council grant EP/E011993/1, as a part of the Match-Up project. I hope they appreciate this return on their investment.

I thought for quite a while about how to acknowledge all of my friends without attempting to actually name everyone. I have narrowed it down to this: if you (i) know the significance of the phrase “Nice, come back, nice”, (ii) have been forced to fold during poker for not paying attention, (iii) have watched “Errand of Mercy”, or (iv) have camped at Bailey’s Bluff, then, whether you realize it or not, you played an important role in this thesis. Thanks, you guys.

Sean, thanks for making such an effort to keep in contact over the last few years, and also for making it your mission to distract me as much as possible. I thought of a thousand things to write here, but somehow this one word summarizes all of it: KHAAAAAAAAN!

Thank you Grandma and Grandpa McDermid, for your generosity and kindness to Megan and I over the last three years. Thank you also for consistently feeding me the most wonderful food imaginable when I visit you. You have made a significant impact on my life. Did I mention the food? Thanks also to Grandma Lipinski for your encouragement and kindness, and for the steady stream of cards and gifts sent to Megan and I. GO PACK GO!

I'm grateful to my sister Julie, for her generosity and encouragement over the years. Thanks for being a great sister. The first theorem of this thesis is the following: my sister is better than yours. Proof: she gave me a laptop. QED.

My parents are an inexhaustible source of encouragement and help, and are certainly the most generous people I have ever known. Thank you so much, you two, for your help, advice, and prayers. It's funny how as I get older I find myself seeking out your advice and opinions *more*, not less – who would have thought that people in their twenties didn't know everything? I have heard it said, that anyone who writes something, does so with someone particular in mind. In this case it is indeed true – this thesis is written for you, because I knew it would make you proud.

Thank you, Megan, for coming with me to a strange land where it rains almost everyday, and things like *fritter rolls* are considered lunch. It is so important to me that you know this: you and I achieved this *together*. We came here with a goal, and we accomplished it. There is no way I could have done it without you. At this point I could make a multitude of corny comments like 'you're my stable marriage partner' or 'you're my first choice on my preference list' (I know what you'd say to that one: "Who ELSE is on your preference list?!"). Let me instead simply say that I love you, and am so thankful to have you as my wife.

I thank God for constantly providing for me, and making sense out of my life. Anything good I have done or will do is because of Him. In Psalms it says, *If I rise on the wings of the dawn, if I settle on the far side of the sea, even there your hand will guide me, your right hand will hold me fast*. Trust me, it's true. I tried it.

Finally, I thank the University of Glasgow for supplying me with the books *How to Get a PhD*,

How to Write a Thesis, and *How to Survive your Viva*. For three years, rain or shine, these three books have done an absolutely amazing job of keeping our office window propped open to let the breeze in.

I wonder if they are any good?

Chapter 1

Introduction and summary

In a *matching problem with preferences*, we seek to match together pairs or groups of agents (such as men and women, or applicants and posts) subject to their preference for one another. Such matching problems incorporate either *full* or *partial* preference information in the following sense. In a matching problem with full preference information, *every* agent has some form of preference list that ranks some or all of the other agents. In the partial preference setting, however, only *some* of the agents have preference lists – the rest of the agents being, for example, inanimate objects. This thesis presents a number of new algorithmic results for five specific problem domains from the realm of both full and partial preference information. We emphasize a *structural* approach to each individual problem in the sense that each new algorithmic result presented relies crucially on some structural property implicitly embedded in the problem.

We begin (in Chapter 2) with a selective survey of the results arising in the context of matching problems with preferences. Naturally, special attention is paid to those results that are particularly relevant in subsequent chapters of this thesis. Next, in Chapters 3 – 7, we present our primary results, which are outlined briefly as follows.

The first problem we consider is a well-studied variant of the stable marriage problem in which we seek to find stable matchings that are as *large* as possible. To be precise, when the men and women of a stable marriage instance are permitted to have ties and incomplete preference lists, stable matchings can have different sizes (in contrast to the classical case). It is known that finding a maximum cardinality stable matching is NP-hard, even when very severe restrictions are placed on the sizes and positions of the ties, and the lengths of the preference lists [74]. Accordingly, there

has been much recent interest in finding polynomial time approximation algorithms with a constant performance guarantee for both the general version of this problem, and for several special cases.

The first contribution of this thesis is to describe an approximation algorithm for the general version of this problem with an improved performance guarantee. A key ingredient of our algorithm is a classic structural result on matchings in bipartite graphs called the *Gallai-Edmonds decomposition theorem* [70], which, roughly speaking, categorizes the vertices of a bipartite graph in terms of its maximum matchings. Our approximation algorithm employs this structural theorem, along with novel techniques, to obtain a performance guarantee of $3/2$. The previously best known approximation algorithm for this problem gave a guarantee of $5/3$ [64].

Moving away from the notion of ties in the preference lists, we next consider the *sex-equal stable marriage problem* (SESM). The goal of this stable marriage variant is to find a stable matching that, in a formal sense, is *fair* to both the men and the women. This problem is known to be strongly NP-hard [58]. We focus our aim specifically on stable marriage instances in which the lengths of the preference lists of the men and/or women are bounded in length by a constant. Our contribution is to strengthen the known hardness results by proving that SESM is $W[1]$-hard, even if the lengths of the men's and women's preference lists are both at most three. Additionally, we give an exact low-order exponential-time algorithm for SESM in which the men's preference lists are bounded in length by a constant l, and the lengths of the women's preference lists are unrestricted. The running time of our algorithm is bounded by $O^\star(1.0725^n)$, $O^\star(1.1503^n)$, $O^\star(1.2338^n)$, ... for $l = 3, 4, 5,$ .... On the other hand, we show that when $l = 2$, SESM is solvable in polynomial time.

These hardness results and algorithms rely heavily on the structural properties of the set of stable matchings. In particular, we make use of some classical – and new – bounds concerning the *rotations* of a stable marriage instance [36]. Our exact algorithm exploits these bounds, along with a recent extremal result concerning the structure of graphs with bounded average degree [25], to achieve the stated running time.

We next study the hospitals / residents problem (HR), a many-to-one generalization of the stable marriage problem. Specifically, we consider the hospitals / residents problem with couples (HRC), in which pairs of (for example, married) residents are allowed to form *couples*, who wish to be matched to the same hospital, or to hospitals geographically nearby [76, 21, 88, 65, 66]. We consider a natural restriction of HRC in which the members of a couple have individual preference lists over hospitals, and the couples form joint preference lists that are, in a formal sense, *consistent* with these

individual preference lists. We give an appropriate stability definition and show that the problem of deciding whether a stable matching exists is NP-complete, even if each resident's preference list has length at most three and each hospital has capacity at most two. In contrast to this result, we give a linear-time algorithm to find a stable matching (or report that none exists) when stability is defined in terms of the classical Gale-Shapley concept. This algorithm makes no assumptions about the preference lists or capacities of the hospitals. Finally, for an alternative formulation of our restriction of HRC, which we call the *hospitals / residents problem with sizes* (HRS), we give a linear-time algorithm that always finds a stable matching for the case that hospital preference lists are of length at most two, and where hospital capacities can be arbitrary.

Our linear-time algorithm utilizes the structure of the set of solutions induced by Gale-Shapley stability. In particular, the set of stable matchings that "keep couples together" adhere to a particular *dominance* relation, which defines a partially ordered set. Our algorithm efficiently navigates a path through the space of possible solutions by exploiting the attributes of this relation. This result represents the most general polynomial time solvable restriction of HRC that we are aware of.

The next chapter explores the generalisation of the stable marriage problem to *three* dimensions, so that we have a set of, say, men, women, and *dogs*. Donald Knuth initiated the study of this problem in the mid-1970s by asking if the results surrounding the stable marriage problem extended to this setting [69]. Over the years, a number of researchers have explored various three dimensional stable matching problems in an effort to answer Knuth's open question. One recurring open problem in the literature is the *cyclic three dimensional stable matching problem* (3DSM), in which men care about only the women, women care about only the dogs, and dogs care about only the men. Several authors have asked, either explicitly or implicitly, if stable matchings always exist in this setting, and, in any case, if there exists a polynomial time algorithm to either return a stable matching, or report that none exists [26, 83, 11]. Our contribution is to examine 3DSM under two natural definitions of stability, given previously in the literature, and show that under both definitions of stability, 3DSM is NP-complete.

The cardinal ingredient in our constructions is a specially constructed instance of 3DSM we call a 9-*Sun*. When we view this instance in terms of its *underlying graph*, it is easy to see that the special structural nature of this instance makes the creation of a stable matching impossible. We use the 9-Sun strategically as a subgraph in the derived instances of our reductions to attain our primary hardness results.

Finally, we consider a problem with partial preference information, in which we seek to match a set of applicants to a set of posts. One notion of optimality in this setting is that of a *popular matching* – a kind of matching that is derived *democratically* from the applicant's preference lists. Being somewhat formal, a matching M is popular if there is no majority of the applicants who would agree to abandon M for a different matching M'. The goal of the *popular matching problem* (POP-M), therefore, is to find a popular matching if one exists. There is a known linear time algorithm to determine whether a popular matching exists for a given POP-M instance [5], and if so, this algorithm finds a largest such matching. A number of variants and extensions of POP-M have recently been studied.

Our contribution is to provide a characterization of the set of popular matchings for an arbitrary POP-M instance in terms of a new structure called the *switching graph*, a directed graph computable in linear time from the preference lists. We show that this structure can be exploited to yield efficient algorithms for a range of associated problems, including the counting and enumeration of the set of popular matchings, generation of a popular matching uniformly at random, finding all applicant-post pairs that can occur in a popular matching, and computing popular matchings that satisfy various additional optimality criteria. Our algorithms for computing such optimal popular matchings improve upon the best previous results for this problem [60].

Thus, each result in this thesis – whether positive or negative – relies heavily on some key underlying structural observations. In some cases, we use the structure to force the problem to "bend to our will". The new results surrounding the switching graph described in Chapter 7, or the half-century old Gallai-Edmonds decomposition theorem given in Chapter 3, for example, introduce sufficient *order* and *organization* into the problem to give us the desired results. On the other hand, sometimes the structural observations reveal the kind of chaos that can occur within a problem instance. This gives invaluable insight into what makes certain problems computationally difficult. For example, as we show in our hardness proofs in Chapter 6, the strategic placement of a few 9-Suns into a three dimensional stable matching instance is enough to bring about complete disorder.

Chapter 2

Background

2.1 Introduction

Computer scientists almost invariably model combinatorial optimization problems by assigning numerical values (costs, weights, etc.) to the various objects of the problem instance. The usual goal is then to either maximize or minimize some objective function derived from these numerical values. A researcher who studies graph algorithms, for example, is bound to have numerous references within an arm's reach that model various problems with graphs that have costs or weights on subsets of the edges and/or subsets of the vertices. Even within the restricted realm of matchings in graphs, entire books have been written to catalogue a number of classic polynomial-time algorithms for finding maximum cardinality matchings, maximum weight matchings, minimum cost maximum matchings, and so on.

With this in mind, suppose we are given an instance of the following *stable marriage problem*. The instance consists of a set of n men and n women, each of whom provides a preference list ranking, in strict order, the n people of the opposite set. This problem clearly gives rise to a bipartite graph, whose two disjoint sets of vertices are the men and women, respectively. Armed with decades of algorithmic machinery, we could attempt to assign costs and weights to the edges and vertices of this graph in a way that somehow captures the preference lists of the men and women. Thus we could perhaps try to find various optimal matchings (mincost, maximum cardinality, etc.) in this weighted bipartite graph.

Perhaps the most subtle contribution Gale and Shapley [30] made when they studied this problem was to realize that all of the common or natural ways of doing so completely *fail* in one particular regard. It will not, in general, be possible to guarantee that the matching will not *unravel* in the following sense: there could be a man and a woman who both prefer each other to their respective marriage partners , and therefore might leave those partners and instead run off together. This exemplifies the fact that, since the vertices of our bipartite graph correspond to *people*, we need an optimality criterion that is convincingly good on an *individual* level.

Gale and Shapley argued that an optimal matching should be one that avoids this unraveling situation – hence no one has motivation to be divorced or seek an arrangement outside of the matching mechanism. Fittingly, they called such a matching a *stable matching*. In a single theorem, they proved that at least one stable matching always exists, and that such a matching can be found in polynomial time [30].

Now, decades later, this single publication has effectively spawned a whole host of research areas with results arising from the fields of mathematics, computer science, and economics. Some of the results are very rich and beautiful theoretical ideas, exploring, for example, structural relationships between various stable matchings. Other results – often algorithmic – are instead motivated by problems arising from real-world applications. A few examples include the central assignment of graduating medical students to their first job at a hospital [105, 106, 107], matching students to schools in urban areas [12], and finding optimal kidney exchanges amongst incompatible (donor,patient) pairs [108].

Whether the focus is on theoretical or practical results, all these various *matching problems with preferences* have a common theme: given a set of agents, each of whom has some form of preference over the set of possible outcomes, find a matching of the agents that is in some sense optimal with respect to these preferences. Specifically, matching problems arising in this context involve a set of agents (for example men and women), where some or all of the agents may have a preference list over a subset of the other agents. Taking a broad view of roughly five decades of literature, one could categorize matching problems with preferences as follows:

1. *Full preference information* – where every agent has some form of preference list (possibly involving ties) ranking some or all of the other agents of the instance. This category can be further refined into the following subcategories:

(a) *Bipartite matching problems* – in which the agents can be partitioned into two disjoint sets, and the task is to match the agents in one set to the agents in the other.

(b) *Nonbipartite matching problems* – where the agents form one homogeneous set with each agent having preferences over a subset of the others.

(c) *d-dimensional matching problems* – where the agents can be partitioned into $d \geq 3$ disjoint sets and the task is to match the agents into d-tuples.

2. *Partial preference information* – in which the agents of the instance can be partitioned into two sets A and P, with the agents in A having preferences (possibly involving ties) over the agents in P, but the agents in P do not have any form of preference lists.

It is impossible to completely review all of the results arising in the context of matching problems with preferences. Instead, this introductory chapter offers a selective survey of various structural and algorithmic results for matching problems with preferences according to the above classification. Naturally, special attention is paid to those results that are particularly relevant in subsequent chapters of this thesis.

2.2 Full preference information

2.2.1 Stable Marriage problem

An instance of the *stable marriage problem* (SM) consists of a set of n men and a set of n women. Associated with each person is a *preference list*, defined to be a total ordering (hence no ties are allowed) ranking all of the members of the opposite set. A preference list is interpreted in such a way that a person a prefers b to c if and only if b precedes c on a's preference list. A stable marriage instance is typically said to have *size* or *order* n, as this is the number of people (henceforth, *agents*) in each set. Observe that this is actually a slight misnomer, as the sum of the lengths of all of the preference lists of the instance, and hence the actual input size for the instance, is $\Theta(n^2)$.

A *matching* (or a *marriage*), is defined to be a one-one correspondence between the men and the women. If a man m and a woman w are matched together in a matching M, then we say that they are *partners* in M, and write $m = M(w)$ and $w = M(m)$. A man m and a woman w are said to be a *blocking pair* for M if m prefers w to $M(m)$ and w prefers m to $M(w)$ – in English, m and w are

$m_1:$	$\underline{w_1}$	w_3	w_4	w_2	w_5
$m_2:$	w_1	$\underline{w_2}$	w_4	w_3	w_5
$m_3:$	w_3	w_4	$\underline{w_5}$	w_1	w_2
$m_4:$	$\underline{w_4}$	w_1	w_3	w_2	w_5
$m_5:$	$\underline{w_3}$	w_1	w_5	w_4	w_2

$w_1:$	$\underline{m_1}$	m_3	m_5	m_2	m_4
$w_2:$	m_5	$\underline{m_2}$	m_1	m_3	m_4
$w_3:$	$\underline{m_5}$	m_1	m_2	m_4	m_3
$w_4:$	m_5	m_1	$\underline{m_4}$	m_2	m_3
$w_5:$	m_1	m_2	$\underline{m_3}$	m_5	m_4

Figure 2.1: An SM instance with a stable matching denoted by underlining.

a blocking pair if they would both rather be matched to each other than to their respective partners. For brevity, we often say m and w are *blocking for M* or that m and w *block M*, and so forth, in a way that should always be clear from the context.

If there is at least one blocking pair relative to a matching M, then M is said to be an *unstable matching*, or *unstable*. Otherwise, if there are no blocking pairs, M is said to be a *stable matching*, or simply *stable*. The goal of the stable marriage problem is to take an arbitrary stable marriage instance and output a stable matching. Figure 2.1 shows a stable marriage instance with a stable matching denoted by underlining.

The primary contribution of Gale and Shapley [30] was to show that a stable matching exists for every stable marriage instance. They proved this result constructively by describing a polynomial-time *deferred acceptance algorithm*, which is now instead widely known as the Gale-Shapley algorithm . It has been observed that this algorithm can be implemented to run in $O(n^2)$ time [69], and is therefore a linear-time algorithm relative to the size of the input. We remark that the stable matching found by the Gale-Shapley algorithm is not necessarily the only stable matching, as there can be many stable matchings for a given instance. We comment further on this in Section 2.2.3.

Roughly speaking, the Gale-Shapley algorithm involves a sequence of iterative proposals from the men to the women, in which the men of the instance essentially compete with one another for the women. An interesting property of the resulting stable matching is that it is *man-optimal*, because each man actually achieves the best partner he can possibly have in any stable matching. Hence if we switch the roles of the men and women, so that the algorithm is *woman-oriented*, the resulting stable matching will be *woman-optimal* . We will use the convention M_0 and M_z to denote the man-optimal and woman-optimal stable matchings, respectively, of a given stable marriage instance.

McVitie and Wilson [78] showed that the optimality of the men in M_0 or the women in M_z always comes at the price of the other set's extreme suboptimality. In particular, they proved that M_0 is *woman-pessimal* – meaning every woman is actually matched to the worst man she can ever be

matched to in any stable matching. Reversing the roles of the men and the women, we can deduce that M_z is therefore *man-pessimal*, meaning every man is matched to the worst woman he can ever be matched to in any stable matching.

In fact, the observations on the inter-relationships between man- and woman-optimal (man- and woman-pessimal) matchings is just the tip of the iceberg. A deeper and broader understanding of the rich structure of the set of all stable matchings would eventually emerge from these observations, but before diving into these details, we move on to some of the natural extensions and generalizations of the stable marriage problem.

2.2.2 Extensions of stable marriage

There are at least three obvious ways to relax the specification of the stable marriage problem.

1. We could allow the number of men and women of the instance to be *unequal*, so that some agents will necessarily be unmatched .
2. We could allow the men and the women to deem some members of the opposite set to be *unacceptable*, giving rise to *incomplete preference lists*, so that the men need only rank a subset of the women on their preference list, and, similarly, the women need only rank a subset of the men.
3. We could allow the agents to express some form of *indifference* in their preference lists, so that the preference lists are no longer restricted to being totally ordered.

The numerous results surrounding stable marriage with ties are worthy of their own section and are discussed separately in Section 2.2.5. In this section, we will just focus on the first two of the above relaxations of the stable marriage problem.

The *stable marriage problem with incomplete lists* (SMI) captures relaxations (1) and (2) in the following way. An instance of this problem consists of a set of n_1 men and a set of n_2 women (possibly $n_1 \neq n_2$). We let $n = n_1 + n_2$ denote the sum of the numbers of men and women. The preference list associated with each agent is a total ordering of a subset of the members of the opposite set.

$m_1:$	$\underline{w_1}$	w_3	w_4		$w_1:$	$\underline{m_1}$	m_2	m_4	
$m_2:$	w_1	$\underline{w_2}$			$w_2:$	m_5	$\underline{m_2}$	m_4	
$m_3:$	w_3	w_4			$w_3:$	$\underline{m_5}$	m_4	m_3	m_1
$m_4:$	$\underline{w_4}$	w_1	w_3	w_2	$w_4:$	m_5	m_1	$\underline{m_4}$	m_3
$m_5:$	$\underline{w_3}$	w_2							

Figure 2.2: An SMI instance with a stable matching denoted by underlining.

Hence SMI allows men (women) to implicitly declare some of the women (men) to be *unacceptable* in that they would rather be unmatched than matched to a person who is not present on their preference list. If an agent a appears on agent b's preference list, then we say that a is *acceptable* to b. If the converse is also true, then a and b are said to be *mutually acceptable*. A *matching* M for an SMI instance is defined to be a set of disjoint (man,woman) pairs (m, w) such that m and w are mutually acceptable. We let m denote the sum of the lengths of the preference lists, hence the input size of an SMI instance is $O(m)$.

It is easy to see that it may not be possible to match every agent in an SMI instance. As a consequence, we require an alternative notion of a blocking pair. A pair (m, w) is a *blocking pair* for a matching M if:

(i) m and w are mutually acceptable, and

(ii) m is either unmatched in M, or prefers w to his partner in M, and

(iii) w is either unmatched in M, or prefers m to her partner in M.

Figure 2.2 gives an example of an SMI instance with a stable matching.

The Gale-Shapley algorithm can be easily extended to the SMI setting [36], proving that a stable matching always exists for an SMI instance. The results concerning the man-optimal and woman-optimal (man-pessimal and woman-pessimal) stable matchings also generalize in the obvious way.

Gale and Sotomayor [31] observed the remarkable result that, while there can still be many stable matching for an arbitrary SMI instance, all stable matchings match exactly the same subset of the agents. Thus if an agent is unmatched (matched) in one stable matching, they are unmatched (matched) in all of them. We can therefore think of the agents of an SMI instance as being partitioned into two sets – the matched set and the unmatched set. If we were to attempt to explore

the set of stable matchings for an SMI instance, we need only turn our attention to the set of stable matchings for the matched set of agents. Because of this fact it is sometimes desirable to first discard the unmatched agents from an SMI instance, deleting them from the preference lists of the agents in the matched set. The set of stable matchings for the resulting instance is exactly the same as the set of stable matchings in the original instance [36].

2.2.3 The structure of stable matchings

So far we have discussed the existence of the man-optimal and woman-optimal stable matchings (M_0 and M_z) of an SMI instance, which are quite literally the most extreme stable matchings. When dealing with an instance in which $M_0 = M_z$, M_0 must be the unique stable matching – this is the only way that every man's best partner could also be his worst partner. If instead $M_0 \neq M_z$, there may indeed be many additional stable matchings. In particular, Knuth [69] showed that the maximum number of stable matchings for an SM instance grows exponentially with n. Irving and Leather [46] reestablished this fact in a different way, and showed that for each $n > 0$, where n is a power of two, there exists an instance of SM of size n with at least 2^{n-1} stable matchings.

In what follows we summarize the rich results concerning the structure of the set of all stable matchings for an SMI instance. We also discuss the key compact representations of the set of stable matchings, along with the algorithmic consequences of this structure.

Henceforth, we let $\mathcal{M}$ denote the set of all stable matchings of an arbitrary SMI instance. We define the following partial order on $\mathcal{M}$. Let M and M' be two (not necessarily distinct) stable matchings in $\mathcal{M}$. We say that M *dominates* M', denoted $M \succeq M'$, if, for each matched man m, $M(m) = M'(m)$ or m prefers $M(m)$ to $M'(m)$. Intuitively, M dominates M' if each man is at least as happy with his partner in M as he is with his partner in M'.

Knuth [69] attributes the following striking result concerning stable matchings to John Conway. Let M and M' be two distinct stable matchings. If each man is matched to the more (less) preferred of his two partners in M and M', then the result is a stable matching. One of the reasons why this result is so surprising is that there is no a priori reason why this operation should even constitute a matching, much less one that is stable. In light of Conway's result, it can be seen that the pair $(\mathcal{M}, \succeq)$ actually forms a distributive lattice, with the meet (join) of two stable matchings being the operation of assigning each man to the better (worse) of his two partners in any two stable

matchings. The maximum and minimum elements of this lattice are the man- and woman-optimal stable matchings, respectively.

The lattice representation of the set of stable matchings is indeed quite interesting, but does not immediately lend itself to any particularly efficient algorithms for, say, generating all stable matchings or finding a stable matching with some special additional property. Since $\mathcal{M}$ can have exponential size, any algorithm that explicitly generates this lattice is doomed to require exponential-time (and possibly even exponential space). Irving and Leather [46] discovered a polynomial-space representation of $\mathcal{M}$, called the *rotation poset*, which essentially captures all the different ways one can 'navigate' the lattice of stable matchings by moving from one stable matching to another (we define this structure formally below).

The key to making a transition from one stable matching to another is a *rotation* [46]. Let M be a stable matching. For each man m, let $s_M(m)$ denote the first woman w on m's preference list succeeding $M(m)$ such that w prefers m to $M(w)$, if such a woman exists. Then, a rotation ρ is defined to be an ordered sequence of pairs (m_0, w_0),...,(m_{r-1}, w_{r-1}), such that for each i $(0 \leq i \leq r-1)$ $(m_i, w_i) \in M$, and $w_{i+1} = s_M(m_i)$ (all subscripts are taken modulo r). Such a rotation is said to be *exposed* in M. To *eliminate* a rotation is to match each man m_i to w_{i+1}, where $i+1$ is taken modulo r, and leave all other agents matched as in M. The resulting matching is denoted by M/ρ, and is in fact always stable [46]. Note that by eliminating a rotation ρ the men in ρ become worse off and the women in ρ become better off, with everyone else not in ρ remaining the same. Every stable matching except M_z has at least one exposed rotation [46].

Consider the set of rotations $\{\rho_0, \rho_1, \ldots \rho_k\}$ exposed in M_0 (this set must be non-empty when $M_0 \neq M_z$). If we choose to eliminate a rotation, say ρ_0, then $\rho_1, \ldots, \rho_k$ remain exposed in M_0/ρ_0. Also, the elimination of ρ_0 may have exposed different rotations that were not exposed in M_0. Thus we may continue to eliminate rotations, arriving at different stable matchings. With each new stable matching, some new rotations may become exposed. Let R denote the union of the sets of exposed rotations taken over all stable matchings M. Irving and Leather showed that R is uniquely determined by the instance, because any two rotations are either identical or disjoint. For two rotations ρ_1 and ρ_2, we say that ρ_1 precedes ρ_2, denoted $\rho_1 \prec \rho_2$, if ρ_2 is never exposed unless ρ_1 has been eliminated. The *rotation poset*, which is uniquely determined by $\mathcal{M}$, is the pair $\Pi = (R, \prec)$. It is important to note that the number of elements of Π is $O(m)$.

A *closed subset* R' of the rotation poset $\Pi = (R, \prec)$ is a subset of R such that if $\rho \in R'$ and ρ'

$\prec \rho$ then $\rho' \in R'$. The key contribution of Irving and Leather was not only to show that the ideas surrounding rotations and the rotation poset allowed one to find different stable matchings, but was to further show that eliminating rotations is, in a sense, the *only* way to arrive at different stable matchings. Specifically, they showed that there exists a one-one correspondence between $\mathcal{M}$ and the closed subsets of Π: let R' be an arbitrary closed subset. If we compute M_0, we can eliminate the rotations in R' in any order that adheres to $\prec$, and arrive at a stable matching. Furthermore, every stable matching can be obtained by starting at M_0 and eliminating a distinct closed subset of Π. In this way, Π encodes $\mathcal{M}$. A Hasse diagram representation (the transitive closure) of Π can be computed in $O(m)$ time and space [47, 36].

Example Before moving on to the next section, we provide an illustrative example of the concept of the lattice of stable matchings, its rotations, and its rotation poset. Consider the SMI instance given below:

$$
\begin{array}{llllll}
m_1: & w_1 & w_3 & w_2 & & \\
m_2: & w_2 & w_1 & w_3 & w_5 & w_4 \\
m_3: & w_3 & w_1 & w_2 & w_5 & \\
m_4: & w_4 & w_2 & w_3 & & \\
m_5: & w_5 & w_4 & w_1 & &
\end{array}
\qquad
\begin{array}{llllll}
w_1: & m_5 & m_3 & m_2 & m_1 & \\
w_2: & m_1 & m_2 & m_3 & m_4 & \\
w_3: & m_4 & m_2 & m_3 & m_1 & \\
w_4: & m_2 & m_4 & m_5 & & \\
w_5: & m_3 & m_5 & m_2 & &
\end{array}
$$

This instance has a total of six stable matchings, where M_1 is the man-optimal stable matching, and M_6 is the woman-optimal stable matching:

$$
\begin{aligned}
M_1 &= \{(m_1, w_1), (m_2, w_2), (m_3, w_3), (m_4, w_4), (m_5, w_5)\} \\
M_2 &= \{(m_1, w_2), (m_2, w_1), (m_3, w_3), (m_4, w_4), (m_5, w_5)\} \\
M_3 &= \{(m_1, w_2), (m_2, w_3), (m_3, w_1), (m_4, w_4), (m_5, w_5)\} \\
M_4 &= \{(m_1, w_2), (m_2, w_4), (m_3, w_1), (m_4, w_3), (m_5, w_5)\} \\
M_5 &= \{(m_1, w_2), (m_2, w_3), (m_3, w_5), (m_4, w_4), (m_5, w_1)\} \\
M_6 &= \{(m_1, w_2), (m_2, w_4), (m_3, w_5), (m_4, w_3), (m_5, w_1)\}
\end{aligned}
$$

The instance has a total of four rotations:

$$\rho_1 = ((m_1, w_1), (m_2, w_2))$$

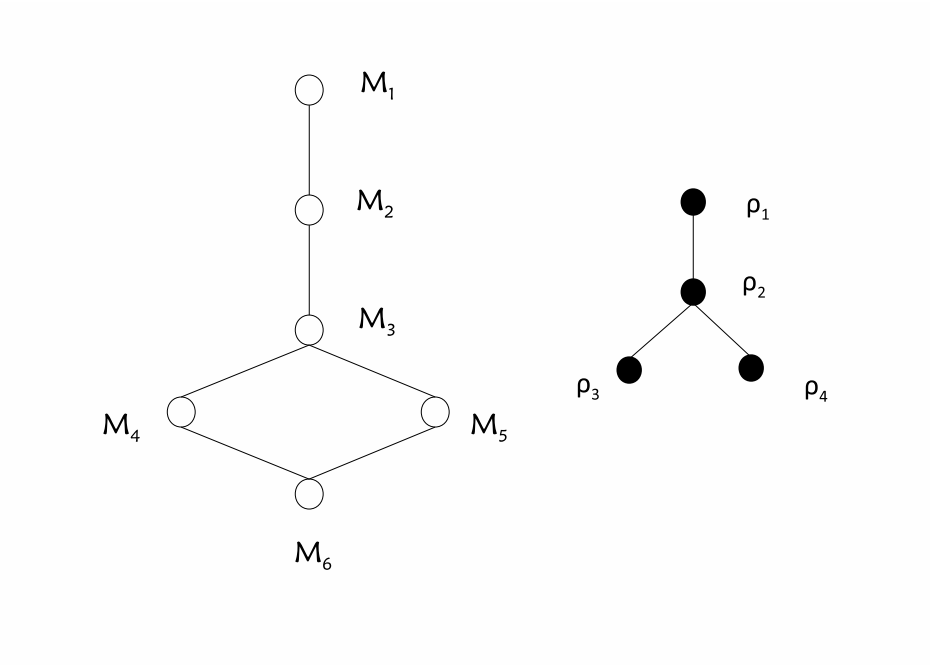

Figure 2.3: The Lattice and poset of an SMI instance

$$\rho_2 = ((m_2, w_1), (m_3, w_3))$$
$$\rho_3 = ((m_2, w_3), (m_4, w_4))$$
$$\rho_4 = ((m_5, w_5), (m_3, w_1))$$

The lattice structure and the Hasse diagram of the poset of this instance is given in Figure 2.3. The correspondence between the elements of the lattice and the closed subsets of the rotation poset is as follows: M_1 and $\emptyset$, M_2 and $\{\rho_1\}$, M_3 and $\{\rho_1, \rho_2\}$, M_4 and $\{\rho_1, \rho_2, \rho_3\}$, M_5 and $\{\rho_1, \rho_2, \rho_4\}$, M_6 and $\{\rho_1, \rho_2, \rho_3, \rho_4\}$.

2.2.4 Exploiting the structure

The algorithmic consequences of the rotation poset Π of an SMI instance are numerous. Of immediate interest is the fact that the rotation poset allows for the efficient generation of all stable matchings. Gusfield [34] showed that $\mathcal{M}$ can be enumerated in $O(m + n|\mathcal{M}|)$ time – hence there is only a linear-time delay between the output of each stable matching. He further showed that the set of all *stable pairs* – the set of (man,woman) pairs that appear in some stable matching – can be computed in $O(m)$ time. Later, Gent et al [33] gave a different approach to enumerating all stable

matchings by using constraint programming. Their method uses arc consistent domains in order to achieve failure-free enumeration of all stable matchings.

An interesting question that arises is whether there are stable matchings that are somehow fair to both the men and women of the instance, as opposed to the extreme unfairness of the stable matchings M_0 and M_z. We next review several such problems, some of which are polynomial-time solvable, thanks to the structural results of the rotation poset. Henceforth, for agents a and b, let $p_a(b)$ denote the *position* of agent b on agent a's preference list. If a finds b unacceptable, $p_a(b)$ is undefined.

Minimum regret stable matchings

Given a stable matching M we define the *regret* of agent a to be $p_a(M(a))$, i.e, the position of a's partner in M. The regret of an unmatched agent is undefined. The regret of a matching M is the maximum regret taken over all agents in M. A *minimum regret stable matching* is a stable matching with minimum possible regret.

For the stable marriage setting, Knuth [69] showed that the minimum regret stable matching problem can be solved in $O(m^2)$ time, attributing the result to Selkow. Gusfield [34] improved this to an optimal $O(m)$-time solution.

Fair stable matchings

Suppose we wish to somehow treat the men and the women of an SMI instance equally. For a stable matching M, define the *egalitarian value* $e(M)$ to be

$$e(M) = \sum_{(m,w)\in M} (p_m(w) + p_w(m)).$$

An *egalitarian stable matching* is a stable matching M that minimizes $e(M')$ over all $M' \in \mathcal{M}$. The egalitarian egalitarian stable matching problem (ESM) is to find an egalitarian stable matching. Intuitively, ESM captures the notion of finding a stable matching with the best "social welfare".

$$
\begin{array}{lllll}
m_1: & \underline{w_1} & w_2^* & \ldots & \\
m_2: & \underline{w_2} & w_3^* & \ldots & \\
\vdots & & & & \\
m_{n-1}: & \underline{w_{n-1}} & w_n^* & \ldots & \\
m_n: & \underline{w_n} & w_1^* & \ldots &
\end{array}
\qquad\qquad
\begin{array}{lllll}
w_1: & m_n^* & \ldots & \underline{m_1} & \\
w_2: & m_1^* & \ldots & \underline{m_2} & \\
\vdots & & & & \\
w_{n-1}: & m_{n-2}^* & \ldots & \underline{m_{n-1}} & \\
w_n: & m_{n-1}^* & \ldots & \underline{m_n} &
\end{array}
$$

Figure 2.4: An SM instance with two stable matchings with greatly varying values of $e(\cdot)$.

Example The example given in Figure 2.4 shows how $e(\cdot)$ can greatly vary for different stable matchings of a particular SMI instance. The example consists of an SM instance with two stable matchings, one denoted by underlining, and the other by star. The ellipses in the preference lists denote any arbitrary ordering of the remaining agents not explicitly mentioned. The egalitarian value $e(\cdot)$ for the stable matching denoted by underlining is $n^2 + n$, whereas $e(\cdot)$ for the matching denoted by star is $3n$.

Another notion of a fair stable matching arises by attempting to find a stable matching with the property that the men's and women's overall happiness is as close as possible. To this end, the *sex-equality measure* $\delta(M)$ of a stable matching is defined to be

$$\delta(M) = \sum_{(m,w)\in M} p_m(w) - \sum_{(m,w)\in M} p_w(m).$$

A *sex-equal stable matching* is a stable matching M that minimizes $|\delta(M')|$ over all $M' \in \mathcal{M}$. The *sex-equal stable matching problem* (SESM) is to find a sex-equal stable matching.

Example The example given in Figure 2.5 shows how $|\delta(\cdot)|$ can also greatly vary for different stable matchings of a particular SMI instance. The example consists of an SM instance with two stable matchings, one denoted by underlining, and the other by star. The ellipses in the preference lists denote any arbitrary ordering of the remaining agents not explicitly mentioned. The absolute value of the sex-equality measure $\delta(\cdot)$ for these two stable matchings is $n^2 - n$ and zero, respectively.

The elimination of a rotation can, in general, result in a stable matching with a different value of $e(\cdot)$ and/or $\delta(\cdot)$. For a rotation $\rho = (m_0, w_0)$, (m_1, w_1),...,(m_{r-1}, w_{r-1}), we define $v(\rho)$ and $w(\rho)$ [47, 56] to capture the change in egalitarian value and sex-equality measure, respectively, by eliminating ρ:

$$
\begin{array}{rllll}
m_1: & \underline{w_1} & w_2^* & \dots & \\
m_2: & \underline{w_2} & w_3^* & \dots & \\
\vdots & & & & \\
m_{n-1}: & \underline{w_{n-1}} & w_n^* & \dots & \\
m_n: & \underline{w_n} & w_1^* & \dots &
\end{array}
\qquad
\begin{array}{llll}
w_1: m_2 & m_n^* & \dots & \underline{m_1} \\
w_2: m_3 & m_1^* & \dots & \underline{m_2} \\
\vdots & & & \\
w_{n-1}: m_n & m_{n-2}^* & \dots & \underline{m_{n-1}} \\
w_n: m_1 & m_{n-1}^* & \dots & \underline{m_n}
\end{array}
$$

Figure 2.5: An SM instance with two stable matchings with greatly varying values of $|\delta(\cdot)|$.

$$v(\rho) = \sum_{i=0}^{r-1}(p_{m_i}(w_{i+1}) - p_{m_i}(w_i)) + \sum_{i=0}^{r-1}(p_{w_i}(m_{i-1}) - p_{w_i}(m_i)),$$

$$w(\rho) = \sum_{i=0}^{r-1}(p_{m_i}(w_{i+1}) - p_{m_i}(w_i)) - \sum_{i=0}^{r-1}(p_{w_i}(m_{i-1}) - p_{w_i}(m_i)).$$

Irving, Leather, and Gusfield [47] showed that ESM can be solved in $O(m^2)$ time by assigning the weight given by $v(\cdot)$ to each rotation in Π, and then finding a maximum weight closed subset of Π. Later, Feder [28] gave an alternative approach that improves this running time to $O(m^{1.5}\log n)$.

At first one would probably suspect that a similar approach would work for the sex-equal stable marriage problem. Indeed, on the surface, these two problems look almost identical: $e(M)$ is a sum of the ranks of the mens' and womens' partners, whereas $\delta(M)$ is the absolute difference. It is perhaps surprising then, that the sex-equal stable matching problem is strongly NP-hard [58]. On the positive side, Iwama et al [56] give a polynomial-time approximation algorithm for the so-called *near-sex-equal stable marriage problem*. Their algorithms involve assigning the weight $w(\rho)$ to each rotation in Π and then attempting to find an appropriate subset of rotations to eliminate. They further study the problem of finding a minimum regret stable matching amongst the set of all near-sex-equal stable matchings. This latter problem is NP-hard, but there is an approximation algorithm with a performance guarantee better than two [56]. We shall study SESM more extensively in Chapter 4.

As a final word on fair stable matchings, we mention the *median stable matching* problem. For each (matched) man m in an SMI instance, sort the multiset of women m is matched to in $\mathcal{M}$ from m's most to least preferred. For example, if man m is matched to woman w in exactly ten stable matchings in $\mathcal{M}$, then w appears ten consecutive times in this sorted list. Let $w_i(m)$ denote the i^{th} woman in a man m's sorted list, and M_i denote the assignment obtained by matching each man m'

to $w_i(m')$. Teo and Sethuraman [99] proved the surprising result that M_i is not only a matching, but is also stable. The i^{th}-median stable matching is defined to be the stable matching obtained by matching every man to $w_i(m)$. Note that, in general, not every M_i so obtained is distinct, and not every stable matching M is equal to some M_j for some j.

The definition of a median stable matching does not lend itself to any natural polynomial-time algorithm – it appears as though we must explicitly enumerate $\mathcal{M}$ to construct a median stable matching. Cheng [18] gave a new characterization of the so-called *generalized* median stable matchings, showing that there is an intimate relationship between median stable matchings and the median elements of the lattice of $\mathcal{M}$. She went on to show that finding a median stable matching is NP-hard, but is approximable in a formal sense, and even polynomial-time solvable for some special cases. Very recently, Kijima and Nemoto [63] improved upon some of Cheng's results.

2.2.5 Indifference

A natural generalization of SM and SMI is to allow the agents involved to express some form of *indifference* in their preference lists. The most natural way for agents to express indifference is in the form of *ties* in the preference list; a tie t on an agent a's preference list is defined to be a set of agents all of whom have the same position on a's list. The notion of a tie is important in the practical applications of SM and SMI – consider, for example a hospital that must attempt to produce a genuinely strict ranking of hundreds of medical students [105, 106, 107]. We use SMT (SMTI) to stand for the variant of SM (SMI) in which preference lists can contain ties.

Of course, with the inclusion of ties, the definition of a blocking pair must be reconsidered. It stands to reason that a (man,woman) pair should still form a blocking pair if they both improve by becoming matched to each other, but what if, for example, m is indifferent between his current partner and w?

There are three particularly natural formulations of blocking pair, each with a corresponding notion of stability. These three kinds of stability are defined as follows:

- *weak-stability:* a (man,woman) pair can block only by both becoming better off
- *strong-stability:* a (man,woman) pair can block if at least one of them becomes better off, and the other no worse off

- *super-stability:* a (man,woman) pair can block if neither of them becomes any worse off.

Notice that each form of stability above is increasingly more restrictive than the previous, so super-stability implies strong-stability implies weak-stability. Irving [43] observed that while weakly-stable matchings always exist for an SMT instance, strong- and super-stable matchings need not. He further gave a polynomial-time algorithm for each of the three forms of stability that either returns a stable matching or reports that none exists (in the case of weak-stability, the algorithm always returns a weakly-stable matching). Manlove extended Irving's results to the SMTI setting [72].

Stability, size, and structure

There is an interesting interplay between the various forms of stability and the cardinality of stable matchings. If a super-stable matching exists for an SMTI instance, then all stable matchings for the instance have equal cardinality, regardless of the definition of stability. Otherwise, if a strongly-stable matching exists, then all strongly-stable matchings have the same size. In general, weakly stable matchings can have different cardinalities, but every strongly-stable matching is at least two-thirds the size of an arbitrary weakly-stable matching [93].

Spieker [95] showed that the set of super-stable matchings for an SMTI instance forms a distributive lattice. Later, Manlove [73] gave an alternative, and perhaps more accessible proof showing that both strong- and super-stable matchings have a distributive lattice structure. The elements of the lattice structure described by Manlove are *sets* of "equivalent" stable matchings, rather than individual stable matchings. The maximum and minimum elements of the lattice correspond to the sets of man- and woman-optimal stable matchings. Scott [93] extended the notion of a rotation to super-stability, and described polynomial-time algorithms for finding egalitarian and minimum-regret stable matchings, along with algorithms for generating all super-stable matchings and finding all super-stable pairs. Extending such results to the strong-stability case remains an open question, but it seems likely that this can be done in light of the structural results of Manlove [73].

$$
\begin{array}{llllll}
m_1: & (\underline{w_2} & w_1^*) & & & \\
m_2: & w_3 & \underline{w_1} & & & \\
m_3: & w_1 & (w_3^* & \underline{w_4}) & w_2 & \\
m_4: & \underline{w_3} & & & &
\end{array}
\qquad\qquad
\begin{array}{llll}
w_1: & (\underline{m_2} & m_1^*) & m_3 \\
w_2: & \underline{m_1} & m_3 & \\
w_3: & (m_3^* & \underline{m_4}) & m_2 \\
w_4: & \underline{m_3} & &
\end{array}
$$

Figure 2.6: An SMTI instance with stable matchings of different sizes.

Weak stability

Irving [43] showed that finding a weakly stable matching in an SMT/SMTI instance is particularly easy: simply arbitrarily break the ties and find any stable matching in the resulting instance. In a sense, this method is the only "easy" thing about weak stability – almost everything else seems to be computationally difficult. In the SMT setting, minimum regret stable matchings and egalitarian stable matchings are both not only NP-hard to find, but are not approximable within $\Omega(n)$ unless P=NP [74]. It is also NP-hard even to determine if a given (man,woman) pair occurs in a stable matching (i.e. is a stable pair). Identifying any structural relationship involving weakly stable matchings is open, although one can construct SMT/SMTI instances that have neither man- nor woman-optimal stable matchings [88]. Efficiently enumerating all weakly-stable matchings also remains an open question.

We mentioned above, that, in general, the weakly-stable matchings of an SMTI instance can have different cardinality. This fact is illustrated when one uses Irving's tie-breaking algorithm: the ways in which the ties are broken can have a significant impact on the cardinality of the stable matchings obtained.

Example Figure 2.6 presents an SMTI instance with two different stable matchings of different cardinality. The example shows two weakly stable matchings, one denoted by underlining, and the other by star. The stable matching denoted by underlining is twice the size of the stable matching denoted by star. These matchings can be arrived at by running the (extended) Gale/Shapley algorithm on two of the different ways that the ties of the instance can be broken.

Manlove et al [74] first observed the fact that weakly stable matchings can have different sizes, and further showed that an arbitrary weakly stable matching M can be as little as one-half the size of a maximum cardinality stable matching. The obvious question then, is, can we find a maximum cardinality weakly stable matching in polynomial-time? Manlove et al [74] showed that finding a maximum cardinality weakly-stable matching is NP-hard, even in the highly restricted setting in

which the preference lists on one side are strictly ordered, and the preference list of each member of the opposite set is either strictly ordered or is a tie of length two (these conditions holding simultaneously). Henceforth, we let MAX-SMTI denote the problem of finding a maximum cardinality weakly stable matching of an SMTI instance.

Motivated by the hardness results of Manlove et al [74], researchers have been interested in finding polynomial-time approximation algorithms for MAX-SMTI. As a first step, we may observe that simply computing an arbitrary stable matching is an easy 2-approximation algorithm, because an arbitrary stable matching must be a maximal matching. A number of improvements have since appeared in the recent literature.

For the general case of SMTI, Iwama et al [53] gave a $2 - c\frac{\log n}{n}$ approximation algorithm, where c is a positive constant. This algorithm was subsequently improved to yield a performance guarantee of $2 - \frac{c'}{\sqrt{n}}$, where c' is a positive constant which is at most $1/4\sqrt{6}$ [55]. The first approximation algorithm for general SMTI with a constant performance guarantee better than two was given by Iwama et al [54], with a performance ratio of $15/8$.

The approximability of several special cases of SMTI have also been studied. Halldórsson et al [37] gave a $(2/(1 + T^{-2})$-approximation algorithm for the restricted case in which ties are only on one side, and the length of the longest tie is T. This bound can be improved to $13/7$ if the ties can appear in both men's and women's preference lists, but are restricted to being size at most two [37]. These same authors later described a randomized algorithm with an expected guarantee of $10/7$ for this special case with the additional restriction that ties appear only on one side [39]. Motivated by a restricted case of SMTI arising in practice [44, 107], Irving and Manlove [48] described a $5/3$-approximation algorithm for MAX-SMTI instances in which the ties appear only on one side, say, the women, and each woman may have at most one tie on her preference list, and this tie, if any, appears at the end of her list.

A recent landmark paper of Király [64] gave two simple algorithms that effectively superseded all previously known approximation algorithms for MAX-SMTI, (save only the randomized algorithm for the very special case studied in [39]). Király's first algorithm provides a $3/2$-approximation for the restricted case of MAX-SMTI in which ties are allowed to only appear on the women's side (this is the only restriction). The second algorithm provides a $5/3$-approximation for the general MAX-SMTI setting, in which no restrictions are placed on the problem input. In Chapter 3, we describe Király's approach in more detail, and give an approximation algorithm with an improved

performance guarantee.

From an inapproximability point of view, it is known that MAX-SMTI is APX-complete [38] and cannot be approximated within $21/19$ (unless P = NP) [37]. Yanagisawa [103] improved this bound to $33/29$, and also showed that MAX-SMTI cannot be approximated within $4/3$ under the assumption that the minimum vertex cover problem cannot be approximated within a factor of $2 - \epsilon$.

We mention one final result regarding weakly stable matchings. Let (α, β)-SMTI denote an SMTI instance in which the men's (women's) preference lists are of bounded maximum length α (β). Define (α, β)-MAX-SMTI similarly. Irving et al [51] showed that $(3, 3)$-MAX-SMTI is NP-hard, but $(2, \infty)$-MAX-SMTI is polynomial-time solvable (the ∞ here denotes preference lists of unbounded length). They furthermore showed that there exists a constant δ_0 such that $(4, 3)$-MAX-SMTI is not approximable within δ_0 unless P=NP. The inapproximability of $(3, 3)$-MAX-SMTI remains open.

2.2.6 The Hospitals/Residents problem

Moving away from the notion of indifference, we turn our attention to the many-one generalization of SMI, the so-called *hospitals/residents* problem (HR)[1] [30, 36]. The problem is so named because of its widespread application to centralised automated matching schemes that allocate graduating medical students (residents) to hospital posts, which we briefly mentioned in Section 2.1. The best known scheme is the National Resident Matching Program (NRMP) [105] in the United States, which annually allocates some 31,000 graduating medical students to their first job at a hospital. Similar schemes exist in Canada [106] and Scotland [107]. In all of these applications, the medical students produce preference lists ranking a subset of the hospitals, who in turns produce preference lists ranking a subset of the available residents. All of these centralized schemes incorporate various extensions of the Gale-Shapley algorithm to find stable matchings of medical students to hospitals.

Formally, an instance I of (HR) [30, 36] involves a set of *residents* $r_1, \ldots, r_n$ and a set of *hospitals* $h_1, \ldots, h_m$. Each hospital h_j has a *capacity* $c_j \in \mathcal{Z}^+$ indicating the maximum number of residents who could be assigned to h_j. Associated with each resident r_i is a strictly ordered preference list ranking a subset of the hospitals, his *acceptable hospitals*, and each hospital h_j ranks, again in strict order, those residents it finds acceptable. The definition of *acceptable pair* and *mutually acceptable*

[1] Gale and Shapley referred to this problem as the *College Admissions problem*, however this problem has now widely become known as the hospitals/residents problem.

$r_1:$ $\underline{h_1}$ h_2 h_3
$r_2:$ h_2 $\underline{h_1}$
$r_3:$ $\underline{h_2}$
$r_4:$ h_1 $\underline{h_3}$
$r_5:$ $\underline{h_3}$

$h_1:$ 2 : $\underline{r_1}$ $\underline{r_2}$ r_3 r_4
$h_2:$ 1 : r_4 $\underline{r_3}$ r_2 r_1
$h_3:$ 3 : r_1 $\underline{r_5}$ $\underline{r_4}$ r_2

Figure 2.7: An HR instance with a stable matching denoted by underlining.

pair are defined in the obvious way.

An *assignment* M is a set of mutually acceptable (resident,hospital) pairs. A *matching* is an assignment such that each resident is assigned at most one hospital, and each hospital h_j is assigned at most c_j residents [36, 30]. For a matching M, we define $M(r)$ to be the hospital resident r is assigned in M, and similarly we let $M(h_j)$ denote the set of residents assigned to h_j in M. If $|M(h_j)| < c_j$, h_j is said to be *undersubscribed*. If instead h_j is full to capacity, h_j is *fully subscribed*.

A blocking pair of a matching M is a resident r_i and hospital h_j such that:

1. r_i and h_j are mutually acceptable; and
2. r_i is unmatched, or r_i prefers h_j to $M(r_i)$; and
3. h_j is undersubscribed in M, or is fully subscribed and prefers r_i to its least-preferred assignee in M.

A matching is *stable* if it admits no blocking pair. Just as in the SMI setting, it is known that every instance of HR admits a stable matching, and that such a matching can be found in linear time using the extended Gale-Shapley algorithm [30];[36, Section 1.6]. Furthermore, the notion of man- and woman-optimal matchings can be extended to *resident-optimal* and *hospital-optimal* stable matchings [36, Section1.6]. Figure 2.7 shows an example of an HR instance, with the capacity of each hospital denoted by the number written next to the hospital.

Structure of HR

One of the first observations on the structural nature of the set of stable matchings for an HR instance is the so-called *Rural Hospitals Theorem*[2], which generalizes the fact that, in the SMI setting, the

[2]Historically, the NRMP found it problematic to match interns to unpopular hospitals, which were often found in the more rural areas of the United States. This theorem essentially explains why this happens, and shows that stability must

same number of agents are matched in all stable matchings. The theorem [91, 36] is described as follows:

Theorem 2.2.1 *(Rural Hospitals Theorem) [91, 36]. For a given hospitals/residents instance,*

(i) exactly the same residents are assigned in all stable matchings, so, in particular, all stable matchings have the same size;

(ii) each hospital is assigned the same number of residents in all stable matchings;

(iii) any hospital that is undersubscribed in one stable matching is matched with precisely the same set of residents in all stable matchings.

The notion of *dominance* amongst the set $\mathcal{M}$ of all stable matchings of an HR instance also generalizes from the SMI setting. Let M and M' be stable matchings for an HR instance. We say that M *dominates* M' (denoted $M \succeq M'$) if, for each assigned resident r, $M(r) = M'(r)$, or r prefers $M(r)$ to $M'(r)$. Hence each resident is at least as happy in M as in M'. Analogously with the SMI setting, $(\mathcal{M}, \succeq)$ forms a distributive lattice, with the maximum and minimum elements being the resident- and hospital-optimal stable matchings.

Couples

By the early 1970s, proportionally fewer residents were voluntarily participating in the NRMP. Checker [17] and later, Roth [88], attributed some of the decline to the existence of *couples*, i.e., pairs of (perhaps married) residents who wish to intern together or geographically close to one another. Such couples would choose to negotiate directly with hospitals to arrange their residency assignments rather than participate in the NRMP.

Today, the NRMP uses a modified algorithm to attempt to better accommodate couples. However, it is known that stable matchings need not exist when couples are present [88], and, moreover, it is NP-complete to decide if a stable matching exists [84]. On top of this, it has been shown that the NRMP algorithm may be prone to strategic manipulation by couples pretending to be single [67].

necessarily be sacrificed for unpopular hospitals to become full to capacity. Hence the strange name of this theorem.

$$
\begin{array}{rlll}
r_1: & \underline{h_1} & h_2 & h_3 \\
r_2: & h_2 & \underline{h_1} & \\
r_3: & \underline{h_2} & & \\
(r_4, r_5): & (h_1, h_3) & (\underline{h_3}, \underline{h_3}) &
\end{array}
\qquad\qquad
\begin{array}{llllll}
h_1: & 2 & : & \underline{r_1} & \underline{r_2} & r_3 \quad r_4 \\
h_2: & 1 & : & r_4 & \underline{r_3} & r_2 \quad r_1 \\
h_3: & 3 & : & r_1 & \underline{r_5} & \underline{r_4} \quad r_2
\end{array}
$$

Figure 2.8: An HRC instance with a stable matching denoted by underlining.

This discussion on couples is formalized by defining an important variant of HR called the *hospitals / residents problem with couples* (HRC). An instance of HRC involves both single residents and *couples* (pairs of residents) such that each resident belongs to at most one couple. Each couple (r_i, r_j) has a preference list over *pairs* of hospitals (h_k, h_l), representing the assignment of r_i to h_k and of r_j to h_l. Ronn [84] (see also [36, Section 1.6.6]) described a stability criterion for a matching in HRC that is a natural generalisation of the analogous concept in the HR context. As we mentioned above, it was Roth [88] who showed that an HRC instance need not admit a stable matching, while Ronn proved that the problem of deciding whether an HRC instance admits a stable matching is NP-complete, even if there are no single residents and each hospital has capacity one [84].

Example The example in Figure 2.8 gives an HRC instance in which residents r_1, r_2 and r_3 are single, and residents r_4 and r_5 are a couple with a joint preference list. There is a stable matching for this instance, denoted by underlining.

There has been much study devoted to HRC by economists in particular (see for example [88, 23, 13, 65, 66, 68], and references therein). From a computer science point of view, the problem is not nearly as well-studied, but there are a couple of exceptions. Marx and Schlotter [76] studied the parameterized complexity of HRC with the number of couples as a parameter. Dean et al [21] studied the so-called *Unsplittable Stable Marriage problem*, which they described in terms of assigning jobs with integral sizes (representing couples or groups of residents) to machines with capacities (representing hospitals). They provide a polynomial-time integral variant of the Gale-Shapley algorithm that finds a stable matching in which each machine is congested by at most the processing time of the largest job. Put differently, their algorithm finds a stable matching in which each hospital is oversubscribed by at most the size of the largest resident. In Chapter 5, we shall revisit HRC and also the unsplittable stable marriage problem.

$$
\begin{array}{lccccc}
r_1: & r_2 & \underline{r_3} & r_5 & r_4 & r_6 \\
r_2: & \underline{r_6} & r_1 & r_5 & r_3 & r_4 \\
r_3: & r_4 & \underline{r_1} & r_6 & r_2 & r_5 \\
r_4: & r_6 & r_1 & r_2 & \underline{r_5} & r_3 \\
r_5: & r_6 & r_1 & \underline{r_4} & r_3 & r_2 \\
r_6: & \underline{r_2} & r_5 & r_3 & r_4 & r_1
\end{array}
$$

Figure 2.9: An SR instance with a stable matching denoted by underlining.

2.2.7 Stable Roommates problem

The *stable roommates problem* (SR), first introduced by Gale and Shapley [30] is the nonbipartite generalization of SM. The definition of the problem, along with the notion of matching and stability generalize in the obvious ways, but for completeness, let us formally spell them out.

An instance of the stable roommates problem consists of one uniform set of agents $R = \{r_1, r_2, \ldots, r_n\}$. Each agent r_i supplies a preference list that ranks the members of $R - \{r_i\}$ in strict order of preference. A *matching* is a partition of the agents into disjoint pairs. A *blocking pair* relative to M is a pair (r_i, r_j) such that r_i prefers r_j to $M(r_i)$ and r_j prefers r_i to $M(r_j)$. A matching is *stable* if it admits no blocking pair. The *Stable Roommates problem with incomplete lists* (SRI) is the generalization of SR that allows incomplete preference lists, and an odd number of agents. The notions of blocking pair and stability are defined according to the obvious generalization of the SMI context. We again use m to denote the sum of the lengths of the preference lists of an SRI instance. Figure 2.9 gives an example of an SR(I) instance and a stable matching.

Notice that SMI is just a special case of SRI. In contrast to SM or SMI, however, not every SR or SRI instance admits a stable matching. The kind of obvious algorithms one would attempt to construct to generalize the Gale/Shapley algorithm are not sufficient to determine if an SR/SRI instance admits a stable matching. Knuth [69] asked if the roommates problem was polynomial-time solvable, or if perhaps this problem was NP-complete. Irving [42] resolved this question by presenting a $O(m)$-time algorithm that either returns a stable matching or reports that none exists. Although he described his algorithm in the SR setting, it clearly generalizes to the SRI case as well.

We briefly remark that the set of stable matchings $\mathcal{M}$ of an SRI instance forms a *meet semi-lattice* [35, 36]. There is also a similar notion of a rotation and rotation poset [35, 36], although these ideas are more involved than that of the bipartite case.

Stable roommates with ties

We can extend SR/SRI to allow for ties in the preference lists, (denoted by SRT and SRTI, respectively) which again gives rise to the notion of weak-, strong-, and super-stability. Ronn [84] showed that in contrast to the stable marriage setting, determining if an SRT instance admits any weakly stable matching is NP-complete. For the case of super-stability, Irving and Manlove [49] described an algorithm with running time $O(m)$ that either returns a super-stable matching or reports that none exists. In his PhD thesis, Scott [93] resolved the strong-stability case by giving a $O(m^2)$-time algorithm to either return a strongly-stable matching or report that none exists.

When the agents of an SRI instance are allowed to have a *capacity*, we obtain the so-called *stable fixtures problem* (SF). Irving and Scott [52] generalized Irving's algorithm [42] to this setting, obtaining a $O(m)$ time algorithm. Scott [93] also showed that SF is polynomial-time solvable under super-stability. The case of strong-stability for SF remains open.

Almost stable roommates

Since a stable matching for the roommates problem need not exist, it is natural to seek matchings that are "as stable as possible" in some well-defined sense. Tan [97] introduced the notion of a *stable partition*, which is a partitioning of the roommates instance into special cycles. These cycles have the property that if each agent r could somehow be matched to both of the agents adjacent to r in the given cycle, then there would be no blocking pairs. In so doing, Tan provided a method for describing a succinct certificate for checking whether or not an SR instance admits a stable matching without explicitly running Irving's algorithm. In a later paper, Tan [98] gave a linear-time algorithm for finding a so-called *maximum stable matching*, defined to be a largest possible set M of disjoint pairs such that there are no blocking pairs within M.

Perhaps the most natural way of finding an "almost stable" matching is to find a matching that admits the fewest blocking pairs. The decision version of this problem, then, is to determine whether a matching exists that admits at most K blocking pairs. Abraham et al [3] proved this problem is NP-complete, and is not approximable within $n^{\frac{1}{2}-\epsilon}$. However, when K is a fixed constant, they showed that the problem is solvable in polynomial time (with K being in the exponent).

Stable roommates and kidney exchange

In recent years, a renewed interest has been found in the stable roommates problem because it provides a way of modelling and solving problems related to real-world *kidney-exchange* programs, which exist in several different countries including the US and the UK.

Living donation is the most effective treatment that is currently known for kidney failure. However, a patient who requires a transplant may have a willing donor who cannot donate to them for immunological reasons. As a result, these incompatible patient-donor pairs may want to exchange kidneys with other pairs. Kidney exchange programs have already been established in several countries such as the Netherlands [62], the USA [85] and the UK [100].

We can capture this kidney exchange problem by creating a graph with a vertex for each patient-donor pair, and a directed arc (u, v) for every pair of vertices u, v such that the donor in the pair corresponding to vertex u can donate a kidney to the patient in the pair corresponding to vertex v. A set of disjoint cycles in this graph corresponds to a cyclic kidney exchange. In practice, however, we cannot find arbitrarily long cyclic exchanges of kidneys, as all operations along a cycle have to be carried out simultaneously. Hence the length of the exchanges are typically bounded to two or three in practice.

In most of the current programs the goal is to maximise the number of patients that receive a suitable kidney in the exchange [86, 87, 92, 2] by regarding only the feasibility of the grafts. Some more sophisticated variants consider also the differences between suitable kidneys. When the value of a kidney for a given patient can be quantified with a numerical value, the "total benefit" could be maximised [94]. However, this is not always feasible and instead the differences between suitable kidneys for a given patient give rise to a preference list. Hence *stability* could be the primary objective of a kidney exchange [90, 14, 15, 9]. When the cyclic exchanges are limited to being of length at most two, the underlying problem is precisely the stable roommates problem.

2.2.8 Three-dimensional stable matchings

Knuth [69] asked if the stable marriage problem could be generalized to three sets, so that the instance contains not only men and women, but also a third set, which he called *dogs*. Let A, B, C be disjoint sets of men, women, and dogs, respectively, and let $|A| = |B| = |C| = n$. In response

$m_1:$	w_2	w_3	w_1	w_4	$w_1:$	d_1	d_2	d_3	d_4	$d_1:$	m_2	m_3	m_1	m_4
$m_2:$	w_2	w_1	w_4	w_3	$w_2:$	d_2	d_3	d_1	d_4	$d_2:$	m_2	m_4	m_1	m_3
$m_3:$	w_1	w_3	w_2	w_4	$w_3:$	d_3	d_4	d_1	d_2	$d_3:$	m_3	m_1	m_4	m_2
$m_4:$	w_3	w_1	w_4	w_2	$w_4:$	d_4	d_3	d_2	d_1	$d_4:$	m_1	m_4	m_3	m_2

Figure 2.10: An instance of cyclic 3DSM

to Knuth's open question, several different variations of this problem have been considered. The common goal of these variations is to find a matching M that is a set of *triples* from the set $A \times B \times C$. Similar to the SMI setting, the common notion of stability in the variants of this problem involves the absence of any blocking triples $(m, w, d) \notin M$ such that a subset of $\{m, w, d\}$ would somehow improve and all of them must be at least as happy if matched together instead of staying with their current triples. The primary differences in the variations of this problem arise in the definition of stability, and also the nature of the preference lists.

In the *three-dimensional stable matching problem* (3DM), each agent has a preference list ranking all pairs of the other two sets. A *matching* is a set of disjoint triples, and a matching is stable if there exists no *blocking triple* $T = (m, w, d) \notin M$ such that every member of T prefers T over their current triple.

Alkan [7] gave the first example of an instance of 3DM where no stable matching exists. Ng and Hirschberg [83] proved that the problem of deciding whether a stable matching exists, given an instance of 3DM, is NP-complete; later Subramanian [96] gave an alternative proof for this. Huang [40] proved that the problem remains NP-complete even if the preference lists are *consistent* in a formal sense.

As an open problem, Ng and Hirschberg [83] mentioned *cyclic 3DSM*, where men care about only the women, women care about only the dogs and dogs about only the men. Boros et al. [11] showed that if the number of agents n, is at most 3 in each set, then a stable matching always exists. Eriksson et al. [26] proved that this also holds for $n = 4$ and conjectured that a stable matching exists for every instance of cyclic 3DSM. The example in Figure 2.10 shows a cyclic 3DSM instance.

Danilov [20] provided a polynomial-time extension of the Gale/Shapley algorithm for a restricted version of 3DSM in which the men care *primarily* about the women they are matched to, and women care primarily about the men. More precisely, a strictly ordered preference list of the women can be derived from a man's preference list of (woman,dog) pairs, and such a strictly ordered list of the

men can be derived from each woman's preference list of (man,dog) pairs. The dog's preference list consists of unrestricted (man,woman) pairs. Danilov defined a reasonable notion of stability in this setting and showed that a stable matching always exists. Moreover, his results generalize to five sets of agents [20].

In Chapter 6, we revist the cyclic 3DSM problem.

2.3 Partial preference information

Matching problems with partial preference information often take the form of two disjoint sets of agents, with one of the sets of agents expressing preferences over members of the other set. Such problems are often described in terms of assigning applicants to houses or applicants to posts. We will use the latter terminology.

In an instance I of the *post allocation problem* (PA) we are given a set of n_1 *applicants* $\{a_1, a_2,\ldots,a_{n_1}\}$, and a set of n_2 *posts* $\{p_1, p_2, \ldots p_{n_2}\}$. Associated with each applicant a_i is a preference list which ranks a subset of the posts. This subset comprises the *acceptable posts* of a_i, and an (applicant,post) pair (a_i, p_j) is an *acceptable pair* if and only if a_i finds p_j acceptable. We let m denote the sum of the lengths of the preference lists of the instance, and $n = n_1 + n_2$ be the number of applicants plus the number of posts. Notice the key difference between this problem and SMI is that the posts of the instance do not express any form of preference. A *matching* M is a disjoint subset of acceptable pairs of I. When ties are allowed in the preference lists, we obtain an instance of the *post allocation problem with ties* (PAT) with the notion of acceptable post, acceptable pair, and matching all generalizing in the obvious way.

Since the posts of a PA/PAT instance do not have preference lists, the notion of a "stable" matching does not have any real meaning in this context. We need some different optimality criteria. Notice that simply finding a maximum cardinality matching is not a satisfactory approach, as an arbitrary maximum cardinality matching will not take into account the preferences of the applicants. In what follows we will review two of the most fruitful approaches researchers have taken to find optimal matchings for PA/PAT instances. The first thread involves finding matchings with good *profiles* (which we define below). The second involves finding matchings that are *pareto optimal*, or even *popular* – two terms we will define also below.

$$
\begin{array}{lll}
a_1: & \underline{p_1} & p_4^* \\
a_2: & \underline{p_2} & p_5^* \\
a_3: & p_3 & \underline{p_6}^* \\
a_4: & \underline{p_3}^* & \\
a_5: & p_3 & p_1^* \\
a_6: & p_3 & p_2^*
\end{array}
$$

Figure 2.11: A comparison of rank-maximal and maximum matchings.

2.3.1 Profile-based optimality

Suppose I is an instance of PA. The *profile* of a matching M is the (n_2+1)-tuple $(x_1, \ldots, x_{(n_2+1)})$ where, for each i $(1 \leq i \leq n_2+1)$, x_i is the number of applicants who are matched in M with their i^{th}-choice post. An applicant who is unmatched is considered to be matched to his $(n_2+1)^{th}$-choice post, regardless of the length of his preference list. There are various ways we can quantify the quality of a matching in terms of its profile.

Suppose that $x = (x_1, \ldots, x_{n_2})$ and $y = (y_1, \ldots, y_{n_2})$ are profiles. We say that x *left-dominates* y (denoted $x \succ_L y$) if, for some j, $x_i = y_i$ for $1 \leq i < j$ and $x_j > y_j$. A *rank-maximal* matching is a matching whose profile is maximal with respect to $\succ_L$. A rank-maximal matching need not be unique, but, for a given instance, all must have the same size.

Example The example in Figure 2.11 demonstrates the difference between a rank-maximal matching and a maximum cardinality matching. The example consists of a PA instance with a rank-maximal matching denoted by underlining and a maximum cardinality matching denoted by star. The rank-maximal matching is smaller than a maximum matching, but assigns more agents to their first choice.

A rank-maximal matching can be computed by a reduction to an instance of the maximum weight bipartite matching problem (MWBM). The resulting instance I' of MWBM has the property that the weights on the edges are of the form n^{k-i} for an edge representing an agent's i^{th} choice. Using the algorithm of Gabow and Tarjan [29], and making the standard assumption that numbers of magnitude $O(n)$ can be handled in constant time and space, a rank-maximal matching can be found in $O(k^2\sqrt{n}m \log n)$ time. The space requirement is $O(km)$. Irving et al [45] improved this by describing a direct algorithm for finding a rank-maximal matching in $O(\min(n + C, C\sqrt{n})m)$ time, where $C \leq k$ is the maximum rank that appears in an optimal solution. Later, Michail [81]

gave a different reduction to MWBM which achieves the same running time as that of Irving et al [45].

A rank-maximal matching can in fact be significantly smaller than an arbitrary maximum matching, so, typically, other definitions of profile optimality first require that the size of the matching must be maximum, and then require that the profile is optimal in some sense. Define a profile $x = (x_1, x_2, \ldots, x_{n_2})$ to be *feasible* if there is some matching with profile x. A *feasible q-profile* is a feasible profile x with $\sum x_i = q$. A feasible q-profile x is *q-left maximal* if there is no other feasible q-profile that left-dominates x. A matching M whose profile is q-left maximal is called a *greedy q-matching*. When q is the size of a maximum matching, a greedy q-matching is called a *greedy maximum matching*.

A different form of optimality arises when we seek a maximum cardinality matching that minimizes the number of applicants who obtain their $(n_2 + 1)^{th}$ choice (i.e., are unmatched), and subject to that, minimizes the number of applicants who receive their ${n_2}^{th}$ choice, and so on. To define this formally, define a second total order $\prec_G$ on two feasible q-profiles so that $x = (x_1, \ldots, x_{n_2+1}) \prec_G y = (y_1, \ldots, y_{n_2+1})$ if, for some j, $x_i = y_i$ for $j < i \leq n_2 + 1$ and $x_j < y_j$. A matching that is maximal with respect to $\prec_G$ is a *generous* matching. When q is the size of an arbitrary maximum matching, a generous q-matching is called a *generous maximum matching*.

One can also reduce greedy and generous matchings to MWBM, again by assigning suitably large weights to the edges of the derived instance. The resulting time requirement is $O(k^2\sqrt{n}m \log n)$, although Mehlhorn and Michail [79] showed this can be reduced to $O(k\sqrt{n}m \log n)$ time. It remains an open question to construct faster, direct algorithms that perhaps do not require the use of MWBM for greedy and generous matchings.

2.3.2 Pareto optimal matchings

We move on to other kinds of non-profile based optimality. For an PA/PAT instance I, we say that an applicant *prefers* a matching M to a matching M' if (i) a is matched in M and unmatched in M', or (ii) a is matched in both M and M' and prefers $M(a)$ to $M'(a)$ (where $M(a)$ is again the post applicant a is assigned in matching M). For two matchings M and M', let $\alpha(M, M')$ denote the number of applicants who prefer M to M'. A matching M is said to be *pareto optimal* if there is no matching M' with $\alpha(M', M) > 0$ and $\alpha(M, M') = 0$. Intuitively, M is pareto optimal if there

is no subset of the applicants who can *all* improve while leaving everyone else no worse off.

For every PA instance, at least one pareto optimal matching always exists, and one can easily be computed with the so-called *serial-dictatorship mechanism* (see, e.g., [1]). However, pareto optimal matchings can have different sizes, and the matching obtained by the serial-dictatorship mechanism will not, in general, return a maximum pareto optimal matching. Finding a maximum cardinality pareto-optimal matching can be solved by a reduction to the assignment problem, but Abraham et al [4] found a faster, more direct algorithm having $O(\sqrt{n}m)$ time complexity. Since every PA/PAT instance has a pareto optimal matching with the same cardinality as an arbitrary maximum matching [4], any improvement in the running time of this algorithm would imply a faster algorithm for finding a maximum matching in a bipartite graph.

2.3.3 Popular matchings

A matching M' is said to be *more popular* than a matching M if $\alpha(M', M) > \alpha(M, M')$. A matching M is *popular* if there is no matching M' with $\alpha(M', M) > \alpha(M, M')$. A moment's reflection reveals that a popular matching is a stronger notion of optimality than pareto optimality, as a popular matching must be pareto optimal. The concept of a popular matching is attributed to Gardenfors [32] who studied the popular matchings in the SMI context. He showed that every stable matching of an SMI instance is also popular; hence popular matchings always exist in the SMI setting. In contrast, there exist PA/PAT instances which have no popular matching, and, if they do exist, they can have different sizes. The goal of the *popular matching problem* (POP-M), then, is to find a popular matching or report that none exists.

Example The example in Figure 2.12 denotes a POP-M instance with two popular matchings that differ in cardinality. The example consists of a POP-M instance with two popular matchings, one denoted by underlining, and the other by star. The popular matching denoted by star is twice the size of the popular matching denoted by underlining.

Abraham et al [5] described an $O(n+m)$ time algorithm which computes a largest possible popular matching, or reports that no popular matching exists for a PA instance. In the case of PAT, they gave an algorithm with $O(\sqrt{n}m)$ time complexity.

The results of Abraham et al [5] led to a number of subsequent papers covering variants and ex-

$$
\begin{array}{lllll}
a_1: & \underline{p_1} & p_2 & p_4^* & \\
a_2: & \underline{p_2} & p_3 & p_1 & p_5^* \\
a_3: & \underline{p_3} & p_6^* & & \\
a_4: & p_1^* & & & \\
a_5: & p_2^* & & & \\
a_6: & p_3^* & & &
\end{array}
$$

Figure 2.12: The difference in cardinality of different popular matchings.

tensions of the popular matching problem. Manlove and Sng [75] studied the *capacitated popular matching problem*, C-POP-M in which each post has a capacity, defined to be the maximum number of applicants that can be assigned to it. Manlove and Sng gave a $O(\sqrt{C}n_1 + m)$ time algorithm for C-POP-M, where C is the sum of the capacities of the posts. Mestre [80] gave a linear time algorithm for a version of the problem in which each applicant has an associated weight; the goal is to find a matching M with the property that there is no other matching M' preferred by a weighted majority of agents. Mahdian [71] showed that popular matchings exist with high probability for random instances of POP-M if the number of posts exceeds the number of applicants by a small constant multiplicative factor. Abraham and Kavitha [6] studied a dynamic version of POP-M allowing for applicants and posts to enter and leave the instance, and for applicants to arbitrarily change their preference lists. They showed the existence of a so-called 2-step *voting path* to compute a new popular matching after every such change, assuming that a popular matching exists. McCutchen [77] focused on instances of POP-M for which no popular matching exists, defined two notions of 'near popularity', and proved that for each of these it is NP-hard to find a matching that is as near to popular as possible. Huang et al [41] built upon the work of McCutchen with a study of approximation algorithms in the context of near popularity. Kavitha and Nasre [60] described algorithms to determine an *optimal* popular matching for various interpretations of optimality; in particular they gave a $O(n^2+m)$ time algorithm to find *minimum cost*, *rank-maximal* and *fair* popular matchings (a fair popular matching being a synonym for a generous popular matching). To cope with POP-M instances which do not admit a popular matching, Kavitha et al [59] defined the notion of a *mixed popular matching*, and showed that a mixed popular matching exists for every POP-M instance. Very recently, Kavitha and Nasre explored popular matchings with variable job capacities [61], a problem they show is NP-complete. We study popular matchings further in Chapter 7.

Chapter 3

An improved approximation algorithm for MAX-SMTI

3.1 Introduction

A crucial objective of many centralised matching schemes is to find matchings that match as many agents as possible – without sacrificing stability. As the reader may recall from Section 2.2.2, finding a maximum stable matching in the SMI setting is straightforward, as all stable matchings must have the same size. However, allowing agents to have *ties* in their preference lists changes everything. As we mentioned in Section 2.2.5, weakly stable matchings for SMTI instances can have different sizes, and the problem of finding a maximum weakly stable matching (MAX-SMTI) is NP-hard [74].

We surveyed all of the relevant background for SMTI, and the long sequence of approximation results for MAX-SMTI in Section 2.2.5. For our purposes in this chapter, we need only recall that the most recent approximation algorithms for MAX-SMTI were due to Király, who gave a $\frac{3}{2}$-approximation algorithm for the restricted case of MAX-SMTI in which ties are not allowed to appear in the men's preference lists, and a $\frac{5}{3}$-approximation algorithm for the general MAX-SMTI setting (meaning that there are no restrictions on the input). These results effectively superseded all previously known approximation algorithms for MAX-SMTI, except for a very special case that was studied by Halldórsson et al [39].

Our contribution in this chapter is to provide a three phase $\frac{3}{2}$-approximation algorithm for MAX-SMTI (no restrictions on the input), which improves upon Király's general performance guarantee of $\frac{5}{3}$. Our work builds from some ideas used in Király's $\frac{3}{2}$-approximation algorithm (henceforth *Király's algorithm*) in the sense that one of the three phases of our algorithm uses a generalisation of this algorithm.

3.2 Background

3.2.1 Király's algorithm

To make our presentation self-contained, we describe a version of Király's algorithm using the concept of *promotion* from a tie rather than that of extra score used by Király [64]. As input to this algorithm, the men of the instance have strictly ordered preference lists and the women have no restriction on the nature of the ties in their preference lists. The idea behind the algorithm is to allow men to make proposals to the women on their preference lists, as in the Gale/Shapley algorithm, but with an additional feature. The change is that a man m who is unmatched after proposing to every woman on his preference list – we use the term *exhausted* to describe such a man – is given one "second chance" in which m is promoted ahead of each tie in which he appears, and is then allowed to propose to each woman on his list a second time.

At the start of the algorithm, each man is set to be *unmatched*, *unpromoted*, and *unexhausted*. The main body of the algorithm is a while loop, which continues as long as there exists a man m who is (i) unmatched and (ii) either unpromoted or unexhausted (or both). If m is exhausted, m is set to be promoted. The operation of promoting m involves examining each woman w who finds m acceptable, and, if m is in a tie of size at least 2 on w's list, m is promoted immediately ahead of this tie on w's preference list. Furthermore, m is set to be unexhausted and is *reactivated*, meaning he will now begin again making proposals to women starting from the beginning of his preference list. The algorithm proceeds by m proposing to the next woman w on his preference list to whom he has not yet proposed (or to whom he has proposed only once, if he has been reactivated). When a man m proposes to a woman w, she rejects her current partner (if any) and accepts m if m is a *strict* improvement for her, taking into account any promotions that may have occurred. Otherwise, she retains her current partner and rejects m. On rejection, a man becomes (or remains) unmatched.

set every man to be unmatched, unpromoted, and unexhausted

while $\exists m$ such that m is unmatched and (m is unpromoted or m is unexhausted):

if m is exhausted:

promote m and set m to be unexhausted

reactivate m so that he begins proposing again from the start of his list

$w \leftarrow$ next woman on m's preference list /* m proposes to w */

if w is unmatched:

$M \leftarrow M \cup \{(m, w)\}$ /* w accepts m */

else if w prefers m to her partner m':

$M \leftarrow M \cup \{(m, w)\} - \{(m', w)\}$ /* w rejects m' and accepts m */

if w is the last woman on the list of m':

set m' to be exhausted

else: /* w rejects m */

if w is the last woman on his list:

set m to be exhausted

Figure 3.1: Király's algorithm

When a man has been rejected by every woman on his list, he is set to be exhausted.

When Király's algorithm terminates, each man is either (i) matched (possibly having been previously promoted as well), or (ii) promoted, exhausted, and unmatched. A pseudocode description of Király's algorithm is given in Figure 3.1.

3.2.2 Gallai-Edmonds decomposition theorem

Phase 2 of our approximation algorithm uses a classical result regarding bipartite matchings known as the *Gallai-Edmonds decomposition theorem*. In this section we review the parts of this theorem that we will need in the forthcoming sections. To this end, let $G = (U \cup V, E)$ be a bipartite graph and M a maximum cardinality matching of G. With respect to M, we partition the vertex set of G in the following way. A vertex v is said to be *odd* (respectively, *even*) if there exists an odd (respectively, even) length alternating path from some unmatched vertex to v. A vertex v is said to be *unreachable* if there is no alternating path to v beginning at some unmatched vertex. The following Gallai-Edmonds decomposition theorem provides an important characterisation of the set

of maximum cardinality matchings of G with respect to this vertex partition [70, 45].

Theorem 3.2.1 *(Gallai-Edmonds decomposition) Let $G = (U \cup V, E)$ be a bipartite graph and M be a maximum cardinality matching for G. Let $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ be the set of even, odd, and unreachable vertices as defined above with respect to G and M. Then*

1. *$\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ are pairwise disjoint. Every maximum matching of G partitions the vertex set of G into the same sets of even, odd, and unreachable vertices.*
2. *In any maximum-cardinality matching of G, every vertex in $\mathcal{O}$ is matched with some vertex in $\mathcal{E}$, and every vertex in $\mathcal{U}$ is matched with another vertex in $\mathcal{U}$. The size of a maximum-cardinality matching is $|\mathcal{O}| + |\mathcal{U}|/2$.*
3. *There is no edge in G connecting a vertex in $\mathcal{E}$ with a vertex in $\mathcal{U}$.*

We note that the Gallai-Edmonds decomposition of a bipartite graph can be obtained as a by-product of a maximum cardinality matching algorithm.

3.3 The approximation algorithm

Our approximation algorithm consists of 3 phases. A pseudocode description is given in Figures 3.2 and 3.3. In general, multiple calls are made to phases 1 and 2, as phase 1 may pass control to phase 2 and vice versa. Control is passed to phase 3 at most once. In the first phase, we use an approach somewhat similar to the Király algorithm, adapted to take into account the ties in the men's preference lists. In this phase, men again may become promoted, exhausted, and matched, but may also enter a different state in which they become *stalled*. The meaning of this state will become clear in the description of phase 1 below. Prior to calling the phase 1 algorithm for the first time, each man is set to be unmatched, unpromoted, unexhausted, and unstalled, and the matching M is initialised to be empty.

3.3.1 Phase 1

For ease of exposition, we think of the entries on a man's preference list as being a series of ties; some ties may be of size exactly one. In the first phase of the algorithm, the men iteratively make

$M \leftarrow \emptyset$

set all men to be unmatched, unpromoted, unexhausted, and unstalled

Phase 1:

while $\exists m$ such that m is unmatched and unstalled and (m is unpromoted or m is unexhausted):

  if m is exhausted:

    promote m and set m to be unexhausted

    reactivate m, set m's current tie to be his first choice

  $t \leftarrow m$'s current tie

  if $|t| \geq 2$:

    if t contains exactly one unmatched woman w:

      promote w ahead of t

    else if t contains no unmatched woman:

      break t arbitrarily

    else:

      set m to be stalled

  else:

    $w \leftarrow$ only woman in t /* m proposes to w */

    if w is unmatched:

      $M \leftarrow M \cup (m, w)$ /* w accepts m */

      unstall the appropriate men, if any

    else if w prefers m to her partner m':

      $M \leftarrow M \cup \{(m, w)\} - \{(m', w)\}$ /* w rejects m' and accepts m */

      if w is the last woman on his list:

        set m' to be exhausted

    else: /* w rejects m */

      if w is the last woman on his list:

        set m to be exhausted

if the set S of stalled men is empty:

  return M

else:

  invoke phase 2

Figure 3.2: Phase 1 of the approximation algorithm.

Phase 2:

Construct the phase-2 graph $G = (U \cup V, E)$

$N \leftarrow$ maximum cardinality matching in G

identify the sets $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$

$N' \leftarrow$ subset of N obtained by removing all pairs

    (m, w) such that $m \in \mathcal{O}$ and $w \in \mathcal{E}$

if $N' = \emptyset$:

    invoke phase 3

else:

    for $(m, w) \in N'$:

        promote w ahead of m's current tie /* m proposes to w */

        $M \leftarrow M \cup (m, w)$

        set m to be unstalled

    unstall all men in U who are unmatched in N

    invoke phase 1

Phase 3:

for $(m, w) \in N$ /* m proposes to w */

    $M \leftarrow M \cup (m, w)$

return M

Figure 3.3: Phases 2 and 3 of the approximation algorithm.

proposals to the women on their preference lists in a similar way to the Király algorithm. The main body of this phase is again a while loop, which continues as long as there exists a man m who is (i) unmatched and unstalled, and (ii) unpromoted or unexhausted (or both). In general, there may be many men who satisfy the loop condition, in which case the choice of m is made arbitrarily. If m is exhausted, he is promoted, set to be unexhausted, and is *reactivated*, precisely as described in the Király algorithm. Next, we let t denote the first tie on m's preference list containing a woman w to whom m has not yet proposed (or to whom he has proposed only once, if he has been reactivated) . We refer to t as m's *current tie*. The algorithm then proceeds based on the following cases concerning t. The first case (i) is if the size of t is at least 2. If t also contains exactly one unmatched woman w, w is promoted ahead of t on m's preference list. If instead t contains no unmatched women, t is broken arbitrarily on m's preference list, creating a total order of these women to replace t on his list. Otherwise, t must contain at least 2 unmatched women, and m is set to be stalled. The second case (ii) is if the size of t is exactly one. In this case m proposes to w, the only woman in t. When a man m proposes to a woman w, she accepts if m is a strict improvement for her, taking into account any promotions that have been made. Otherwise, she rejects m. When an unmatched woman w becomes matched, the men who, as a result, have now just one unmatched woman in their current tie are unstalled. As before, when a man has been rejected by every woman on his preference list, he is set to be exhausted.

The primary task of phase 1 ends with the termination of this while loop. At this point in the execution of the approximation algorithm every man m is in exactly one of three categories: (i) m is matched to an acceptable woman (and possibly is promoted as well), or (ii) m is exhausted, promoted and unmatched, having been rejected by every woman on his preference list despite his promotion, or (iii) m is stalled.

If the set S of stalled men is empty, the algorithm returns the current matching and halts. Otherwise, we proceed to phase 2.

3.3.2 Phase 2

The goal of the algorithm in this phase is to attempt to match a certain subset of the stalled men. We construct a bipartite graph $G = (U \cup V, E)$ with U being the set of men in S and V being the set of unmatched women appearing in the current tie of at least one man in S. We refer to these

men and women and the vertices of G representing them interchangeably. The set of edges are those (man,woman) pairs (m, w) such that $w \in V$ appears in m's current tie. We call this graph the *phase-2 graph*. The algorithm then computes a maximum cardinality matching N in G.

We proceed by removing selected pairs from N in the following way. We identify the sets $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ of vertices as described according to the Gallai-Edmonds decomposition theorem in Section 3.2.2. All pairs in N consisting of a man $m \in \mathcal{O}$ and a woman $w \in \mathcal{E}$ are removed from N, yielding a new matching $N' \subseteq N$. One of the crucial properties of N' (proved in Lemma 3.4.2) is that, for each man m who is matched in N', if $w_1, w_2, \ldots w_t$ are the unmatched (in M) women in m's current tie, then $w_1, w_2, \ldots, w_t$ are also matched in N'. This important property of N' is key to the establishment of the performance guarantee of the algorithm.

If N' is empty, we proceed to phase 3. Otherwise, for every pair $(m, w) \in N'$, w is promoted ahead of m's current tie. Man m then proposes to w, who accepts because she is unmatched in M, and this pair is added to M. All the men matched in N' are now set to be unstalled.

At this point in phase 2, the assignment of any man not in S has remained unchanged, as the matching has changed only by matching previously unmatched women to men in S. However, the situation of the men who were in S at the beginning of phase 2 has, of course, changed. We claim (proved in Lemma 3.4.2) that those men m remaining in S fall into one of two categories: (i) m was matched in N, is not matched in N', and still has at least 2 unmatched women in his current tie, or (ii) m was unmatched in N and every woman in his current tie is now matched in M. The men in (ii) are set to be unstalled, and the algorithm returns to phase 1.

3.3.3 Phase 3

Phase 3 takes as input the current matching M along with the matching N constructed in the execution of phase 2 that passed control to phase 3. The algorithm arrives at phase 3 if and only if the matching N' of phase 2 is empty. We will show (in Lemma 3.4.1) that this implies that N matches every man in S. The algorithm terminates after the man in each pair in N proposes to his partner in N – all of these women are single – and these pairs are added to M. The current matching M is returned.

3.4 Correctness

Let us establish a few key properties of the algorithm, and verify certain claims made in the description of the pseudocode.

Lemma 3.4.1 *Let S denote the set of stalled men at the start of an arbitrary execution of phase 2 of the approximation algorithm. If the matching N' constructed in this call is empty, then the corresponding maximum cardinality matching N matches every man in S.*

Proof Suppose N' is empty, and that a man m is unmatched by N. Let w be an arbitrary neighbour of m in G. Since m is not matched in N, m is even (i.e. $m \in \mathcal{E}$), and therefore w is odd ($w \in \mathcal{O}$). Since N is maximal, w was matched to a man m' in N, who therefore must also be even. But this implies the pair (m', w) could not have been removed from N, as it consists of an even man and an odd woman. □

Corollary 3.4.1 *Phase 3 of the approximation algorithm finds a matching that matches every man who was in S in the preceding execution of phase 2.*

Proof By Lemma 3.4.1, when control of the algorithm reaches phase 3, every man in S is matched, for control is passed to this point only if N' is empty. □

Lemma 3.4.2 establishes the key properties of the matchings N and N' constructed in phase 2 of the approximation algorithm.

Lemma 3.4.2 *Let m be a stalled man in the set S with current tie t at the start of an arbitrary execution of phase 2. Then, exactly one of the following is true of m when that execution of phase 2 ends (i.e., the instant before either* **invoke** *statement in phase 2 is executed).*

1. *m was matched in N', so m is now matched in M to a woman in t, and every woman in t is matched in M.*
2. *m was matched in N but not in N', m's current tie is still t, and there are at least two women in t who are still unmatched in M.*

3. *m was unmatched in N, m's current tie is still t, and every woman in his current tie is now matched in M.*

Proof (1) Suppose m was matched in N'. Then, m was an even or unreachable vertex with respect to M. Therefore, all neighbours of m in G are either odd or unreachable, and could not have been deleted from N, for only even women are removed from N. It follows that all of m's neighbours are in N', and therefore they all receive proposals in this execution of phase 2, and are matched in M.

(2) If instead m is matched to a woman w in N but is unmatched in N', then m was removed from N because he is an odd vertex. We establish the claim by showing there is another even woman $w' \neq w$ who is adjacent to m and is unmatched in N' as well. To see this, consider the path of odd length that makes m an odd vertex. This path cannot reach him via his partner in the matching, for alternate edges in that path would have to be edges in the matching. Hence the first edge in the path would be in the matching (since the last edge is), contradicting the fact that the starting vertex in the path must be unmatched. Therefore this path must reach him from another neighbouring vertex w', which must be even. This woman is unmatched in N', for she can only be matched to an odd man in N or unmatched in N.

(3) Finally, if m is unmatched by N he is an even vertex. All women in his current tie are therefore odd vertices, are matched in N because N is maximal, and could not have been removed from N. Therefore, these women are all matched in N' and all receive proposals in this execution of phase 2, and hence are matched in M.

Having considered every possibility of the outcome of m's participation in phase 2, the lemma is established. □

Lemma 3.4.3 *On termination of the approximation algorithm, any man who remains unmatched has been promoted, and has been rejected by every woman on his list even after becoming promoted.*

Proof The execution of the algorithm can only halt in one of two places. The first place is at the end of phase 1, on the condition that there are no stalled men. This implies that every unmatched man is promoted and has still been rejected by every woman on his list. The other point at which

the algorithm may terminate is in phase 3. Now, control reaches phase 3 only if, in phase 2, it is discovered that N' is empty, implying that N matches every man in S by Corollary 3.4.1. Notice that when this happens nothing is done in phase 2 to modify the assignment of any agent, rather phase 2 simply passes control to phase 3, which matches every man in S. Hence, the unmatched men are those who were unmatched after the final call to phase 1, and, as described above, they must have become exhausted while promoted. □

Lemmas 3.4.4 and 3.4.5 establish the stability of the matching output by the approximation algorithm.

Lemma 3.4.4 *Suppose a woman w becomes matched to a man m at some point in the execution of the approximation algorithm. Then w only rejects m if she accepts a proposal from a man ranked at least as highly as m on w's (original) preference list.*

Proof Matched women can only change their partner in one place in the approximation algorithm, and that is when receiving a proposal in phase 1 from a man they strictly prefer, possibly after promotions, to their current partner. This new suitor must be ranked at least as highly as w's current partner on w's original preference list. □

Lemma 3.4.5 *The matching M returned at the end of the approximation algorithm is a stable matching.*

Proof Suppose that (m, w) blocks M. The essence of the approximation algorithm from a man's point of view is a left-to-right sweep of his preference list in which, if necessary, he becomes promoted and again makes another left-to-right sweep of his preference list. Hence, for m to prefer w, he must have proposed to her at least once, whether it be in phase 1 or phase 2 (he cannot have proposed to her in phase 3, for otherwise they would be matched in M). The fact that w has rejected m along with Lemma 3.4.4 implies that w does not prefer m to her current partner in M, and hence (m, w) does not block M. □

Lemma 3.4.6 *The approximation algorithm runs in $O(n^{3/2}m)$ time, where n is the sum of the numbers of the men and women and m is the sum of the lengths of the preference lists.*

Proof The algorithm essentially constitutes one or two partial or complete left to right sweeps of the men's preference lists, interleaved with calls to phase 2. The total number of calls to phase 2 is bounded by the number of men, as each call to phase 2 either strictly increases the size of M or passes control to phase 3, in which phase the algorithm terminates. Let $|V|$ and $|E|$ denote the numbers of vertices and edges, respectively, in the phase-two graph. Any one execution of phase 2 requires a total of $O(\sqrt{|V|}|E|) = O(\sqrt{n}m)$ time, as the construction of N is the dominant step of phase 2. In the worst case, $\Omega(n)$ calls could be made to phase 2, each of which computes a matching N of size $\Omega(n)$ but a matching N' of size $O(1)$. These successive calls to phase 2 would clearly dominate the complexity, yielding a bound of $O(n^{3/2}m)$. □

3.5 The performance guarantee

For a given instance of MAX-SMTI, let M be the stable matching returned by the approximation algorithm and let M_{opt} denote an optimal stable matching for a given instance of MAX-SMTI. Consider the symmetric difference $M \oplus M_{opt}$ of these two matchings. The components of the underlying graph of $M \oplus M_{opt}$ consist of alternating cycles and paths. Each cycle component in $M \oplus M_{opt}$ is of even length, so the ratio of M-edges to M_{opt}-edges in these components is one. For an alternating path component, the ratio of M_{opt}-edges to M-edges is always at most 3/2 except for a component that is a path of length 3 with its endpoints in M_{opt}. Therefore, if we can establish that $M \oplus M_{opt}$ contains no such path, we will have shown that the ratio of M_{opt}-edges to M-edges in each component is at most 3/2, establishing that the algorithm is a $\frac{3}{2}$-approximation algorithm. Lemma 3.5.1 is the missing piece of the puzzle to establish the performance guarantee.

Lemma 3.5.1 *Let $P_3 = w' - m - w - m'$ be an alternating path in $M \oplus M_{opt}$ with $(m, w) \in M$ and (m, w'), $(m', w) \in M_{opt}$ (as described in Figure 3.4). Then, the following facts hold.*

(i) The man m' in P_3 must be exhausted and promoted.

(ii) The man m in P_3 was never promoted by the approximation algorithm.

(iii) Woman w in P_3 strictly prefers m to m' in her original preference list.

(iv) Man m in P_3 is indifferent between w and w' in his original preference list.

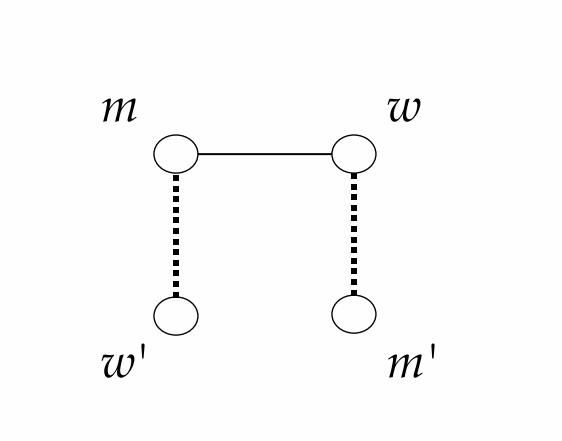

Figure 3.4: A P_3 in $M \oplus M_{opt}$. Dashed edges belong to M_{opt}, the undashed edge to M.

(v) Man m' proposed to woman w prior to the end of the final execution of phase 1 and was rejected by her. Hence, w is matched prior to any potential call to phase 3.

Proof (i) This follows from Lemma 3.4.3 and the fact that m' is unmatched in M. (ii) Since m has an unmatched woman w' on his preference list, he could never have become exhausted, for w' cannot have received a proposal. (iii) If w strictly prefers m' to m, then (m', w) is a blocking pair for M, a contradiction. If, instead, she were indifferent between these two men, she could not have rejected m', who, by Lemma 3.4.3, must have proposed to w at some point after being promoted. But at that moment w was matched to m or someone ranked lower, and m was never promoted. (iv) If m strictly prefers w' to w, then (m, w') blocks M. But by (iii), m forms a blocking pair with w in M_{opt} if he strictly prefers w to w'. (v) Every man who participates in phase 3 becomes matched, hence m' did not participate in phase 3, and since no matched man becomes unmatched during phase 3 or phase 2, man m' was unmatched at the end of the final call to phase 1. By Lemma 3.4.3, m' proposed to w even after becoming promoted, but since he is single in M, she must have rejected him. Hence, w is matched to someone ranked at least as highly as m at the final call to phase 1. □

Now, we arrive at the contradiction. Consider again the path P_3 as shown in Figure 3.4. Since matched women never become unmatched, man m always had woman w' unmatched on his preference list, and by Lemma 3.5.1 (iv) she is tied with w. In what phase of the algorithm can m have become matched to w? It cannot have been in phase 1, for the phase 1 algorithm does not allow him to propose to w, regardless of whether or not w is matched, because of w' being tied with w

$m_1:$	$w_2 \quad w_3$	$w_1:$	$m_2 \quad m_3$
$m_2:$	(w_1, w_2)	$w_2:$	(m_1, m_2)
$m_3:$	w_1	$w_3:$	m_1

Figure 3.5: An instance of SMTI that yields a performance ratio of 3/2.

and unmatched. But m cannot become matched to w in some call to phase 2 either, for the fact that w' is unmatched at the end of the algorithm implies she could never be in N' at any call to phase 2. By Lemma 3.4.2, this implies that the pair (m, w) would have to be deleted in the creation of N' as well. Thus, we conclude that m became matched to w in phase 3. However, men only become matched to unmatched women in phase 3, implying that w is single at the start of phase 3, a contradiction of Lemma 3.5.1 (v).

We are forced to conclude that P_3 cannot exist in $M \oplus M_{opt}$.

Theorem 3.5.1 *The polynomial-time approximation algorithm outputs a stable matching at least* $\frac{2}{3}$ *the size of an optimal stable matching.*

3.6 Tightness of the performance guarantee

We give an example to show that this is the tightest bound possible for our approximation algorithm. Yanagisawa [104] first observed that $3/2$ was the tightest possible bound, later, Chebolu [16] gave a different example to establish the tightness of the performance guarantee. We have chosen to present the example of Chebolu because the particularly symmetric nature of the preference lists make it very easy to understand. Consider the SMTI instance given in Figure 3.5; note that we use parentheses to denote a tie in a preference list. One possible execution of the algorithm begins by m_1 proposing to w_2, followed by a proposal from m_2 to w_1. Man m_3 could then propose to w_1, and, although he will become promoted, and propose again, he will still remain unmatched. The algorithm stops with a matching containing the pairs (m_1, w_2) and (m_2, w_1). An optimal solution, however, is the perfect stable matching given by the pairs (m_1, w_3),(m_2, w_2), and (m_3, w_1). Notice that because of the symmetry of the instance, this example still applies if the roles of the men and women are reversed.

3.7 Conclusion and open questions

We have presented a polynomial-time approximation algorithm for general MAX-SMTI with a performance guarantee of $3/2$, improving the previously best known algorithm for this problem. An obvious open problem is to find a further improved approximation algorithm or to further tighten the inapproximability bound for MAX-SMTI.

Finally, we note that our approximation algorithm also extends to the Hospitals/Residents with ties setting (HRT), by a technique involving "cloning" [48], with the same performance guarantee of $3/2$.

Chapter 4

Sex-equal stable matchings

4.1 Introduction

How can we find stable matchings that are somehow fair to both the men and the women of an SMI instance? Of course, this begs the question, what does it mean to treat the men and the women fairly? A natural definition of fairness could arise from the following intuition. Suppose we could somehow quantify the overall "happiness" of the men, and the overall "happiness" of the women. Then, a stable matching could be considered fair if the happiness of the men is equal, or as close as possible, to the happiness of the women.

This is precisely the goal of the sex-equal stable marriage problem (SESM) we discussed in Section 2.2.4. Let us recall the definition of this problem. Let I be an arbitrary SMI instance, and let n denote the number of agents of I, i.e., the number of men plus the number of women. For agents a and b, let $p_a(b)$ denote the *position* of agent b on agent a's preference list.

Define the *sex-equality measure* $\delta(\cdot)$ for a stable matching $M \in \mathcal{M}$ as follows.

$$\delta(M) = \sum_{(m,w)\in M} p_m(w) - \sum_{(m,w)\in M} p_w(m).$$

The goal of the *sex-equal stable marriage problem* (SESM) is to find a stable matching $M \in \mathcal{M}$ that minimizes $|\delta(M)|$, where $\mathcal{M}$ is the set of all stable matchings. SESM is NP-hard [58], and

the only positive results known for this problem are due to Iwama et al [56], as reviewed in Section 2.2.4.

This chapter explores SESM for SMI instances in which the lengths of the preference lists of the men and/or women are bounded in length by a constant. We use the notation (α, β)-SESM to denote the problem of finding a sex-equal stable matching of an SMI instance in which the men's (women's) preference lists have length at most α (β). We use ∞ for the case when α or β can be arbitrarily large, so, for example, (l, ∞)-SESM means the men's lists are bounded by l but the women's lists can be arbitrarily long.

This chapter specifically explores (α, β)-SESM from the viewpoint of exact exponential-time algorithms and parameterized (FPT) complexity (a review of FPT and parameterized complexity is given in Section 4.4). There has been much recent interest in exact exponential-time algorithms for computationally hard problems. We refer the reader to the surveys of Woeginger [101, 102].

Our results are summarized as follows. On the negative side, we show that $(3, 3)$-SESM is $W[1]$-hard. This strengthens the NP-hardness results of Kato [58]. Furthermore, we show that our hardness result is "tight" by giving a polynomial-time dynamic programming algorithm for $(2, \infty)$-SESM and $(\infty, 2)$-SESM. On the positive side, we give a low-order exponential-time algorithm for (l, ∞)-SESM. To be precise, we give an algorithm with running time[1] bounded by $O^\star(1.0725^n)$, $O^\star(1.1503^n)$, $O^\star(1.2338^n)$, ... for $l = 3, 4, 5, \ldots$. By reversing the roles of the men and the women, this algorithm applies to (∞, l)-SESM as well.

Our algorithm is built on a number of new observations regarding the rotation poset and the rotation digraph (Hasse diagram) of an (l, ∞)-SESM instance (see Section 2.2.3 for a review of the structural results for SMI). We show that, in a formal sense, when the number of rotations in the rotation poset Π is at most a certain threshold, then a brute-force algorithm that enumerates all closed subsets of the rotations of Π suffices to find a SESM. Otherwise, if the number of rotations exceeds this threshold, then we show that the rotation digraph D_Π must be *sparse*. We then use existing results concerning sparse graphs to design an exponential-time algorithm with a running time as described above.

We reviewed the structural results for SMI in Section 2.2.3, and we shall rely on these results quite heavily in this chapter. In the next section, we review these results and also cover some of the finer

[1] We use the standard $O^\star$ notation that suppresses polynomial factors in any terms to analyze the running time of an exponential-time algorithm.

structural details omitted from Section 2.2.3 that we specifically require only in this chapter.

4.2 Further structural results for SMI

4.2.1 The number of men

As we discussed in Section 2.2.2, Gale and Sotomayor [31] showed all stable matchings of an SMI instance match exactly the same subset of the agents. Hence, we may assume without loss of generality that those agents who are never matched in a stable matching are discarded from the instance. These agents can never affect the sex-equality measure of a stable matching, and thus can be ignored. A consequence of this is that the number of remaining men must equal the number of remaining women. Henceforth we let n denote the number of men plus the number of women of this remaining instance in which all unmatched agents have already been discarded.

4.2.2 Rotations, rotation posets, and SESM

The rotation poset Π

For an arbitrary SMI instance I, we let M_0 and M_z denote the man- and woman-optimal stable matchings of I, respectively, and $\Pi = (R, \preceq)$ the rotation poset of I. For a subset of rotations $R' \subseteq R$, we denote by $\Pi[R']$ the partially ordered set induced by R'. The canonical reference for the following details regarding rotations is the monograph of Gusfield and Irving [36, Chapter 3].

Let $\rho = ((m_0, w_0), \ldots, (m_{r-1}, w_{r-1}))$ be a rotation. We say that ρ *moves* m_i *down* from w_i to w_{i+1} and *moves* w_i *up* from m_i to m_{i-1}. If w is either w_i or is strictly between w_i and w_{i+1} in m_i's list, then ρ *moves* m_i *below* w. Similarly, ρ *moves* w_i *above* m_i if m is m_i or is strictly between m_i and m_{i-1} in w_i's list.

Fact 4.2.1 *(Gusfield and Irving [36]) Let* Π *be the rotation poset of an arbitrary SMI instance. Then,*

1. *For any man* m *and woman* w*, there is at most one rotation that moves* m *down to* w*, and* w *up to* m*. Furthermore, there is at most one rotation that moves* m *from* w*.*

2. *For any man m and woman w, there is at most one rotation that moves w to a man strictly above m in w's preference list.*

The rotation digraph D_Π and the underlying graph G_Π

We let D_Π denote the rotation digraph of Π (recall that this is the directed Hasse diagram of Π). There is a key characterisation of the arcs of D_Π that is given by the fact below.

Fact 4.2.2 *(Gusfield and Irving [36]) Let D_Π denote the rotation digraph of an arbitrary SMI instance.*

1. *If $(m, w) \in \rho$, and ρ' is the (unique) rotation that moves m to w, then (ρ', ρ) is a directed edge in D_Π. In this case, ρ' is called a type-1 predecessor of ρ.*
2. *If ρ moves m below w, and $\rho' \neq \rho$ is the (unique) rotation that moves w above m, then (ρ', ρ) is a directed edge in D_Π. In this case, ρ' is called a type-2 predecessor of ρ.*
3. *Every arc $(\rho', \rho) \in D_\Pi$ satisfies either (1) or (2) (or both) for some m, w.*

When referring to D_Π we will sometimes find it useful to consider the arcs of D_Π as being undirected. So, we let G_Π denote the undirected graph obtained by replacing every directed arc of D_Π with an undirected edge. We also take a moment to remark that we refer to the rotations of Π and the vertices of G_Π and D_Π as both rotations and vertices interchangeably. The meaning should always be clear from the context.

Weighted rotations and weighted subsets

Recall the goal of SESM is to find a stable matching M_S minimizing the absolute value of

$$\delta(M) = \sum_{(m,w)\in M} p_m(w) - \sum_{(m,w)\in M} p_w(m).$$

We sometimes use the δ notation for a closed subset of rotations S, so that $\delta(S)$ provides a shorthand for $\delta(M_S)$, where M_S is the stable matching obtained by eliminating the rotations in S.

For a rotation $\rho = (m_0, w_0), (m_1, w_1), \ldots, (m_{r-1}, w_{r-1})$, Iwama et al [56] define the following weight $w(\rho)$, which captures the change in sex-equality measure resulting from the elimination of ρ:

$$w(\rho) = \sum_{i=0}^{r-1}(p_{m_i}(w_{i+1}) - p_{m_i}(w_i)) - \sum_{i=0}^{r-1}(p_{w_i}(m_{i-1}) - p_{w_i}(m_i))$$

For a set of rotations R', we let $w(R')$ denote the sum of the weights of the rotations in R'. An understanding of the following facts is necessary for the methods used in the forthcoming sections.

Fact 4.2.3 *(Iwama et al [56]) Let I be an arbitrary SMI instance. Then*

1. $w(\rho) > 0 \; \forall \rho \in R$.
2. $\delta(M/\rho) = \delta(M) + w(\rho)$ *for any stable matching M and rotation ρ exposed in M.*
3. *For a closed subset R',* $\delta(R') = \delta(M_0) + \sum_{\rho \in R'} w(\rho) = \delta(M_0) + w(R')$.

Notice that in light of Fact 4.2.3 (1), if $\delta(M_0) > 0$, then M_0 must necessarily be the unique sex-equal stable matching, as the elimination of any rotations will only worsen the sex-equality measure of the stable matching. We also briefly remark on the important difference between $w(S)$ and $\delta(S)$. The notation $w(S)$ refers to the sum of the weights of a (not necessarily closed) set of rotations while $\delta(S)$ is the sex-equality measure of the stable matching obtained by eliminating a (closed) subset S.

4.3 Series-parallel graphs

Our exact algorithm in Section 4.8 relies heavily on the properties of so-called *series-parallel graphs*. In this section we briefly review the necessary definitions and properties of series-parallel graphs.

A *two-terminal labelled graph* (G, s, t) consists of an undirected graph G with two distinct marked vertices $s, t \in V$, where s is called the *source* and t is called the *sink*. The *series composition* of two-terminal labelled graphs (G_1, s_1, t_1) and (G_2, s_2, t_2), where s_1 and s_2 (t_1 and t_2) are the sources

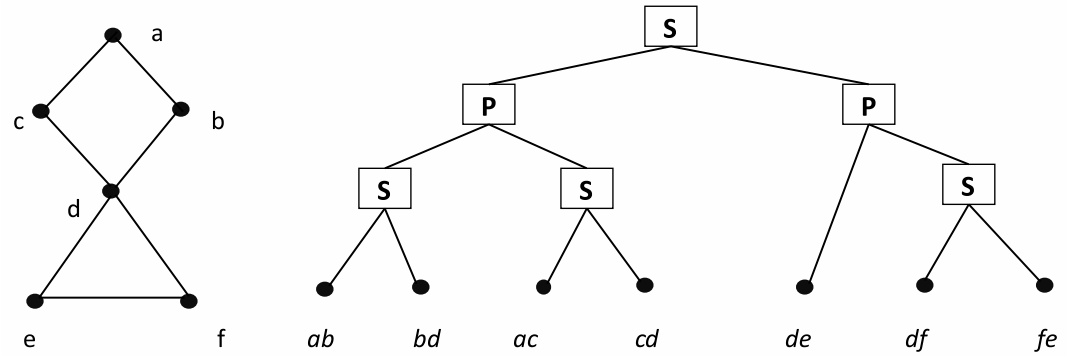

Figure 4.1: A series-parallel graph and a corresponding SP tree

(sinks) of G_1 and G_2, respectively, is the two-terminal labelled graph obtained by identifying t_1 with s_2. The *parallel composition* of two-terminal labelled graphs (G_1, s_1, t_1) and (G_2, s_2, t_2) is the two-terminal labelled graph obtained by identifying s_1 with s_2 and t_1 with t_2. A graph is a *series-parallel graph* if and only if it can be created from single two-terminal edges by a sequence of series and/or parallel compositions.

An interesting side-effect of the definition of series-parallel graphs is that the way in which the series-parallel graph is constructed implicitly describes a binary tree, called an *SP tree*. The leaves of the SP tree $\mathcal{T}$ are the edges of G, and every internal node of G is labelled either S or P to denote whether a series or parallel operation was used to join the two series-parallel graphs described by its children. See Figure 4.1 for an example of a series-parallel graph and a corresponding SP tree.

4.4 Parameterized problems, FPT, and W[1]-hardness

Before presenting the primary $W[1]$-hardness result of this chapter, we give a very basic review of the necessary definitions and background of parameterized problems and parameterized complexity. We refer the reader to the texts of Neidermeier [82] and Downey and Fellows [22] for a more thorough treatment. We begin our basic tutorial with the formal definition of a parameterized problem. The key point of interest in the definition of a parameterized problem is that it is a decision problem that asks for a solution of size *exactly* k, as opposed to saying, for example, *at most* k or *at least* k. The formal definition is given as follows.

Definition 4.4.1 *(Neidermeier [82]) A parameterized problem L over an alphabet Σ is a set of pairs (x, k) with $x \in \Sigma^*$ and k a non-negative integer such that there is no x with $(x, k) \in L$ and $(x, k') \in L$ for some $k' \neq k$.*

By way of example of a parameterized problem, consider the parameterized versions of Clique and (l, ∞)-SESM given below.

Clique
Input: A graph $G = (V, E)$ and a non-negative integer k.
Question: Does G contain a complete subgraph $C \subseteq V$ with exactly k vertices?

(l, ∞)**-SESM**
Input: An (l, ∞)-SMI instance I and a non-negative integer k.
Question: Is there a stable matching M for I such that $\delta(M)$ is exactly k?

We continue our brief tutorial with the definition of fixed-parameter tractability.

Definition 4.4.2 *(Neidermeier [82]) A parameterized problem L over an alphabet Σ is fixed-parameter tractable if it can be determined in $f(k)n^{O(1)}$ time whether or not $(x, k) \in L$, where f is a computable function depending only on k. The corresponding complexity class is called FPT.*

The class FPT contains the complexity class P. In addition, many NP-hard problems are known to lie within FPT, including, for example, the well-known minimum vertex cover problem. However, this is not true of all NP-hard problems.

As is commonly known, proving that a combinatorial problem is NP-complete provides a proof that the problem cannot be solved in polynomial-time unless P = NP. In what follows we explain the notion of so-called W[1]-hardness, an idea analogous to NP-hardness, which provides the theoretical background for formally establishing some notion of fixed-parameter intractability of a problem. We first define the concept of a parameterized reduction, followed by the definition of the complexity class W[1].

Definition 4.4.3 *(Neidermeier [82]) Let $L, L' \subseteq \Sigma^* \times \mathcal{N}$, where $\mathcal{N}$ denotes the positive integers, be two parameterized problems. We say that L reduces to L' by a standard parameterized reduction if there are functions $f(k) \to k'$ and $g(k) \to k''$ from $\mathcal{N}$ to $\mathcal{N}$ and a function $h(x,k) \to x'$ from $\Sigma^* \times \mathcal{N}$ to Σ^* such that:*

1. *$h(x,k) \to x'$ is computable in $k''|(x,k)|^c$ time for a constant c and*
2. *$h(x,k) \in L$ if and only if $(x',k') \in L'$.*

Consider the following examples that distinguish the difference between a 'classical' polynomial time reduction and a parameterized reduction. It is easily seen that a graph G has a vertex cover of size k if and only if it has an independent set of size $n-k$. Therefore, vertex cover reduces to independent set in polynomial time. This is not a parameterized reduction, because the derived instance of independent set has the parameter value $n-k$, which does not exclusively depend on k but also on n. On the other hand, it is easy to see that a graph G has an independent set of size k if and only if its complement graph G' has a clique of size k. This constitutes a parameterized reduction.

The Weighted 2-CNF-Satisfiability problem, which plays a key role in defining the complexity class $W[1]$ (given in Definition 4.4.4 below) is defined as follows.

Weighted 2-CNF-Satisfiability
Input: A boolean formula $\mathcal{F}$ in conjunctive normal form, in which every clause of $\mathcal{F}$ has at most two literals, and a nonnegative integer k.
Question: Is there a satisfying truth assignment for $\mathcal{F}$ that has exactly k variables set to true?

Now we may present the definition of the class $W[1]$.

Definition 4.4.4 *(Neidermeier [82])*

1. *The class $W[1]$ contains all problems that can be reduced to Weighted 2-CNF-Satisfiability by a parameterized reduction.*
2. *A parameterized problem is said to be W[1]-hard if the parameterized problem Weighted 2-CNF-Satisfiability can be reduced to it by a parameterized reduction.*

3. A problem in $W[1]$ that is $W[1]$-hard is said to be W[1]-complete.

It is known that FPT $\subseteq W[1]$, and, if any $W[1]$-complete problem were shown to be in FPT, the result would imply a collapse of complexity classes FPT and $W[1]$, which is thought to be unlikely. Quite a few well-studied problems are known to be $W[1]$-hard, including dominating set and independent set. For our purposes, we need only the following result, which concludes this section.

Theorem 4.4.1 *(Neidermeier [82, Corollary 13.5]) Clique is $W[1]$-hard.*

4.5 $(3, 3)$-SESM is $W[1]$-hard

We next describe a parameterized reduction from the $W[1]$-hard problem Clique to (l, ∞)-SESM. In fact, our reduction will prove W[1]-hardness even for the special case of $(3, 3)$-SESM. The reduction is inspired by a construction of Johnson and Niemi [57] who reduce an instance of Clique to an instance of the partially ordered knapsack problem, defined below.

Partially ordered knapsack
Input: Directed acyclic graph $G = (V, A)$, a weight $w(v) \in \mathcal{Z}^+$ and a value $p(v) \in \mathcal{Z}^+$ for each vertex $v \in V$, a knapsack capacity $B \in \mathcal{Z}^+$, and a bound $C \in \mathcal{Z}^+$.
Question: Is there a subset $V' \subseteq V$, closed under predecessor, such that $w(V') \leq B$ and $p(V') \geq C$?

To make the description of our transformation more easily understood, we review the construction of Johnson and Neimi. Given an instance $I = (G = (V, E), K)$ of the Clique problem, they create an instance $I' = (G' = (V', A'), B', C')$ of the partially ordered knapsack problem as follows.

$$
\begin{aligned}
&V' = V \cup E, \\
&A' = \{(v, e) : v \in V, e \in E, v \text{ is an endpoint of } e\} \\
&w(v) = p(v) = |E| + 1 && \text{for all } v \in V, \\
&w(e) = p(e) = 1, && \text{for all } e \in E, \\
&B' = C' = K(|E| + 1) + \tbinom{K}{2}.
\end{aligned}
$$

Hence G' is a bipartite acyclic graph in which each arc is directed from an element of V to an element of E. Each element of E necessarily has exactly two predecessors, and each element of V has the same number of successors in G' as it has edges incident to it in G. Suppose now that G has a clique (V_K, E_K) of size K. Then the set of vertices $V_K \cup E_K$ is a closed subset of G' of weight and value $K(|E| + 1) + \binom{K}{2}$. Suppose instead that G' has a closed subset S' of weight and value $K(|E| + 1) + \binom{K}{2}$. Notice that the choice of weights and values for I' are such that each element of V weighs more than the sum of all elements of E, which have weight and value one. A closed subset of G' that has exactly a weight and value of $K(|E| + 1) + \binom{K}{2}$ must consist of K vertices from V and $\binom{K}{2}$ vertices from E. Since S' is closed, S' corresponds to K vertices from V with $\binom{K}{2}$ edges between them in G, i.e. a clique of size K in G.

4.5.1 Reduction idea

Our reduction to SESM will use the transformation of Johnson and Niemi in the following way. The idea is to reduce an instance I of Clique to an instance I' of SESM such that the rotation poset of I' has precisely the same structure as that constructed for the derived partially ordered knapsack instance above. Our reduction will map every $v \in V$ to a rotation v with weight $w(v) = 8|E| + 2$, and every edge $e = \{v_i, v_j\}$ of E to a rotation e with predecessors v_i and v_j and $w(e) = 8$. We will construct our derived instance in such a way that the man-optimal stable matching M_0 for I' will have the property that $\delta(M_0) = -[K(8|E| + 2) + 8\binom{K}{2}]$. Hence a closed subset of weight exactly $K(8|E| + 2) + 8\binom{K}{2}$ corresponds to a stable matching M_S having $\delta(M_S) = 0$. Since the rotation poset of our derived instance will have the same structure as that of the reduction of Johnson and Niemi, such a closed subset must correspond to a clique of size exactly K in G. We next describe this reduction formally.

4.5.2 The parameterized reduction

Step 1: the vertex gadget

For each vertex $v_i \in V$, we create $4|E| + 1$ men $\{m_i^0, m_i^1, \ldots, m_i^{4|E|}\}$ and $4|E| + 1$ women $\{w_i^0, w_i^1, \ldots, w_i^{4|E|}\}$. Each of these men will have at most three entries on his preference lists while each of these women will have exactly three entries on her preference list. However, in this step, we only

define two entries for each man and woman. The first two entries on the preference list of a man m_i^j are w_i^j and w_i^{j+1}, respectively, where $j+1$ is taken modulo $4|E|+1$. The second and third man on a woman w_i^j's preference list are m_i^{j-1} and m_i^j, respectively, where $j-1$ is taken modulo $4|E|+1$. The third entry of a man m_i^j and the first entry of a woman w_i^j will be defined below at a later step. The preference lists created by this step are described below; an underlined star in a preference list indicates an entry that has not yet been created.

$$\begin{array}{rllll} m_i^0: & w_i^0 & w_i^1 & \underline{*} \\ m_i^1: & w_i^1 & w_i^2 & \underline{*} \\ \vdots & & & \\ m_i^{4|E|}: & w_i^{4|E|} & w_i^0 & \underline{*} \end{array} \qquad \begin{array}{rllll} w_i^0: & \underline{*} & m_i^{4|E|} & m_i^0 \\ w_i^1: & \underline{*} & m_i^0 & m_i^1 \\ \vdots & & & \\ w_i^{4|E|}: & \underline{*} & m_i^{4|E|-1} & w_i^{4|E|} \end{array}$$

Step 2: the edge gadget

For each edge $e = \{v_r, v_s\} \in E$, we create two men $\{m_{r,s}^1, m_{r,s}^2\}$ and two women $\{w_{r,s}^1, w_{r,s}^2\}$. These men and women will each have two agents on their preference lists. The preference lists for these agents are shown below, where again the blanks denote entries not yet specified.

$$\begin{array}{rlll} m_{r,s}^1: & w_{r,s}^1 & \underline{*} \\ m_{r,s}^2: & w_{r,s}^2 & \underline{*} \end{array} \qquad \begin{array}{rlll} w_{r,s}^1: & \underline{*} & m_{r,s}^1 \\ w_{r,s}^2: & \underline{*} & m_{r,s}^2 \end{array}$$

Step 3: complete the preference lists

For each edge $e = \{v_r, v_s\} \in E$, with $r < s$, we choose two men created in correspondence to vertices v_r and v_s by selecting the first man m_r^p (respectively, m_s^q) from the sorted list $m_r^1, m_r^2, \ldots, m_r^{4|E|}$ (respectively, m_s^1, m_s^2,...,$m_s^{4|E|}$) whose third choice has not yet been specified. We complete the preference lists of agents m_r^p, m_s^q, w_r^{p+1}, w_s^{q+1}, $m_{r,s}^1$, $m_{r,s}^2$, $w_{r,s}^1$, and $w_{r,s}^2$ as described in the figure below. The underlining is in place to illustrate which entries are completed by this step.

$$\begin{array}{rllll} m_r^p: & w_r^p & w_r^{p+1} & \underline{w_{r,s}^1} \\ m_s^q: & w_s^q & w_s^{q+1} & \underline{w_{r,s}^2} \\ m_{r,s}^1: & w_{r,s}^1 & \underline{w_r^{q+1}} \\ m_{r,s}^2: & w_{r,s}^2 & \underline{w_s^{p+1}} \end{array} \qquad \begin{array}{rllll} w_r^{p+1}: & \underline{m_{r,s}^2} & m_r^{p-1} & m_r^p \\ w_s^{q+1}: & \underline{m_{r,s}^1} & m_r^{q-1} & m_r^q \\ w_{r,s}^1: & \underline{m_r^p} & m_{r,s}^1 \\ w_{r,s}^2: & \underline{m_s^q} & m_{r,s}^2 \end{array}$$

After the above step has been performed for every edge, every agent created in step 2 has had their preference list completed. However, there will still be a set of women w_i^j created in step 1 who still have an unspecified first choice. For each of these women, we create a dummy man and a dummy woman who rank each other first, and place w_i^j last on the dummy man's list and place the dummy man first on w_i^j's list. Note there will also be a set of men created in step 1 with an unspecified third choice on their preference lists; these men only require a total of two women on their lists. This completes the construction of the agents created in steps 1 and 2 and their preference lists.

Step 4: pad the instance

The final step of the reduction is to pad the instance to appropriately 'offset' $\delta(M_0)$, where M_0 is the man-optimal stable matching of the derived instance. To this end, let $t = 8|V||E| + 2|V| + 2|E| - [K(8|E| + 2) + 8\binom{K}{2}]$ (this expression is intentionally left unsimplified). We create $2t$ men $\{x_0^1, x_1^1, \ldots, x_0^t, x_1^t\}$ and $2t$ women $\{y_0^1, y_1^1, \ldots, y_0^t, y_1^t\}$. The preference lists of men x_0^i, x_1^i, y_0^i, and y_1^i for $(1 \leq i \leq t)$ are shown below.

$$\begin{array}{llll} x_0^i: & y_1^i & y_0^i & \\ x_1^i: & y_1^i & & \end{array} \qquad \begin{array}{lll} y_0^i: & x_0^i & \\ y_1^i: & x_1^i & x_0^i \end{array}$$

The final step of the reduction maps the parameter K to $K' = 0$. Thus we have reduced an instance I of Clique to an instance I' of SESM. We now prove that I has a clique of size exactly K if and only if I' has a stable matching M_S with $\delta(M_S) = K' = 0$. Our first concern are the properties of the man-optimal stable matching of I'. The first lemma follows immediately from the reduction and requires no proof.

Lemma 4.5.1 *The man-optimal stable matching M_0 for the derived instance I' of SESM matches every man created in step 1 and step 2 to his first choice. Equivalently, every woman created in step 1 and step 2 is matched to her last choice in M_0.*

Lemma 4.5.2 *Let M_0 denote the man-optimal stable matching for the derived instance I'. Then, $\delta(M_0) = -[K(8|E| + 2) + 8\binom{K}{2}]$.*

Proof As stated in Lemma 1, M_0 matches every man created in step 1 and step 2 to his first choice and every woman created in step 1 and step 2 to her last choice. Hence the difference in happiness

of these men and women is $|V|(4|E|+1)+2|E|-[|V|(12|E|+3)+4|E|]$, which simplifies to $-8|V||E|-2|V|-2|E|$. Every dummy man created in step three is matched to his first choice and his partner in M_0 is matched to her first choice, so these agents contribute the same sum to the men's and women's happiness, respectively, and can be ignored. What remains are the agents created in step 4. For each i $(1 \leq i \leq 2t)$, the pairs (x_0^i, y_0^i) and (x_1^i, y_1^i) must always be matched together in any stable matching. Therefore, in any stable matching M' for I', each such group of four agents contributes a sum of one to $\delta(M')$. Since there are t such groups of four, the difference in the men's and women's happiness amongst those agents created in step 4 is $t = 8|V||E| + 2|V| + 2|E| - [K(8|E|+2) + 8\binom{K}{2}]$. Therefore $\delta(M_0) = -[K(8|E|+2)+8\binom{K}{2}]$. □

Corollary 4.5.1 *I' has a stable matching M with $\delta(M) = K' = 0$ if and only if there is a closed subset of the rotation poset of I' weight exactly $-[K(8|E|+2)+8\binom{K}{2}]$*

The next three lemmas establish the structure and nature of the rotations and rotation poset of the derived instance of SESM.

Lemma 4.5.3 *For each vertex $v_i \in I$, there exists a rotation $\rho_i = (m_i^0, w_i^0)$, (m_i^1, w_i^1), $\ldots$ $(m_i^{4|E|}, w_i^{4|E|})$ exposed in M_0 with weight $8|E|+2$.*

Proof Since M_0 matches every man to his first choice, it is easy to verify that the successor woman of any man m_i^j in M_0 is w_i^{j+1}, where $j+1$ is taken modulo $4|E|+1$, implying ρ_i is indeed exposed in M_0. The elimination of ρ_i moves every man down one place to his second choice, decreasing the sum of the positions of the men's partners by $4|E|+1$, and moves every woman up one place to her second choice, increasing the sum of the positions of the women's partners by $4|E|+1$. Hence ρ_i has weight $8|E|+2$. □

Lemma 4.5.4 *Let $\{v_r, v_s\} \in E$ be an edge in I where $r < s$. Then, the elimination of both $\rho_r = (m_r^0, w_r^0)$, (m_r^1, w_r^1), $\ldots$ $(m_r^{4|E|}, w_r^{4|E|})$ and $\rho_s = (m_s^0, w_s^0)$, (m_s^1, w_s^1), $\ldots$ $(m_s^{4|E|}, w_s^{4|E|})$ exposes a rotation $\sigma_{r,s} = (m_{r,s}^1, w_{r,s}^1)$, (m_r^q, w_r^{q+1}), $(m_{r,s}^2, w_{r,s}^2)$, (m_s^p, w_s^{p+1}) for some $p, q \in \{0, 1, \ldots, 4|E|\}$ with weight 8.*

Proof Suppose (v_r, v_s) with $r < s$ is an edge of I. In step 3 of the reduction, two men, say m_r^p and m_s^q, whose third choice had not yet been defined, were selected and the preference lists of m_r^p, m_s^q,

w_s^{q+1}, w_r^{p+1}, $m_{r,s}^1$, $m_{r,s}^2$, $w_{r,s}^1$, and $w_{r,s}^2$ were completed. After the elimination of rotations ρ_r and ρ_s, the men m_r^p and m_s^q are matched to the women w_r^{p+1} and w_s^{q+1}, respectively. Therefore after the elimination of both of these rotations the successor women of men m_r^p and m_s^q are $w_{r,s}^1$ and $w_{r,s}^2$, respectively. Furthermore, the successor women of $m_{r,s}^1$ and $m_{r,s}^2$ are w_r^{p+1} and w_s^{q+1}, respectively. It follows that $\sigma_{r,s} = (m_{r,s}^1, w_{r,s}^1)$, (m_r^q, w_r^{q+1}), $(m_{r,s}^2, w_{r,s}^2)$, (m_s^p, w_s^{p+1}) is a rotation whose set of predecessors is precisely $\{\rho_r, \rho_s\}$. The elimination of $\sigma_{r,s}$ moves every man down one place on his list, and every woman up one place on her list, hence the weight of $\sigma_{r,s}$ is 8. □

Lemma 4.5.5 *The rotation poset for I' contains exactly one rotation ρ_i for every $v_i \in I$, and one rotation $\sigma_{r,s}$ for every edge $\{v_r, v_s\}$ such that $r < s$ in I. The predecessors of $\sigma_{r,s}$ are exactly ρ_r and ρ_s, and the rotations ρ_i have no predecessors.*

Proof By the previous lemmas, it is clear that the rotation poset for I' contains ρ_i for every $v_i \in I$, and $\sigma_{r,s}$ for every edge $\{v_r, v_s\}$, such that $r < s$, with the predecessors of $\sigma_{r,s}$ being exactly $\{\rho_r, \rho_s\}$. To see that these are precisely the rotations of the rotation poset of the derived instance, notice that the elimination of all rotations ρ_i and $\sigma_{r,s}$ assigns every man created in step 1 and step 2 to his last choice. Every dummy agent created in step 3 must always be matched to his/her first choice in any stable matching, and the same is true of the $2t$ men created in step 4. Hence no other rotations can exist. □

Lemma 4.5.6 *The given instance I has a clique of size exactly K if and only if I' has a stable matching M_S with $\delta(M_S) = K' = 0$.*

Proof

The first direction of the proof is almost immediate. Let (V_K, E_K) be a clique of size exactly K in I. Then, the rotations created in correspondence to (V_K, E_K) from a closed subset of the rotation poset of I' with weight precisely $K(8|E| + 2) + 8\binom{K}{2}$, which, by Corollary 4.5.1 must correspond to a stable matching of cost exactly $K' = 0$.

Now suppose that I' has a SESM of cost exactly $K' = 0$. Then, again by Corollary 4.5.1, the closed subset of rotations S eliminated to obtain such a stable matching has cost exactly $K(8|E| + 2) + 8\binom{K}{2}$. Since the rotation poset of I' was constructed in a correspondence to the reduction

of Johnson and Niemi [57], S must contain K rotations ρ_i and $\binom{K}{2}$ rotations $\sigma_{r,s}$, but as in the construction of Johnson and Niemi, such a choice must correspond to a clique of size K in G. □

Theorem 4.5.1 *The sex-equal stable matching problem is $W[1]$-hard, even if both the men's and women's preference lists are of length at most three.*

Proof It is clear that the men and the women have preference lists of length at most 3. Let us be thorough in verifying the reduction satisfies the requirements of a many-one parameterized reduction. Clearly, for a given instance (x, k) of Clique, our derived instance x' can be computed easily in $|(x, k)|^3$ time, without any attempt at optimization. Our mappings from the parameter k are therefore such that $k \to k'' = 1$ and $k \to k' = 0$. Finally, we have established in Lemma 4.5.6 that (x, k) is a 'yes' instance if and only if (x', k') is as well. □

4.6 Inapproximability results for SESM

Recall that, for an arbitrary SESM instance, it could be that an optimal solution M_{opt} has $|\delta(M_{opt}| = 0$. In order to reason about approximability results for this problem, let us define a new optimality measure $f(\cdot)$ of a stable matching M to be $f(M) = |\delta(M)| + 1$. Hence the value of $f(\cdot)$ is always greater than zero.

Since the $W[1]$-hardness of a problem implies NP-hardness as well, a corollary of Lemma 4.5.6 and Theorem 4.5.1 is that it is NP-complete to decide whether a stable matching instance admits a stable matching M such that $f(M) = 1$. The following theorem shows that there is no polynomial-time approximation algorithm with a performance guarantee (measured according to $f(\cdot)$) less than two.

Theorem 4.6.1 *The sex-equal stable matching problem is NP-hard to approximate (relative to the measure $f(\cdot)$) within a factor less than two, even if the men's and women's preference lists are of length at most three.*

Proof

For a contradiction, let I be an arbitrary stable matching instance, and A a c-approximation algorithm with $c < 2$ for SESM. Consider a stable matching M returned by an execution of algorithm

A on the instance I. If I admits a stable matching M' with $f(M') = 1$, then M satisfies $f(M) < 2$, i.e. $f(M') = 1$, since $f(M')$ must be integral. On the other hand, if I does not admit a stable matching M with $f(M) = 1$, then M' must satisfy $f(M') >= 2$.

Hence, A decides an NP-complete problem in polynomial-time, a contradiction, unless P = NP. □

4.7 Polynomial-time algorithm for $(2, \infty)$-SESM

The polynomial-time solvability of $(2, \infty)$-SESM follows almost immediately from the following lemma.

Lemma 4.7.1 *Let I be a $(2, \infty)$-SESM instance, and $\Pi = (R, \preceq)$ its rotation poset. Then, the relation $\preceq$ is the empty set. In other words, D_Π has no edges.*

Proof Suppose for a contradiction that there are rotations ρ', $\rho \in R$ with $\rho' \prec \rho$. By Fact 4.2.2, ρ' is either a type-1 or a type-2 predecessor of ρ (or both). Suppose that ρ' is a type-1 predecessor of ρ. By definition, there exists $(m, w) \in \rho$, such that ρ' is the unique rotation that moves m to w. This implies that w is m's second (and therefore last) choice on his preference list. Thus ρ does not exist.

Suppose instead that ρ' is a type-2 predecessor. By definition there is (i) a man m and a woman w such that ρ moves m below w, and (ii) ρ' moves w above m. Since m's preference list has length at most two, (i) forces us to conclude that $(m, w) \in \rho$ and w is m's first choice. But (ii) forces us to conclude that ρ' matches m to w. This cannot be the case if w is m's first choice. □

Now we describe the algorithm. Let $R = \rho_1, \ldots, \rho_k$ be the set of all rotations for I, all of which must be exposed in the man-optimal stable matching by Lemma 4.7.1. Let $W = w(\rho_1) + w(\rho_2) + \ldots + w(\rho_k)$. Using the standard dynamic programming algorithm for the subset-sum problem, we can determine if Π has a (closed) subset of weight k, for each $k \in \{0, 1, \ldots, W\}$ in $O(nW)$ time. Since the weight of each rotation is $O(n)$, W is polynomially bounded. Hence a closed subset S minimizing $\delta(S)$ can be computed in polynomial-time. Clearly, this algorithm works for $(\infty, 2)$-SESM instances by reversing the roles of the men and the women.

Theorem 4.7.1 *Let I be a $(2, \infty)$-SESM (or $(\infty, 2)$-SESM) instance. Then, a sex-equal stable matching for I can be found in polynomial-time.*

4.8 An exact algorithm for (l, ∞)-SESM

4.8.1 The structure of D_Π

Properties of D_Π

In this section we describe an exact exponential-time algorithm for SESM when the men's preference lists are bounded in length by a constant $l \geq 3$ (if $l \leq 2$ then we can solve the problem in polynomial time). Our method hinges on the observation that when the number of rotations in the rotation digraph D_Π is at most $(5-\sqrt{24})(l-2)n$ (the reason for this particular factor of n becomes apparent later on), a brute-force algorithm that enumerates all subsets of the vertices of D_Π suffices to find a SESM. Otherwise, if the number of rotations exceeds that factor of n, we prove that G_Π must have bounded average degree. This allows us to use existing results concerning graphs with bounded average degree to design a moderately exponential time algorithm. In particular, we will apply the following theorem, which is due to Edwards and Farr [25], to G_Π.

Theorem 4.8.1 *(Edwards and Farr [25]). Let G be an undirected graph with n vertices and m edges of average degree $d \geq 4$, or a connected graph of average degree $d \geq 2$. Then, in $O(nm)$ time, a series-parallel induced subgraph P of G can be found such that $|P| \geq 3n/(d+1)$. Hence, if $N = G - P$, then $|N| \leq (d-2)n/(d+1)$.*

To see how this theorem may be used we will establish several key properties regarding D_Π. We establish a few bounds on the number of vertices and edges in G_Π in terms of the number n of agents of the instance, the lengths l of the men's preference lists, and the number r of rotations in D_Π. We begin by bounding from above the number of rotations, i.e. the number of vertices in D_Π, and the number of edges in D_Π.

Lemma 4.8.1 *Let I be an (l, ∞)-SMI instance, and D_Π its rotation digraph. Then, D_Π contains at most $(l-1)n/4$ rotations (vertices).*

Proof Any mutually acceptable (man,woman) pair can appear as a pair in at most one rotation in D_Π, except for a pair (m, w) such that w is ranked last on m's preference list, which cannot appear in any rotation. A rotation must always have at least two (man,woman) pairs. It follows that each rotation accounts for at least two distinct (man,woman) pairs, and the woman in such a pair may not be last on the man's preference list. Since there are $n/2$ men, $(l-1)n/4$ is an upper bound on the number of rotations. □

Lemma 4.8.2 *Let I be an (l, ∞)-SMI instance, and D_Π its rotation digraph. Then, the number of edges of D_Π is at most $(l-2)n/2$.*

Proof Consider any edge $e = (\rho', \rho) \in D_\Pi$. If ρ' is a type-1 predecessor of ρ (and possibly also a type-2 predecessor), then by definition there exists a pair (m_i, w_i) in ρ such that the elimination of ρ' matches m_i to w_i. Notice that w_i can neither be m_i's first or last choice. If instead ρ' is a type-2 predecessor of ρ, then there is a pair (m_i, w_i) in ρ such that ρ moves m_i below a woman $w \neq w_i$ and ρ' is the unique rotation that moves w above m_i. Notice in this case as well, w cannot be the first or last choice of m'.

Therefore, for every edge of D_Π we are able to identify a distinct (man,woman) pair (m, w) such that w is neither m's first nor last choice. Hence the number of edges of D_Π is bounded above by $(l-2)n/2$. □

Recall that our ultimate goal is to apply Theorem 4.8.1 to G_Π in a particular way as a part of the algorithm of this section. But, notice that Theorem 4.8.1 does not apply to graphs that are disconnected and have average degree less than 4. We will find it useful later on to know that we can connect G_Π as described in the following lemma.

Lemma 4.8.3 *Let G be a graph with c components and $m > 0$ edges. Then by adding a single vertex and $c+1$ edges to G a new connected graph G' may be formed with average degree ≥ 2.*

Proof Suppose that G has r vertices, so that $m \geq r - c$. Add a new vertex v together with an edge connecting v to a vertex in each component of G, and a second edge connecting v to a second vertex in one particular component. (Since $m > 0$ some component has more than one vertex.)

Then the new graph G' is connected, has $r' = r + 1$ vertices, $m' = m + c + 1$ edges, and average degree

$$d' = \frac{2m'}{r'} = \frac{2(m + c + 1)}{r + 1} \geq \frac{2(r + 1)}{r + 1} \geq 2.$$

□

Dealing with small components

The algorithm we describe in the forthcoming sections will rely on the fact that G_Π has no components with c vertices or fewer, where c is a fixed constant independent of the size of the input. In what follows we will show that we can use dynamic programming to preprocess the constant-sized components of G_Π in polynomial-time, allowing to make the assumption that no such components are present in G_Π.

Let $Q = Q_1, \ldots Q_t$ be the components of G_Π with at most c_0 vertices, where c_0 is an arbitrary constant. For each Q_i, construct a binary vector X_i, whose jth component is 1 if and only if there exists a closed subset of $\Pi[C_i]$ with weight exactly j (recall $\Pi[C_i]$ is the partially ordered set induced by C_i). The length of this vector is polynomially bounded (for example, the sum of the weights of all of the rotations in Π suffices). Since Q_i has constant size, computing this vector takes polynomial-time.

The next step is to compute a sequence of *combined binary vectors* Y_k such that the jth component of Y_k is 1 if and only if there exists a closed subset of $Q_1 \cup \ldots \cup Q_k$ with weight exactly j. To begin, set $Y_1 = X_1$. Suppose now that Y_i is known for some i $(1 \leq i < t)$. We compute the jth entry of Y_{i+1} (denoted Y_{i+1}^j) by the following formula:

$$Y_{i+1}^j = Y_i^j \vee \bigvee_{l=0}^{j} (Y_i^l \wedge X_{i+1}^{j-l}).$$

Hence the non-zero components of Y_t are exactly the weights attainable by the closed subsets of Q. The set Q may now be discarded from G_Π, giving a new graph G'_Π.

This procedure is invoked in the step labeled by (1) in the pseudocode description of the algorithm in Figure 4.2. The vector Y_t is stored and used later (specifically, in Section 4.9) when a closed subset corresponding to a sex-equal stable matching for the (original) instance is computed.

4.8.2 The algorithm

Algorithm idea

The general idea of the algorithm is the following. If Π contains sufficiently few rotations, the algorithm finds a sex-equal stable matching by brute force. Otherwise, we use Theorem 4.8.1 to partition G_Π into two parts, N and P, where P is a series-parallel graph (defined in Section 4.3) and the size of N is bounded. The algorithm then decides which rotations from N should be eliminated by explicitly trying all subsets N' of N. Notice of course that at least one such subset is a maximal subset of N that is contained in an optimal closed subset of Π. For a fixed subset of N', it may be that there exists $\rho \in N - N'$ such that, in D_Π, ρ precedes some $\rho' \in N'$, in which case we may immediately reject N'. Otherwise, if N' is *valid* in this sense, then some rotations from P, namely those that precede a rotation in N', are *forced* also to be eliminated. Other rotations, namely those with a predecessor in $N - N'$ cannot be eliminated and are *forbidden* for this choice of N'. Notice that since N' is valid, the sets of forced and forbidden rotations are disjoint. All other rotations in P are neither forced nor forbidden. Our goal is to find a subset $P' \cup Q'$ such that $P' \subseteq P$ and $Q' \subseteq Q$ (recall Q is the set of components of size at most c_0) such that $P' \cup Q'$ *extends* N' optimally in the following way:

(i) $N' \cup P' \cup Q'$ is a closed subset of Π. Note that this is equivalent to saying that P' includes every forced rotation and no forbidden rotations, and Q' is a closed subset of Q.
(ii) $N' \cup P' \cup Q'$ is an *optimal extension* of N' i.e. $P' \cup Q'$ minimizes $\delta(N' \cup P' \cup Q')$ over all choices of P' and Q' that satisfy (i).

In Section 4.9 we will show that a choice of $P' \cup Q'$ that satisfies the above criteria can be found in polynomial time. Hence the running time of the algorithm will be within a polynomial factor of $2^{|N|}$. We next describe the algorithm in detail.

Formal description of the algorithm

The algorithm, which is outlined in Figure 4.2, takes as input an (l, ∞)-SMI instance. It consists of two phases, the first is the preprocessing phase, which sets the stage for the second phase, which is

the main loop of the algorithm.

The preprocessing phase starts by computing the man-optimal stable matching M_0. If $\delta(M_0) \geq 0$, we are done, and simply output M_0. Next, in polynomial time we find Π, D_Π, and G_Π and assign each rotation the appropriate weight. If Π contains fewer than kn rotations, where n is the number of agents and k is $(5 - \sqrt{24})(l - 2)$, find a sex-equal stable matching by enumerating all closed subsets of Π. The justification for this choice of k becomes clear in the time complexity analysis of the algorithm. Otherwise, Π has at least kn rotations. In that case, we next preprocess the components of G_Π containing at most c_0 vertices as described in Section 4.8.1, computing the vector Y_t, and remove these components from G_Π, giving a new graph G'_Π.

If G'_Π is disconnected with average degree less than four, then connect G'_Π, so that it has average degree at least two as described in Lemma 4.8.3. Give the artificial vertex created in this process a weight of zero. Let G''_Π denote the resulting graph. Apply the Edwards and Farr algorithm described in Theorem 4.8.1 to G''_Π and find the sets N and P of the vertex partition. After the partition is found, we may discard the additional vertex if it lies in N, along with any edges incident to it. It is kept if it lies in P.

The main body of the algorithm then begins in the form of a loop, which iteratively considers every valid subset N' of N. For a given valid subset N', we identify the forced and forbidden vertices of P, and colour them black and red respectively. All other vertices of P are coloured white. The final step of the loop is to compute an optimal extension $P' \cup Q'$ for N'. The closed subset $S = N' \cup P' \cup Q'$ found in this loop minimizing $\delta(S)$ is kept and returned.

The next section is devoted to showing how step (2) in the pseudocode description of the algorithm may be accomplished in polynomial time.

4.9 Computing $P' \cup Q'$ in polynomial time

We assume that P is a connected graph, for, if it is not, we can always connect two series-parallel components (P_1, s_1, t_1) and (P_2, s_2, t_2) of P in series by creating a dummy vertex v with weight 0,

Preprocessing phase

compute M_0

if $\delta(M_0) \geq 0$:

    return M_0

compute Π, D_Π, and G_Π, and assign the rotations the appropriate weights

$k \leftarrow (5 - \sqrt{24})(l-2)$

if Π has fewer than kn rotations:

    return a SESM using complete enumeration of the closed subsets of rotations of Π

Compute the vector Y_t described in Section 4.8.1 (1)

$G'_\Pi \leftarrow$ graph resulting from preprocessing step described in Section 4.8.1

$G''_\Pi \leftarrow G'_\Pi$

if G'_Π is not connected **and** has average degree < 4:

    $G''_\Pi \leftarrow$ graph resulting by connecting G'_Π as described in Lemma 4.8.3

    assign the new vertex the weight zero

$N, P \leftarrow$ vertex partition of G''_Π (and Π) found by the Edwards and Farr algorithm

remove any artificial vertex from N along with any edges incident to it

Main loop

keep the closed subset S of Π minimising $\delta(S)$ found in the following loop

for each valid choice of $N' \subseteq N$:

    $P' \cup Q' \leftarrow$ optimal extension for N' (2)

    $S \leftarrow N' \cup P' \cup Q'$

Figure 4.2: Algorithm to find a SESM for an (l, ∞)-SMI instance

and adding the edges (t_1, v) and (s_2, v) to P. In terms of $\Pi[P]$, v is added as a maximal element of P. Since our algorithm for finding $P' \cup Q'$ is polynomial in n and m, this transformation will not influence the overall running time of the algorithm, as this step is performed *after* N and P have been computed. It is also irrelevant if the inclusion of additional edges changes the average degree of G_Π, again because N and P have already been found.

The plan is to use dynamic programming on an SP tree $\mathcal{T}$ for P to allow us to compute the choice of P'. Henceforth let H^i denote the series-parallel graph rooted at node i of $\mathcal{T}$, and s_i and t_i denote the two terminals of H^i. We will use the terminology *feasible* closed subset to denote a closed subset of H^i that contains every black vertex in H^i and none of the red vertices of H^i. Our goal, then, is to compute four binary vectors AA^i, AB^i, BA^i, and BB^i, the jth element of each of those being defined as follows:

AA^i_j = 1 if and only if there exists a feasible closed subset C of $\Pi[H^i]$ of weight exactly j such that $s \in C$ and $t \in C$.
AB^i_j = 1 if and only if there exists a feasible closed subset C of $\Pi[H^i]$ of weight exactly j such that $s \in C$ and $t \notin C$.
BA^i_j = 1 if and only if there exists a feasible closed subset C of $\Pi[H^i]$ of weight exactly j such that $s \notin C$ and $t \in C$.
BB^i_j = 1 if and only if there exists a feasible closed subset C of $\Pi[H^i]$ of weight exactly j such that $s \notin C$ and $t \notin C$.

The length of each vector is bounded by $K = \sum_{\rho \in P}(w(\rho))$, which is polynomially bounded. For a leaf node i of $\mathcal{T}$ corresponding to an edge $e = (s, t)$ the four values are simple to compute. The first step is to initialize every component of each vector to be 0. The vectors are then potentially changed according to the following rules.

AA^i : If neither s nor t is red, then change $AA^i_{w(s)+w(t)}$ to be 1.
AB^i : If $s \prec t$, s is not red, and t is not black, change $AB^i_{w(s)}$ to be 1.
BA^i : If $t \prec s$, t is not red, and s is not black, change $BA^i_{w(t)}$ to be 1.
BB^i : If neither s nor t is black, set BB^i_0 to 1.

Lemma 4.9.1 *Let i be an internal node of $\mathcal{T}$ with a child node i_1. Let C be a feasible closed subset of $\Pi[H^i]$, and $C_{i_1} = C \cap \Pi[H^{i_1}]$. Then, C_{i_1} is a feasible closed subset of $\Pi[H^{i_1}]$.*

Proof Let $c \in C_{i_1}$. If some predecessor $b \in \Pi[H^{i_1}]$ of c is not also in C, then clearly C is not a closed subset of $\Pi[H^i]$. Therefore b is in C and hence also in C_{i_1}. It follows that every predecessor of c in $\Pi[H^{i_1}]$ is in C_{i_1}, implying that C_{i_1} is a closed subset of $\Pi[H^{i_1}]$. If c, or any of c's predecessors were red, then C could not be feasible. Similarly, if C_{i_1} did not contain every black vertex in i_1, C could not be feasible. So, C_{i_1} is also feasible. □

We therefore derive the following lemma, whose proof is immediate in light of Lemma 4.9.1.

Lemma 4.9.2 *Every feasible closed subset C of $\Pi[H^i]$ consists of the union of two feasible closed subsets C_{i_1} and C_{i_2} of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, where i_1 and i_2 are the children of node i.*

4.9.1 Series nodes

The following lemma is necessary to understand how to compute the vectors associated with a series node i of $\mathcal{T}$.

Lemma 4.9.3 *Let i be a series node of $\mathcal{T}$ with child nodes i_1 and i_2, and let r denote the single vertex in $H^{i_1} \cap H^{i_2}$. Suppose C_{i_1} and C_{i_2} are feasible closed subsets of $\Pi[H^{i_1}]$, $\Pi[H^{i_2}]$, respectively. If either (i) $r \notin C_{i_1} \wedge r \notin C_{i_2}$ or (ii) $r \in C_{i_1} \wedge r \in C_{i_2}$, then $C = C_{i_1} \cup C_{i_2}$ is a feasible closed subset of $\Pi[H^i]$.*

Proof

(*Feasibility*). Since C_{i_1} and C_{i_2} are both feasible, they contain no red vertices and every black vertex in $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, so C must be feasible.

(*Closure*). Suppose for a contradiction that C is not a closed subset of $\Pi[H^i]$. Then, $\exists x \in C$ with a predecessor $y \notin C$. Suppose, without loss of generality, that $x \in \Pi[H^{i_1}]$. If $y \in \Pi[H^{i_1}]$ as well (note that possibly $y = r$), then $y \notin C_{i_1}$, implying C_{i_1} is not closed for $\Pi[H^{i_1}]$, a contradiction. So, suppose that $y \in \Pi[H^{i_2}]$, and that $y \neq r$. Then, there exists a sequence of immediate successors

of $y = y_0 \prec \ldots \prec y_z = x$ in $\Pi[H^i]$, and since r is the unique vertex in $H^{i_1} \cap H^{i_2}$, we must have $y \prec r \prec x$. If case (i) from the statement of the lemma holds, then we have that $r \prec x$, but $r \notin C_{i_1}$, so C_{i_1} is not closed for $\Pi[H^{i_1}]$. If case (ii) holds, then $y \prec r$, but $r \in C_{i_2}$, implying C_{i_2} is not closed for $\Pi[H^{i_2}]$. □

Lemma 4.9.3 essentially established a sufficiency condition to create a feasible closed subset C from the union of two feasible closed subsets C_{i_1} and C_{i_2}. Lemma 4.9.4 will, in a sense, establish necessity in that feasible closed subsets C_{i_1} and C_{i_2} satisfying either case (i) or (ii) of the previous lemma always exist for a given C.

Lemma 4.9.4 *Let i be a series node of $\mathcal{T}$ with child nodes i_1 and i_2, and let r denote the single node in $H^{i_1} \cap H^{i_2}$. Let C be a feasible closed subset of $\Pi[H^i]$. Then, there exist feasible closed subsets C_{i_1} and C_{i_2} of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, such that $C_{i_1} \cup C_{i_2} = C$, and either (i) $r \notin C_{i_1} \wedge r \notin C_{i_2}$ or (ii) $r \in C_{i_1} \wedge r \in C_{i_2}$.*

Proof Lemma 4.9.2 establishes the existence of feasible closed subsets C_{i_1} and C_{i_2} of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, such that $C_{i_1} \cup C_{i_2} = C$. We will establish the claim by showing that if $r \in C$ then $C_{i_1} \cup \{r\}$ and $C_{i_2} \cup \{r\}$ remain closed. This is easy to see – since $C_{i_1} \cup C_{i_2} = C$ is closed, every predecessor of r in $\Pi[H^{i_1}]$ (respectively, $\Pi[H^{i_2}]$) is also in C_{i_1} (respectively, C_{i_2}). □

The following theorem follows from Lemmas 4.9.3 and 4.9.4, and is the key to describing the dynamic programming procedure for processing a series node i of $\mathcal{T}$.

Theorem 4.9.1 *Let i be a series node of $\mathcal{T}$ with child nodes i_1 and i_2, and let r denote the single node in $H^{i_1} \cap H^{i_2}$. A feasible closed subset of $\Pi[H^i]$ of weight exactly j exists if and only if there exist feasible closed subsets C_{i_1}, C_{i_2} of $\Pi[H^{i_1}]$, $\Pi[H^{i_2}]$, respectively, such that (i) $r \notin C_{i_1} \wedge r \notin C_{i_2}$, with $w(C_{i_1}) = l$ and $w(C_{i_2}) = j - l$ or (ii) $r \in C_{i_1} \wedge r \in C_{i_2}$ with $w(C_{i_1}) = l$ and $w(C_{i_2}) = j - l + w(r)$.*

We are now in a position to describe the construction of the four binary vectors associated with a series node. Let i be a series node of $\mathcal{T}$ with children i_1 and i_2 and terminal nodes s_i and t_i. Let r denote the unique vertex in both H^{i_1} and H^{i_2}. Suppose we wish to compute AA^i_j, and that there

exists a feasible closed subset C of H^i. There are two cases to consider. If $r \in C$, then by Theorem 4.9.1 there exists a value l such that $(AA_l^{i_1} \wedge AA_{j-l+w(r)}^{i_2}) = 1$. If instead $r \notin C$, then there exists a value l such that $(AB_l^{i_1} \wedge BA_{j-l}^{i_2}) = 1$. This leads to the formula,

$$AA_j^i = (\bigvee_{l=w(r)}^{j} AA_l^{i_1} \wedge AA_{j-l+w(r)}^{i_2}) \vee (\bigvee_{l=0}^{j} AB_l^{i_1} \wedge BA_{j-l}^{i_2}).$$

The reasoning behind the next 3 formulae is similar, we present only the final formulae below.

$AB_j^i = (\bigvee_{l=w(r)}^{j} AA_l^{i_1} \wedge AB_{j-l+w(r)}^{i_2}) \vee (\bigvee_{l=0}^{j} AB_l^{i_1} \wedge BB_{j-l}^{i_2})$.

$BA_j^i = (\bigvee_{l=w(r))}^{j} BA_l^{i_1} \wedge AA_{j-l+w(r)}^{i_2}) \vee (\bigvee_{l=0}^{j} BB_l^{i_1} \wedge BA_{j-l}^{i_2})$.

$BB_j^i = (\bigvee_{l=w(r)}^{j} BA_l^{i_1} \wedge AB_{j-l+w(r)}^{i_2}) \vee (\bigvee_{l=0}^{j} BB_l^{i_1} \wedge BB_{j-l}^{i_2})$.

4.9.2 Parallel nodes

Our approach for parallel nodes is similar to that for series nodes. Our first goal is to establish an analogous claim for parallel nodes to that made in Theorem 4.9.1. The first step is the following lemma.

Lemma 4.9.5 *Let i be a parallel node of $\mathcal{T}$ with child nodes i_1 and i_2, and $\{s,t\}$ the two nodes in $H^{i_1} \cap H^{i_2}$. Let C_{i_1}, C_{i_2} be feasible closed subsets of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively. If $C_{i_1} \cap \{s,t\} = C_{i_2} \cap \{s,t\}$, then $C_{i_1} \cup C_{i_2}$ is a feasible closed subset of $\Pi[H^i]$.*

Proof (*Feasibility*). By definition, C_{i_1} and C_{i_2} contain all black vertices in $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, and neither can contain any red vertices. Hence $C_{i_1} \cup C_{i_2}$ contains all black vertices in $\Pi[H^i]$ and contains no red vertices.

(*Closure*). Suppose $C = C_{i_1} \cup C_{i_2}$ is not closed for $\Pi[H^i]$. Then, $\exists\ x \in C$ such that x has a predecessor $y \neq x$ and $y \notin C$. Suppose without loss of generality that $x \in \Pi[H^{i_1}]$, implying $x \in C_{i_1}$. If $y \in \Pi[H^{i_1}]$, (note that possibly $y \in \{s,t\}$) then C_{i_1} is not closed, a contradiction.

So, suppose $y \in \Pi[H^{i_2}]$ and $y \notin \{s,t\}$. Since $y \prec x$, there exists a sequence of successors $y = y_0 \prec \ldots \prec y_z = x$ in $\Pi[H^i]$. The only nodes in $H^{i_1} \cap H^{i_2}$ are s and t, so either $y \prec s \prec x$ or $y \prec t \prec x$ (or both). We continue the proof based on the following four cases.

(i) $s,t \notin C_{i_1}$ and $s,t \notin C_{i_2}$. Since we have either $s \prec x$ or $t \prec x$ (or both), C_{i_1} cannot be closed, a contradiction. (ii) $s,t \in C_{i_1}$ and $s,t \in C_{i_2}$. Since $y \notin C_{i_2}$ and either $y \prec s$ or $y \prec t$, C_{i_2} is not closed, a contradiction. (iii) $s \in C_{i_1}$, $t \notin C_{i_1}$ and $s \in C_{i_2}$, $t \notin C_{i_2}$. If $y \prec s \prec x$, then C_{i_2} is not closed, a contradiction. If instead $y \prec t \prec x$, then C_{i_1} is not closed, as $t \prec x$ and $x \in C_{i_1}$. (iv) $s \notin C_{i_1}$, $t \in C_{i_1}$ and $s \notin C_{i_2}$, $t \in C_{i_2}$. This case is analogous to case (iii).

□

The next lemma is analogous to Lemma 4.9.4 for series nodes.

Lemma 4.9.6 *Let i be a parallel node of $\mathcal{T}$ with child nodes i_1 and i_2, and $\{s,t\}$ the two nodes in $H^{i_1} \cap H^{i_2}$. Suppose that C is a feasible closed subset of $\Pi[H^i]$. Then, there exist feasible closed subsets C_{i_1} and C_{i_2} of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, such that $C_{i_1} \cap \{s,t\} = C_{i_2} \cap \{s,t\} = C \cap \{s,t\}$.*

Proof Lemma 4.9.2 establishes the existence of feasible closed subsets C_{i_1} and C_{i_2} of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, such that $C_{i_1} \cup C_{i_2} = C$. Let $B = C \cap \{s,t\}$. We will establish the claim by showing that $C_{i_1} \cup B$ and $C_{i_2} \cup B$ remain closed. This is easy to see – since $C_{i_1} \cup C_{i_2} = C$ is closed, every predecessor of an element $b \in B$ from $\Pi[H^{i_1}]$ (respectively $\Pi[H^{i_2}]$) is also in C_{i_1} (respectively C_{i_2}). □

Now we state the main theorem for describing the dynamic programming procedure for processing a parallel node i of $\mathcal{T}$.

Theorem 4.9.2 *Let i be a parallel node of $\mathcal{T}$ with child nodes i_1 and i_2, and $\{s,t\}$ the two nodes in $H^{i_1} \cap H^{i_2}$. A feasible closed subset of $\Pi[H^i]$ of weight exactly j exists if and only if there exist feasible closed subsets C_{i_1} and C_{i_2} of $\Pi[H^{i_1}]$ and $\Pi[H^{i_2}]$, respectively, such that*

1. *$s,t \notin C_{i_1}$ and $s,t \notin C_{i_2}$, $w(C_{i_1}) = l$, and $w(C_{i_2}) = j - l$; or*

2. $s \in C_{i_1}, t \notin C_{i_1}$ *and* $s \in C_{i_2}, t \notin C_{i_2}$, $w(C_{i_1}) = l$, *and* $w(C_{i_2}) = j - l + w(s)$; *or*

3. $s \notin C_{i_1}, t \in C_{i_1}$ *and* $s \notin C_{i_2}, t \in C_{i_2}$, $w(C_{i_1}) = l$, *and* $w(C_{i_2}) = j - l + w(t)$; *or*

4. $s, t \in C_{i_1}$ *and* $s, t \in C_{i_2}$, $w(C_{i_1}) = l$, *and* $w(C_{i_2}) = j - l + w(s) + w(t)$.

This leads to the following four formulae.

$AA^i_j = (\bigvee_{l=w(s)+w(t)}^{j} AA^{i_1}_l \wedge AA^{i_2}_{j-l+w(s)+w(t)})$.

$AB^i_j = (\bigvee_{l=w(s)}^{j} AB^{i_1}_l \wedge AB^{i_2}_{j-l+w(s)})$.

$BA^i_j = (\bigvee_{l=w(t)}^{j} BA^{i_1}_l \wedge BA^{i_2}_{j-l+w(t)})$.

$BB^i_j = (\bigvee_{l=0}^{j} BB^{i_1}_l \wedge BB^{i_2}_{j-l})$.

Suppose now that the four binary vectors have been computed for the root node *root* of $\mathcal{T}$. Recall that we also have computed the so-called combined vector Y_t as described in Section 4.8.1. We choose a position j of AA^{root}, AB^{root}, BA^{root}, or BB^{root} with a non-zero entry along with a position k of Y_t with a nonzero entry that together minimizes $\delta(M_0) + w(N') + j + k$. The actual feasible closed subset $P' \cup Q'$ of P can be found by simple modifications and the standard traceback technique through the dynamic programming tables. Thus we have computed an optimal extension for N' in polynomial time.

Theorem 4.9.3 *Let N' be a valid subset in an arbitrary iteration of the main loop of the algorithm described in Figure 4.2. An optimal extension $P' \cup Q'$ for N' can be computed in polynomial-time.*

4.10 Putting it all together

We have established the correctness of the algorithm described in the preceding sections. All that remains is to provide an upper bound on the time complexity. The following theorem establishes the running time.

Theorem 4.10.1 *Let I be an (l, ∞)-SMI instance. Then, a sex-equal stable matching for I can be computed in $O^\star(2^{\alpha n})$ time, where $\alpha = (5 - \sqrt{24})(l - 2)$. Hence the running time is $O^\star(1.0725^n)$, $O^\star(1.1503^n)$, $O^\star(1.2338^n)$, ... for $l = 3, 4, 5, \ldots$.*

Proof

Let G_Π denote the input graph, with r vertices (rotations), m edges, and average degree d. If the algorithm terminates because the number of rotations is at most αn, the theorem is obviously true, as we can enumerate every subset of Π to compute a sex-equal stable matching in $O^\star(2^{\alpha n})$ steps. Suppose instead that Π has tn rotations for some constant t, which, by Lemma 4.8.1 is at most $((l-1)/4)$.

We consider two cases. In the first case we suppose that after preprocessing the constant-sized components of G_Π, the resulting graph G'_Π satisfies Theorem 4.8.1, so that step (1) in Figure 4.2 is not performed. Clearly the time complexity of the algorithm is $O^\star(2^{(d'-2)r'/(d'+1)})$, where d is the average degree of G'_Π. This is defined to be $2m'/r'$, where m' and r' are the number of edges and vertices, respectively, of G'_Π. Since $m' \leq m$, and, by Lemma 4.8.2 we have that $m \leq (l-2)n/2$, we can provide the following upper bound on d':

$$d' \;=\; \frac{2m'}{r'} \;\leq\; \frac{2m}{r'} \;\leq\; \frac{(l-2)n}{r'}.$$

Consider the exponent $e(x)$ of the expression $2^{\frac{(d'-2)r'}{d'+1}}$, namely,

$$e(x) \;=\; \frac{(l-2)nr' - 2r'^2}{(l-2)n + r'} \;=\; \frac{kx - 2x^2}{k + x}.$$

where $x = r'$ and $k = (l-2)n$.

Let us use differentiation to compute the maximum value of $e(x)$:

$$e'(x) = \frac{(k+x)(k-4x) - (kx - 2x^2)}{(k+x)^2}.$$

$$e'(x) = 0 \Leftrightarrow 2x^2 + 4kx - k^2 = 0.$$

$$e'(x) = 0 \Leftrightarrow x = (\sqrt{\frac{3}{2}} - 1)k.$$

Substituting into (1) shows that the maximum value of $e(x)$ is $(5 - \sqrt{24})(l - 2)n$, precisely the expression in the claim of the theorem.

The second case of the proof is if, after preprocessing the constant-sized components of G_{Π}, the resulting graph G'_{Π} with m' edges, r' vertices, and average degree d' does not satisfy Theorem 4.8.1, so that step (1) in the pseudocode description is performed. Denote the graph resulting from this step (1) in the pseudocode by $G_{\Pi''}$, with m'' edges, r'' vertices, and average degree d'' = $2m''/r''$. We have the following facts.

1. $r'' = r' + 1$.
2. $m'' = m' + c' + 1$, where c' is the number of components in G'.
3. $r' \geq c'/(c_0 + 1)$, since each component of G' has at least $c_0 + 1$ vertices.
4. $d' < 4$, as G'_{Π} does not satisfy the requirement of Theorem 4.8.1.

This allows us to establish an upper bound on d'':

$$d'' = \frac{2m''}{r''} = \frac{2(m' + c' + 1)}{r''} < \frac{2m'}{r'} + \frac{2c'}{r'} + \frac{2}{r''} < d' + 2 + 1 = 7.$$

We next derive an upper bound for r'' by rewriting the expression $m'' = m' + c' + 1$ as below:

$$\frac{r''d''}{2} = \frac{r'd'}{2} + c' + 1 \quad \Rightarrow \quad \frac{r''d''}{2} < \frac{r''d'}{2} + c' + 1$$

giving

$$r'' < \frac{2(c' + 1)}{d'' - d'}.$$

Now, consider the exponent $e(r'') = \frac{(d''-2)r''}{d''+1}$ of the expression $2^{\frac{(d''-2)r''}{d''+1}}$. Since $d'' < 7$ and $r'' < \frac{2(c'+1)}{d''-d'}$, we have that

$$e(r'') \leq \frac{5}{8}\frac{2(c'+1)}{d''-d'}.$$

By fact (3) above, this gives

$$e(r'') \leq \frac{5}{4}\frac{r'+c_0+1}{(d''-d')(c_0+1)} \leq \frac{5}{4}\frac{\frac{(l-1)n}{4}+c_0+1}{(d''-d')(c_0+1)}.$$

Therefore, by choosing c_0 to be sufficiently large, $e(r'')$ can be made strictly less than $(5-\sqrt{24})(l-2)n$. Hence the theorem.

□

4.11 Conclusions and open problems

We have given a complete characterisation of the parameterized complexity of (α, β)-SESM. When the preference lists on one side are of length at most two, the problem is solvable in polynomial time, but, if the preference lists on either side are allowed instead to be of length three or greater, the problem is W[1]-hard.

As far as we know, our exponential-time algorithm is the first 'moderately' exponential-time algorithm for any computationally hard stable marriage variant. Perhaps further research could be devoted to finding reasonably fast exponential-time algorithms for other SMI-based problems. As a next step, one could consider searching for an exact algorithm for SESM with no bound on the lengths of the preference lists.

In his PhD thesis, Feder [27] describes the so-called *balanced stable matching problem*, namely to find a stable matching M that minimizes

$$\max\Big\{\sum_{(m_i,w_j)\in M} p_{m_i}(w_j), \sum_{(m_i,w_j)\in M} p_{w_j}(m_i)\Big\}$$

over all $M \in \mathcal{M}$. Intuitively, a balanced stable matching minimizes the unhappiness of the most unhappy group of people (the men or the women). Feder [27] proved that this problem is NP-hard. Is the balanced stable matching problem solvable by similar techniques to that presented in this chapter?

An immediate corollary to our dynamic programming algorithm presented in Section 4.9 is that whenever the underlying graph G_Π of the rotation poset Π of an arbitrary SESM instance is series-parallel, a sex-equal stable matching can be computed in polynomial time. We conjecture that SESM can be solved in polynomial time whenever G_Π has bounded *treewidth* (see, for example, one of the many surveys by Bodlaender [10] for the relevant background on treewidth). Specifically, whenever G_Π has treewidth bounded by a value k, we conjecture a sex-equal stable matching can be found in time $O(n^{O(1)}f(k))$, where $f(k)$ is a (probably exponential) function dependent only on k. For example, the running time could be $O(n^{O(1)}2^k)$.

This leads to further questions about treewidth and hard stable marriage problems. Are there other hard stable marriage problems that can be solved in polynomial time when G_Π has bounded treewidth? A particularly interesting problem could be the median stable matching problem (discussed in Section 2.2.4). Is this efficiently solvable when G_Π has bounded treewidth?

Finally, we remark that very recently an improvement in the Edwards and Farr theorem has been made [24]. This probably implies an improvement in the upper bound of the running time described in Theorem 4.10.1.

Chapter 5

Keeping couples together

5.1 Introduction

It is something of a folklore theorem that when the residents of an HR instance are not only interested in their own assignment, but are also interested in *someone else's* assignment, then stable matchings may not exist and they may be computationally difficult to find when they do exist. It is perhaps not surprising, then, that determining how a centralized matching algorithm should deal with *couples* (pairs of residents) is a significant computational challenge.

In Section 2.2.6, we reviewed the relevant literature and background regarding the Hospitals / Residents problem with couples (HRC). In the present chapter, we continue the search for algorithmic results in the HRC setting. Specifically, we consider a natural restriction of HRC in which each member of a given couple (r_i, r_j) has an individual preference list over a subset of hospitals, and the joint preference list of the couple is consistent with the individual preferences of r_i and r_j in a precise sense. Specifically, (r_i, r_j) ranks distinct pairs of hospitals in order of preference, such that if (h_p, h_q) precedes (h_r, h_s) on this list then (i) either r_i prefers h_p to h_r or $h_p = h_r$, and (ii) either r_j prefers h_q to h_s, or $h_q = h_s$. We refer to this restriction of HRC as the *Hospitals / Residents problem with Consistent Couples* (HRCC).

Thus HRCC models a situation in which the members of each couple can agree to construct a joint preference list from their individual preferences consistently, in the sense that if a couple jointly prefers (h_p, h_q) to (h_r, h_s), then when comparing h_p to h_r, r_i would be no worse off, and similarly

when comparing h_q to h_s, r_j would be no worse off. This includes the case where both members of a given couple have identical individual preference lists, with the intended outcome being that they are either matched to the same hospital or not matched at all.

HRCC does not seem to have been studied previously in the literature from an algorithmic point of view. In this chapter we show that an instance I of HRCC need not admit a stable matching, and that the problem of deciding whether I admits a stable matching is NP-complete. This result holds even if the length of each resident's individual list and the length of each couple's joint list is at most three, and the capacity of each hospital is at most two, thus providing another highly restricted version of HRC that remains NP-complete, in addition to the case considered by Ronn [84]. This restriction is important from a practical viewpoint, because in many applications the preference lists on one side tend to be short (for example in the context of SFAS, residents are asked to rank up to ten hospitals in order of preference).

By contrast, we also give a linear-time algorithm to find a stable matching or report that none exists, for the case that stability is defined in terms of the classical Gale-Shapley stability (that is, each member of a couple can form a blocking pair with a hospital without regard to the other member of the couple). This version of stability can be motivated in the HRCC context as follows. Suppose that a given couple (r_i, r_j) is given the joint assignment (h_r, h_s) by a matching algorithm. Now suppose that r_i prefers some hospital h_p to h_r, whilst the joint assignment (h_p, h_s) is not acceptable to the couple for whatever reason (perhaps geographical separation). The previous agreement of the couple to supply a joint (consistent) preference list could be overridden in practice if r_i has an overarching desire to be allocated to h_p as opposed to h_r. In reality this could mean that either r_j moves with r_i to remain geographically close, and attempts to make an arrangement with h_p (or a hospital nearby) outside of the matching scheme, or r_j changes career, or indeed the couple even split up. In the spirit of "keeping couples together", this is a situation that we seek to avoid, thus motivating this stronger form of stability in the context of HRCC. Hence we obtain a natural restriction of HRC that, unlike the general problem, is solvable in polynomial time. Our algorithm does not make any assumptions regarding the lengths of the preference lists or regarding the hospital capacities.

We remark that a matching that satisfies classical stability in the context of HRCC is stable with respect to the criteria defined earlier by Roth and Ronn [89, 84] for HRC (see Section 5.2 for a formal definition of this stability criterion). The converse, however, is not true in general. Our

algorithm for HRCC under classical stability helps to narrow the search for the boundary between polynomial time solvable and NP-complete variants of HRC. In particular, HRCC under classical stability is the most general restriction of HRC that we are aware of that is solvable in polynomial time.

Hospitals / Residents problem with Sizes

A special case of HRCC arises when each couple (r_i, r_j) is such that that the individual preference lists of r_i and r_j are identical, and the joint preference list of (r_i, r_j) satisfies the property that $h_p = h_q$ for any element (h_p, h_q) on this list. Thus r_i and r_j wish to be either assigned to the same hospital, or both be unassigned. We refer to this restriction of HRCC as the *Hospitals / Residents problem with Inseparable Couples* (HRIC).

Let I be an instance of HRIC and let (r_i, r_j) be a couple in I. Given the structure of (r_i, r_j)'s preference list, it is natural to replace (r_i, r_j) by a single entity $C_{i,j}$ whose preference list is obtained from that of (r_i, r_j) by replacing each occurrence of (h_k, h_k) by h_k. Thus each single resident occupies one post at a given hospital, whilst each couple occupies two posts. This suggests a natural generalisation of HRIC to the case where each resident $r_i \in R$ has a *size* $s_i \in \mathbb{Z}^+$, indicating the number of posts that r_i occupies at any hospital. Hospitals will now rank residents of any size (including couples) as a single entity. We refer to this variant of HRC as the *Hospitals / Residents problem with Sizes* (HRS).

A formal definition of HRS is given in Section 5.2, in which we formulate an appropriate notion of stability for this context. With this stability definition we later prove that, given an HRS instance where the size of each resident is at most two and the capacity of each hospital is at most two, the problem of deciding whether a stable matching exists is NP-complete, even if the length of each preference list is at most three. We also show that the restriction of HRS in which each resident has size at most two is reducible to HRCC (essentially each resident of size two becomes a couple), thus implying the aforementioned NP-completeness result for HRCC.

However, by contrast, we also prove that, given an instance of HRS in which the length of each hospital's preference list is at most two, a stable matching always exists and can be found in linear time. The result holds for arbitrary resident sizes and hospital capacities. This result therefore indicates a boundary between the polynomial-time solvability and NP-completeness of HRS with

respect to the length of a hospital's preference list.

5.2 Formal definitions of HRS and HRCC

We firstly give a formal definition of the Hospitals / Residents problem with Sizes (HRS). An instance I of this problem is defined in the same way as an instance of HR (as defined in Section 5.1) except that each resident $r_i \in R$ has a *size* $s_i \in \mathbb{Z}^+$. An *assignment* M in I is a set of (resident,hospital) pairs such that $(r_i, h_j) \in M$ only if r_i and h_j find each other acceptable. For $r_i \in R$ we denote the set $\{h_j \in H : (r_i, h_j) \in M\}$ by $M(r_i)$, for $h_j \in H$ we denote $\{r_i \in R : (r_i, h_j) \in M\}$ by $M(h_j)$, and for $h_j \in H$ we denote $\sum\{s_i : r_i \in M(h_j)\}$ by O_j^M and refer to this as the *occupancy* of h_j in M. We say that h_j is *undersubscribed* if $O_j^M < c_j$, where c_j is the capacity of hospital h_j.

A *matching* is an assignment M such that $|M(r_i)| \leq 1$ for each $r_i \in R$ and $O_j^M \leq c_j$ for each $h_j \in H$. In other words, in a matching, each resident is assigned to at most one hospital, and the sum of the sizes of the residents assigned to a hospital does not exceed its capacity. Given a matching M in which a resident r_i is matched to a hospital h_j, with a slight abuse of notation we let $M(r_i)$ denote h_j. A pair $(r_i, h_j) \in R \times H$ *blocks* a matching M, or is a *blocking pair* for M, if

1. r_i is unmatched, or r_i prefers h_j to $M(r_i)$, and
2. $O_j^M + s_i \leq c_j$, or h_j prefers r_i to residents $r_{k_1}, \ldots r_{k_t} \in M(h_j)$ such that

$$O_j^M + s_i - \sum_{p=1}^{t} s_{k_p} \leq c_j.$$

The definition implies that h_j could participate in a blocking pair with r_i if (i) either h_j currently has room for r_i, or (ii) h_j can make room for r_i by rejecting a set of residents it ranks lower than r_i. A matching is *stable* if it admits no blocking pair.

We assume without loss of generality that, for each $r_i \in R$ and for each hospital h_j on r_i's preference list, $s_i \leq c_j$, for otherwise (r_i, h_j) could never belong to a stable matching, nor could (r_i, h_j) form a blocking pair.

We firstly observe that HR is clearly the special case of HRS in which $s_i = 1$ for each $r_i \in R$. The

$$\begin{array}{llll} 1: & r_1: & h_2 & h_1 \\ 1: & r_2: & h_1 & h_2 \\ 2: & r_3: & h_1 & \end{array} \qquad \begin{array}{llll} 2: & h_1: & r_1 & r_3 \quad r_2 \\ 1: & h_2: & r_2 & r_1 \end{array}$$

Figure 5.1: An HRS instance for which no stable matching exists

blocking pair definition for HR, which is given in Section 2.2.6, can then be deduced from that for HRS by interpreting Condition (2) as follows: either h_j is undersubscribed or prefers r_i to some resident in $M(h_j)$.

We next observe that, in contrast to HR, an HRS instance may not admit a stable matching. An example instance I that illustrates this is shown in Figure 5.1 (in this figure, and throughout the chapter, sizes and capacities are written next to the residents and hospitals, respectively). Suppose for a contradiction that I admits a stable matching M. If $(r_3, h_1) \in M$, then $(r_1, h_2) \in M$ or else (r_1, h_1) blocks M. Hence (r_2, h_2) blocks M, a contradiction. Suppose instead that $(r_3, h_1) \notin M$. Then $(r_2, h_1) \in M$, or else (r_2, h_1) blocks M (since h_1 has capacity two). Hence $(r_1, h_2) \in M$ or else (r_1, h_2) blocks M. This implies that (r_3, h_1) blocks M, a contradiction.

Our third observation is that the restriction of HRS where each resident has size at most two is reducible to the Hospitals / Residents problem with Consistent Couples (HRCC), which is a special case of the Hospitals / Residents problem with Couples (HRC). We demonstrate this in Lemma 5.2.1, but first we give a formal definition of each of HRC and HRCC.

An instance I of HRC involves a set $R = \{r_1, \ldots, r_n\}$ of *residents*, a set $H = \{h_1, \ldots, h_m\}$ of *hospitals*, and a set C of *couples*, i.e., ordered pairs of residents such that each resident appears in at most one pair. As in the HR case, each hospital $h_j \in H$ has a *capacity* $c_j \in \mathbb{Z}^+$.

Each *single* resident $r_i \in R$ (i.e., a resident who does not belong to a couple) submits a strict preference list of acceptable hospitals. Each couple (r_i, r_j) submits a joint (strict) preference list over pairs of acceptable hospitals. Each entry in this list is an ordered pair (h_k, h_l) of (not necessarily distinct) hospitals representing the assignment of r_i to h_k and of r_j to h_l. Finally, each hospital $h_j \in H$ ranks those residents (whether single or a member of a couple) who find h_j acceptable in strict order of preference.

In this context, the definition of a matching M is the same as in the classical HR setting (see Section 2.2.6), with the additional requirement that, for each couple (r_i, r_j), if $(r_i, h_k) \in M$ and

$(r_j, h_l) \in M$ then the pair (h_k, h_l) must appear on the joint preference list of that couple. A matching M is unstable if at least one of the following holds:

1. The matching is blocked by a hospital h_j and a single resident r_i, as in the classical HR problem (defined in Section 2.2.6).

2. The matching is blocked by a hospital h_k and a resident r_i who is coupled, say with r_j; that is, (r_i, r_j) prefers $(h_k, M(r_j))$ to $(M(r_i), M(r_j))$, and h_k is either undersubscribed in M or prefers r_i to some member of $M(h_k) \backslash \{r_j\}$.

3. The matching is blocked by a couple (r_i, r_j) and (not necessarily distinct) hospitals $h_k \neq M(r_i)$, $h_l \neq M(r_j)$; that is, (r_i, r_j) prefers the joint assignment (h_k, h_l) to $(M(r_i), M(r_j))$, and *either*

   (a) $h_k \neq h_l$, and h_k (respectively h_l) is either undersubscribed in M or prefers r_i (respectively r_j) to at least one of its assigned residents in M; *or*

   (b) $h_k = h_l$, and h_k has at least two free posts in M, i.e., $c_k - |M(h_k)| \geq 2$; *or*

   (c) $h_k = h_l$, and h_k has one free post in M, i.e., $c_k - |M(h_k)| = 1$, and h_k prefers at least one of r_i, r_j to some member of $M(h_k)$; *or*

   (d) $h_k = h_l$, h_k is full in M, h_k prefers r_i to some $r_s \in M(h_k)$, and h_k prefers r_j to some $r_t \in M(h_k) \backslash \{r_s\}$.

The above stability definition for HRC extends that given in [36, Section 1.6.6], in order to deal with the case that $h_k = h_l$, given a couple (r_i, r_j) who prefer (h_k, h_l) to $(M(r_i), M(r_j))$. As far as we are aware, this possibility does not appear to have been covered adequately by previous stability definitions for HRC in the literature [88, 36, 84, 23, 13, 65, 66, 68].

HRCC is the special case of HRC in which each resident (whether single or a member of a couple) ranks a subset of H in strict order of preference. Each couple (r_i, r_j) ranks a subset of $H \times H$ in strict order, subject to the constraint that this joint preference list be *consistent* with the individual preference lists of r_i and r_j. That is, (r_i, r_j) prefers (h_p, h_q) to (h_r, h_s) only if (i) either r_i prefers h_p to h_r or $h_p = h_r$, and (ii) either r_j prefers h_q to h_s or $h_q = h_s$.

We now show that the restriction of HRS in which each resident has size at most two is polynomially reducible to HRCC.

Lemma 5.2.1 *The restriction of HRS in which each resident has size at most two can be reduced in polynomial time to HRCC.*

Proof Given an instance I of HRS, construct an instance I' of HRCC in the following way. For each resident r_i of size two, create a couple $(r_{i,1}, r_{i,2})$ in I'. Suppose the preference list of r_i in I is $h_1, h_2, \ldots h_t$. Assign to each of $r_{i,1}$ and $r_{i,2}$ an individual list equal to that of r_i. Let the joint preference list of $(r_{i,1}, r_{i,2})$ in I' be $(h_1, h_1), (h_2, h_2), \ldots, (h_t, h_t)$ – this is clearly consistent with the lists of $r_{i,1}$ and $r_{i,2}$.

For each hospital h_j that finds r_i acceptable in I, replace the entry r_i on h_j's preference list in I' with $r_{i,1}$ and $r_{i,2}$ in arbitrary order. Leave all residents of size one the same in I' as in I. This ends the transformation. We claim that a stable matching exists for I' if and only if one exists for I.

Suppose a stable matching M exists for I. Then, construct a matching M' for I' in the following way. If (r_i, h_j) is in M, place (r_i, h_j) into M' if r_i has size one, else place $(r_{i,1}, h_j)$ and $(r_{i,2}, h_j)$ into M'. Notice that the capacities of the hospitals are preserved in the reduction, and also that if a hospital h_j has an occupancy of t in M, then h_j is assigned t residents in M'. Suppose a blocking pair exists for M' in I'. Then, the blocking pair must be of Type 1 or 3 above, as Type 2 is impossible by the special nature of the couple's preference lists. If there is a blocking pair (r_i, h_j) by Type 1, M surely also had the same blocking pair in I. If instead M' is blocked by a pair of Type 3, then it must be because a couple $(r_{i,1}, r_{i,2})$ block with the pair (h_j, h_j) in I'. But then resident r_i of size two in I must also block with hospital h_j in M.

Conversely suppose a stable matching M' exists for I'. Then, construct a stable matching M for I in the following way. If (r_i, h_j) is in M', place (r_i, h_j) into M, if r_i has size one in I, else if $(r_{i,1}, h_j)$ and $(r_{i,2}, h_j)$ are in M', place (r_i, h_j) into M. By the nature of the preference lists, $r_{i,1}$ and $r_{i,2}$ are always assigned the same hospital. Suppose (r_i, h_j) blocks M in I. Then, by an argument similar to the above, (r_i, h_j) must have blocked M' in I' if r_i has size one, otherwise the pair $(r_{i,1}, r_{i,2})$ must have blocked M' in I' with (h_j, h_j). □

It follows immediately from the example of Figure 5.1 and Lemma 5.2.1 that an HRCC instance need not admit a stable matching.

5.3 NP-completeness of HRS and HRCC

This section describes a polynomial-time reduction that establishes NP-completeness for the problem of deciding whether a stable matching exists, given an HRS instance where the size and capacity of each resident and each hospital, respectively, is at most two, and the length of each preference list is at most three. This reduction is from a restricted variant of MAX-SMTI (see Section 2.2.5 and Chapter 3 for background and definitions). Define $(3,3)$-COM-SMTI to be the problem of deciding whether a complete stable matching exists (i.e., a stable matching that matches every agent), given an instance of SMTI in which each preference list is of length at most three, every woman's preference list is strictly ordered, and each man's preference list is either strictly ordered or is a tie of length two (all of these conditions holding simultaneously).

The proof that $(3,3)$-COM-SMTI is NP-complete is somewhat peripheral to the results of this chapter, moreover, we use the hardness of this problem again in Chapter 6. For this reason, the proof of the following theorem is presented in the Appendix.

Theorem 5.3.1 $(3,3)$*-COM-SMTI is NP-complete.*

5.3.1 The reduction

Given an instance I of $(3,3)$-COM-SMTI with n men m_1,m_2,. . .,m_n and n women w_1, w_2,. . .,w_n, we create an instance I' of HRS, whose residents and hospitals are constructed as follows. Firstly, a hospital h_t is created for each woman w_t in I.

Next, for each man m_i in I with a preference list consisting of a two-way tie (w_k, w_l) where $k < l$, create eight residents $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}, r_{i_{\alpha,1}}, r_{i_{\alpha,2}}, r_{i_{\beta,1}}, r_{i_{\beta,2}}\}$ and six hospitals $\{h_{i,1}, h_{i,2}, h_{i_{\alpha,1}}, h_{i_{\alpha,2}}, h_{i_{\beta,1}}, h_{i_{\beta,2}}\}$. The preference lists, sizes and capacities of these eight residents and six hospitals are shown in Figure 5.2. For each man m_s in I with a strictly ordered preference list $w_{s_1}, w_{s_2}, \ldots, w_{s_y}$, create three residents $\{r_s, r_{s_{\gamma,1}}, r_{s_{\gamma,2}}\}$ and two hospitals $\{h_{s_{\gamma,1}}, h_{s_{\gamma,2}}\}$. The preference lists, sizes and capacities of these three residents and two hospitals are also shown in Figure 5.2.

Finally, for each hospital h_t created from a woman w_t with preference list $m_{t_1}, \ldots, m_{t_z}$, set the preference list of h_t to initially be equal to $m_{t_1}, \ldots, m_{t_z}$, temporarily placing "men" on h_t's pref-

$2: \; r_{i,1}: \; h_{i,1} \; h_k \; h_{i_{\alpha,1}}$ $\qquad$ $2: \; h_{i,1}: \; r_{i,4} \; r_{i,1} \; r_{i,3}$
$2: \; r_{i,2}: \; h_{i,2} \; h_l \; h_{i_{\beta,1}}$ $\qquad$ $2: \; h_{i,2}: \; r_{i,3} \; r_{i,2} \; r_{i,4}$
$1: \; r_{i,3}: \; h_{i,1} \; h_{i,2}$
$1: \; r_{i,4}: \; h_{i,2} \; h_{i,1}$

$1: \; r_{i_{\alpha,1}}: \; h_{i_{\alpha,2}} \; h_{i_{\alpha,1}}$ $\qquad$ $2: \; h_{i_{\alpha,1}}: \; r_{i_{\alpha,1}} \; r_{i,1} \; r_{i_{\alpha,2}}$
$1: \; r_{i_{\alpha,2}}: \; h_{i_{\alpha,1}} \; h_{i_{\alpha,2}}$ $\qquad$ $1: \; h_{i_{\alpha,2}}: \; r_{i_{\alpha,2}} \; r_{i_{\alpha,1}}$

$1: \; r_{i_{\beta,1}}: \; h_{i_{\beta,2}} \; h_{i_{\beta,1}}$ $\qquad$ $2: \; h_{i_{\beta,1}}: \; r_{i_{\beta,1}} \; r_{i,2} \; r_{i_{\beta,2}}$
$1: \; r_{i_{\beta,2}}: \; h_{i_{\beta,1}} \; h_{i_{\beta,2}}$ $\qquad$ $1: \; h_{i_{\beta,2}}: \; r_{i_{\beta,2}} \; r_{i_{\beta,1}}$

$2: \; r_s: \; h_{s_1} \; h_{s_2} \; \ldots \; h_{s_y} \; h_{s_{\gamma,1}}$
$1: \; r_{s_{\gamma,1}}: \; h_{s_{\gamma,2}} \; h_{s_{\gamma,1}}$ $\qquad$ $2: \; h_{s_{\gamma,1}}: \; r_{s_{\gamma,1}} \; r_s \; r_{s_{\gamma,2}}$
$1: \; r_{s_{\gamma,2}}: \; h_{s_{\gamma,1}} \; h_{s_{\gamma,2}}$ $\qquad$ $1: \; h_{s_{\gamma,2}}: \; r_{s_{\gamma,2}} \; r_{s_{\gamma,1}}$

Figure 5.2: Preference lists in the constructed instance of HRS

erence list. Now, suppose that w_t finds some man m_j acceptable. If m_j's preference list is strictly ordered in I, replace m_j on h_t's preference list with r_j. If m_j's preference list is not strictly ordered, his preference list consists of a two-way tie, say, (w_t, w_k). If $t < k$, replace m_j with $r_{j,1}$ on h_t's preference list, else, replace m_j with $r_{j,2}$ on h_t's preference list. Set the capacity of h_t to be two.

This ends the reduction. Clearly, it is computable in polynomial time. We now argue that it is correct by the following sequence of lemmas, each of which states a property of any stable matching M' in I'.

Lemma 5.3.1 *Let m_s be a man with a strictly ordered preference list in I, and let m_i be a man with a preference list consisting of a two-way tie in I. Then, every resident in the set $\{r_s, r_{i,1}, r_{i,2}\}$ is matched to some hospital in M', and that hospital is not the last entry on his preference list.*

Proof Suppose that r_s is unmatched in M'. Then, $r_{s_{\gamma,1}}$ must be matched to $h_{s_{\gamma,1}}$, to prevent r_s from forming a blocking pair with $h_{s_{\gamma,1}}$. Resident $r_{s_{\gamma,2}}$ must be matched to $h_{s_{\gamma,1}}$ as well, for otherwise he forms a blocking pair with $h_{s_{\gamma,1}}$. But this implies $(r_{s_{\gamma,1}}, h_{s_{\gamma,2}})$ is a blocking pair for M'.

Suppose instead that r_s is matched to $h_{s_{\gamma,1}}$. Then, neither $r_{s_{\gamma,1}}$ nor $r_{s_{\gamma,2}}$ is matched to $h_{s_{\gamma,1}}$, else its capacity would be exceeded. So, if $r_{s_{\gamma,1}}$ is matched to $h_{s_{\gamma,2}}$, $r_{s_{\gamma,2}}$ is unmatched, and forms a blocking pair with $h_{s_{\gamma,2}}$. If, instead, $r_{s_{\gamma,2}}$ is matched to $h_{s_{\gamma,2}}$, then $r_{s_{\gamma,1}}$ is unmatched, and forms a blocking pair with $h_{s_{\gamma,1}}$. Clearly, if neither $r_{s_{\gamma,1}}$ nor $r_{s_{\gamma,2}}$ is matched to $h_{s_{\gamma,2}}$, they form blocking

pairs with $h_{s_{\gamma,2}}$. This exhausts every possibility. It follows that if r_s is unmatched in M' or is matched to the last hospital on his preference list, a blocking pair cannot be avoided.

The same argument holds for $r_{i,1}$ by substituting $h_{i_{\alpha,1}}$ and $h_{i_{\alpha,2}}$ for $h_{s_{\gamma,1}}$ and $h_{s_{\gamma,2}}$, respectively, and $r_{i_{\alpha,1}}$ and $r_{i_{\alpha,2}}$ for $r_{s_{\gamma,1}}$ and $r_{s_{\gamma,2}}$, respectively. Similarly, the argument is analogous for $r_{i,2}$, by replacing $r_{s_{\gamma,1}}$ and $r_{s_{\gamma,2}}$ with $r_{i_{\beta,1}}$ and $r_{i_{\beta,2}}$, respectively, and $h_{s_{\gamma,1}}$ and $h_{s_{\gamma,2}}$ with $h_{i_{\beta,1}}$ and $h_{i_{\beta,2}}$, respectively. □

Lemma 5.3.2 *For all men $m_i \in I$ with a preference list consisting of a two-way tie, the residents $r_{i,3}$ and $r_{i,4}$ are matched to some hospital in M'. Moreover, $r_{i,3}$ and $r_{i,4}$ are matched to the same hospital in M'.*

Proof If $r_{i,3}$ is not matched in M', he clearly forms a blocking pair with $h_{i,2}$, which has $r_{i,3}$ first on its preference list. Similarly, if $r_{i,4}$ is not matched, he blocks with $h_{i,1}$.

For the second claim, suppose $(r_{i,3}, h_{i,1})$ and $(r_{i,4}, h_{i,2})$ are in M'. Then $r_{i,1}$ cannot be matched to $h_{i,1}$ and $r_{i,2}$ cannot be matched to $h_{i,2}$ in M', for otherwise the capacities of $h_{i,1}$ and $h_{i,2}$ would be exceeded. However, this implies $(r_{i,1}, h_{i,1})$ and $(r_{i,2}, h_{i,2})$ form blocking pairs for M'. On the other hand, if $(r_{i,3}, h_{i,2})$ and $(r_{i,4}, h_{i,1})$ are in M', then $(r_{i,3}, h_{i,1})$ forms a blocking pair in M', for $h_{i,1}$ has enough spare capacity to admit $r_{i,3}$. □

Lemma 5.3.3 *For all men $m_i \in I$ with a preference list consisting of a two-way tie, exactly one of the residents in the set $\{r_{i,1}, r_{i,2}\}$ is matched to his first choice, and the other is matched to his second choice in M'.*

Proof By Lemma 5.3.2, it is clear that $r_{i,1}$ and $r_{i,2}$ cannot both be matched to their first choice in M', for this would result in $r_{i,3}$ and $r_{i,4}$ being unassigned in M', a contradiction.

On the other hand, if $r_{i,1}$ and $r_{i,2}$ are both matched to their second choice, then if $r_{i,3}$ and $r_{i,4}$ are both matched to $h_{i,1}$, and $r_{i,2}$ forms a blocking pair with $h_{i,2}$. If instead $r_{i,3}$ and $r_{i,4}$ are both matched to $h_{i,2}$, then $r_{i,1}$ forms a blocking pair with $h_{i,1}$.

Finally, by Lemma 5.3.1, $r_{i,1}$ and $r_{i,2}$ cannot be unmatched or matched to the last hospitals on their preference lists, so exactly one of $\{r_{i,1}, r_{i,2}\}$ is matched to his first choice in M', and the other to his second. □

Lemmas 5.3.1 – 5.3.3 lead us to the following corollary.

Corollary 5.3.1 *Every resident is matched in any stable matching M' for I'.*

Proof The only residents not yet shown to be matched in M' are those residents $r_{i_{\delta,k}}$ for $\delta \in \{\alpha, \beta\}$ and $k \in \{1, 2\}$ created from a man m_i with a preference list consisting of a tie of size two in I, and the residents $r_{s_{\gamma,k}}$ for $k \in \{1, 2\}$ created from a man m_s with a strictly ordered preference list in I. Each of these residents must be matched in M', for otherwise they would form a blocking pair with the last hospital on their preference list, which has the resident in question in first place on its list □

We are now in a position to prove the correctness of the reduction in one direction.

Lemma 5.3.4 *If the derived HRS instance I' admits a stable matching M', then the given instance I of (3,3)-COM-SMTI admits a complete stable matching M.*

Proof Given a stable matching M' for I', we describe how to construct a complete stable matching M in I' as follows. Consider the residents $r_{i,k}$ for $k \in \{1, 2, 3, 4\}$ that were created in correspondence to a man m_i in I with a preference list consisting of (w_k, w_l), a tie of size two, where $k < l$. By Lemma 5.3.3, either $r_{i,1}$ is matched to h_k or $r_{i,2}$ is matched to h_l in M', and, since the capacity of every hospital in I' is either two or one, no other resident is assigned to h_k if $r_{i,1}$ is, and similarly for $r_{i,2}$ and h_l. Hence, we construct M by placing (m_i, w_k) into M if and only if $(r_{i,1}, h_k) \in M'$, and (m_i, w_l) into M if and only if $(r_{i,2}, h_l) \in M'$. Again, Lemma 5.3.3 ensures that we always place exactly one such pair into M. To complete the construction of M, for each resident r_i corresponding to a man m_i with a strictly ordered preference list, place (m_i, w_j) into M if and only if $(r_i, h_j) \in M'$. Corollary 5.3.1 ensures every man in I is assigned to some woman in M; in what follows we will show that M is indeed a matching, and is also stable.

We have already argued by Lemma 5.3.3 that no two men with ties on their preference lists are matched to the same woman in M. For any resident r_i corresponding to a man m_i in I with a strictly ordered preference list, r_i must have size two, and is matched in M to a hospital h_j, which is not his last choice by Lemma 5.3.1, and which therefore corresponds to woman w_j in I. Hospital h_j has capacity two, and so is matched to only r_i in M'. This means that exactly one man is matched to w_j in M. Therefore, M is a matching.

We will show that M' is stable by demonstrating that no man can be part of a blocking pair relative to M. For any man m_i with a preference list consisting of a tie of size two, this must be true, for such a man is indifferent between the only two women on his preference list. Suppose instead that m_i has a strictly ordered preference list. Consider any woman w_l whom m_i prefers to his partner in M. Then, in M', resident r_i must also have preferred hospital h_l to its assigned hospital. By Lemma 5.3.1 and the construction of the hospitals of I', h_l's preference list contains residents of size two only. Since M' is stable, h_l is assigned a resident it strictly prefers to r_i, and hence, in M, w_l is assigned a man she strictly prefers to m_i. It follows that M is a complete stable matching in I. □

We now prove that the reduction is correct in the other direction.

Lemma 5.3.5 *If the given instance I of (3,3)-COM-SMTI admits a complete stable matching M, then, the derived HRS instance I' admits a stable matching M'.*

Proof Given a complete stable matching M for I, we describe how to construct a complete stable matching M' in I. For each man m_i with a strictly ordered preference list, place (r_i, h_j) into M' if and only if $(m_i, w_j) \in M$. For each man m_i in M with a tie of size two consisting of, say, (w_k, w_l), where $k < l$, place pairs in M' by the following two rules:

1. If $(m_i, w_k) \in M$, place $(r_{i,1}, h_k), (r_{i,2}, h_{i,2}), (r_{i,3}, h_{i,1}), (r_{i,4}, h_{i,1})$ into M' , and assign all residents $r_{i_{\delta,k}}$ $\forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ their first choice.

2. If $(m_i, w_l) \in M$, place $(r_{i,1}, h_{i,1}), (r_{i,2}, h_l), (r_{i,3}, h_{i,2}), (r_{i,4}, h_{i,2})$ into M', and assign all residents $r_{i_{\delta,k}}$ $\forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ their first choice.

It is easy to verify that the capacity of each hospital is not exceeded in M', and that M' is a matching. We claim that M' is also stable.

For, suppose residents $r_{i,t}$ for $t \in \{1, 2, 3, 4\}$ are matched by Rule 1 above. Immediately we notice that $r_{i,2}$ and $r_{i,3}$ are matched to their first choices, and hence cannot form a blocking pair with any hospital in I'. Resident $r_{i,1}$ prefers only hospital $h_{i,1}$ to his assignment in M', but does not form a blocking pair with it because $r_{i,3}$ and $r_{i,4}$ are matched to $h_{i,1}$. The remaining resident, $r_{i,4}$ prefers only $h_{i,2}$, who is matched to $r_{i,2}$, and hence does not form a blocking pair with $r_{i,4}$. All residents

$r_{i_{\delta,k}}$ $\forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ are matched to their first choice and cannot form a blocking pair with any hospital.

Suppose residents $r_{i,t}$ for $t \in \{1, 2, 3, 4\}$ are matched by Rule (2) above. In an argument analogous to the previous case, $r_{i,1}$ and $r_{i,4}$ are matched to their first choices, and cannot be part of a blocking pair. Resident $r_{i,2}$ prefers only hospital $h_{i,2}$ to his assignment in M', but does not form a blocking pair with it because $r_{i,3}$ and $r_{i,4}$ are matched to $h_{i,2}$. The remaining resident, $r_{i,3}$ prefers only $h_{i,1}$, who is matched to $r_{i,1}$, and hence cannot form a blocking pair with $r_{i,3}$. Again, the residents $r_{i_{\delta,k}}$ $\forall \delta \in \{\alpha, \beta\}$ and $\forall k \in \{1, 2\}$ are matched to their first choice and cannot form a blocking pair with any hospital.

In the final case, a resident r_i corresponding to a man m_i with a strictly ordered preference list in I' cannot block for the same reasons that m_i did not block in M. If m_i preferred a woman w_j in M, then r_i must also prefer h_j in M'. However, h_j must be matched to a resident that precedes r_i on its preference list, since w_j is matched to a man preceding m_i on her preference list. The capacity of h_j is two, and the size of r_i is also two, so h_j has insufficient capacity to accommodate r_i. Therefore, M' is a stable matching for I'. □

Lemmas 5.3.4 and 5.3.5 immediately imply the following theorem.

Theorem 5.3.2 *The problem of determining whether an HRS instance admits a stable matching is NP-complete, even if the size of each resident and the capacity of each hospital is at most two, and the lengths of the residents' and hospitals' preference lists are at most three (these conditions holding simultaneously).*

The following corollary follows immediately from Theorem 5.3.2 and Lemma 5.2.1.

Corollary 5.3.2 *The problem of determining whether an HRCC instance admits a stable matching is NP-complete, even if the individual preference list of each resident and the joint preference list of each couple has at most three entries, and the capacity of each hospital is at most two (these conditions holding simultaneously).*

5.4 HRCC under classical (Gale-Shapley) stability

In this section we introduce a variant of HRCC in which stability is defined in terms of the classical (Gale-Shapley) stability. We provide a linear time algorithm for this problem, without any assumptions about the lengths of the preference lists or capacities of the hospitals. The problem is defined in the same manner as HRCC, the difference, however, lies in the definition of a blocking pair. So, as before, each hospital $h_j \in H$ provides a preference list of acceptable residents, denoted $\mathcal{L}_{h_j}$, and each resident $r_i \in R$ (whether they are a member of a couple or not) submits an individual preference list $\mathcal{L}_{r_i}$ of acceptable hospitals. Each couple c_k then constructs a joint preference list $\mathcal{L}_{c_k}$ that is *consistent* as defined in Section 5.2. A blocking pair for a matching is defined to be a (resident,hospital) pair (r_i, h_j) such that (i) r_i is unmatched, or, according to $\mathcal{L}_{r_i}$, r_i prefers h_j to $M(r_i)$ and (ii) either h_j is undersubscribed, or according to $\mathcal{L}_{h_j}$, h_j prefers r_i to at least one member of $M(h_j)$. Notice the difference in the stability definition to that defined in Section 2 for HRC is that blocking pairs are defined with respect to the individual preference lists, rather than the couples' joint preference lists. We partition the set of preference lists $\mathcal{L}$ of an instance of HRCC into three sets $\mathcal{L} = \mathcal{L}^C \cup \mathcal{L}^R \cup \mathcal{L}^H$ where $\mathcal{L}^R$ is the set of individual preference lists of the residents, $\mathcal{L}^H$ is the set of hospitals' preference lists, and $\mathcal{L}^C$ is the set of joint lists created by the couples. The goal in this setting is to find a matching satisfying the following two criteria:

1. The matching contains no blocking pairs relative to $\mathcal{L}^R$ and $\mathcal{L}^H$ under the classical notion of Gale-Shapley stability as defined above.
2. Each couple $c_k = (r_i, r_j)$ is assigned to a pair of hospitals $(h_p, h_q) \in \mathcal{L}_{c_k}$ or both r_i and r_j are unassigned.

The instance induced by the preference lists $\mathcal{L}^R$ and $\mathcal{L}^H$ is a classical Hospitals / Residents instance, so finding a matching satisfying (1) above simply involves using the extended Gale-Shapley algorithm to compute a stable matching M. Of course, M may not satisfy (2), in which case we need to determine if there is a different stable matching which does. Henceforth, let such a stable matching be called a *feasible stable matching*.

Given an instance I of HRCC, let $\mathcal{M}$ denote the set of all stable matchings under classical stability with respect to $\mathcal{L}^R$ and $\mathcal{L}^H$. We shall obtain a polynomial-time algorithm for determining the existence of a feasible stable matching by exploiting the known results on the rich structure of $\mathcal{M}$,

which we discussed in detail in Section 2.2.6. We briefly recall here the necessary structural results, beginning with the *Rural Hospitals theorem* [88, 31] (see also [36, Section 1.6.5]).

Theorem 5.4.1 *For any given HR instance, (i) each hospital is assigned the same number of residents in all stable matchings; (ii) exactly the same set of residents are unassigned in all stable matchings; (iii) any hospital which is undersubscribed in one stable matching is matched with precisely the same set of residents in all stable matchings.*

Part (ii) of Theorem 5.4.1 implies that if for some instance of the problem we have a stable matching in which one member of a couple is assigned and the other is unassigned, no feasible stable matching exists, for there is no stable matching in which they are either both assigned or both unassigned. By the same token, we cannot, in general, trivially obtain a feasible stable matching by forcing every resident in a couple simply to be unassigned. We continue with the following known relation which induces a partial order on $\mathcal{M}$ [36].

Definition 5.4.1 *Let M and M' be stable matchings for an HR instance. We say that M dominates M' (denoted $M \succeq M'$) if, for each assigned resident r, $M(r) = M'(r)$, or r strictly prefers $M(r)$ to $M'(r)$. Intuitively, M dominates M' if each resident is at least as happy in M as in M' (The case that $M \succeq M'$ and $M \neq M'$ is denoted by $M \succ M'$).*

Notice that a stable matching dominates itself. As we mentioned in Section 2.2.6, the pair $(\mathcal{M}, \succeq)$ actually forms a distributive lattice, with the maximum element being *resident-optimal*, in that every resident is matched to the most-preferred hospital he can ever obtain in any stable matching. It is this underlying structure of $\mathcal{M}$ that will allow us to develop the efficient algorithm presented in this section. We end our review of the structural results for $\mathcal{M}$ with the following fact.

Fact 5.4.1 *The resident-optimal stable matching M_R dominates all stable matchings in $\mathcal{M}$, and the hospital-optimal stable matching M_H is dominated by every stable matching.*

5.4.1 Breakmarriage

The algorithm we develop will use as a subroutine a generalised version of an algorithm known as Algorithm *Breakmarriage*, first defined by McVitie and Wilson [78] and used again by Gusfield

[34]. Our modified version of Algorithm Breakmarriage takes as input any stable matching $M \neq M_H$, and takes as a second parameter any resident r who is matched in M such that $M(r) \neq M_H(r)$, and always outputs a new stable matching dominated by M. Intuitively, M is the 'most dominant' stable matching subject to the constraint that r must be matched to a hospital further down his list. A description of this algorithm is as follows:

Breakmarriage(M, r)
Given the stable matching M and a matched resident r as input, let $R' \subseteq R$ denote the set of residents r' matched to $M(r) = h$ in M such that $r' = r$ or r' succeeds r on the preference list of h. Restart the extended Gale-Shapley algorithm (described in Section 2.2.6 by unassigning all pairs (r', h) for $r' \in R'$. All residents in R' are now free and are pushed onto a stack S in arbitrary order. Hospital h is defined to be "semi-free" in that it only accepts new proposals from residents it strictly prefers to r. Algorithm Breakmarriage iteratively pops a resident r'' from S with r'' proposing to the first hospital following $M(r'')$ on his preference list. This initiates a sequence of proposals, rejections, and acceptances as given by the resident-oriented Gale-Shapley algorithm [36, Section 1.6.3] in which free residents that have not been rejected by every hospital on their preference list are pushed onto S. Algorithm Breakmarriage terminates when S becomes empty. The current set of assignments M' is then output.

The following facts hold about Algorithm Breakmarriage(M, r).

Lemma 5.4.1 *Suppose M and M' are stable matchings such that $M \succeq M'$, and that resident r is matched in M and satisfies $M(r) \neq M'(r)$. Then, in the execution of Algorithm Breakmarriage(M, r), no resident $r' \in R$ ever proposes to a hospital succeeding $M'(r')$ on his preference list (in the case that $M(r') = M'(r')$, this implies that r' remains matched to $M(r')$ in the execution of Breakmarriage(M, r)).*

Proof Let $h = M(r)$, and let $R' \subseteq R$ denote the set of residents $r' \in M(h)$ such that $r' = r$ or r' succeeds r on the preference list of h. Since $r \notin M'(h)$ and $M \succeq M'$, it follows that r prefers h to $M'(r)$. Hence h is full in M' and prefers each of its assignees in M' to r. It follows that h prefers each of its assignees in M' to each member of R'.

Now suppose that a resident r' who is matched in M is the first resident in the execution of Algorithm Breakmarriage(M, r) to be rejected by the hospital he is assigned to in M'. Let $h' = M'(r')$.

Clearly if $h' = h$ and $r' \in R'$ then $M(r') = M'(r')$, which is impossible by the conclusion of the previous paragraph. Hence r' was rejected by h' during the phase of Algorithm Breakmarriage(M, r) that corresponds to the restart of the resident-oriented Gale-Shapley algorithm. Let M_A be the matching at the point during the execution of the algorithm when h' rejected r'. Then h' is full in M_A and prefers each of its assignees in $M_A(h')$ to r'. Since $r' \in M'(h') \setminus M_A(h')$ and h' is full in M_A, it follows that there exists some $r_w \in M_A(h') \setminus M'(h')$.

If r_w is matched in M', then r_w cannot yet have proposed to $M'(r_w)$ (as $h' \neq M'(r_w)$, and hence this would contradict the fact that r' is the first resident to be rejected by the hospital that he is assigned to in M'). Hence either r_w is unmatched in M' and finds h' acceptable, or r_w prefers h' to $M'(r_w)$. But this implies that (r_w, h') forms a blocking pair for M', as h' prefers r_w to r'. Therefore, r' is not rejected by h' in the call to Algorithm Breakmarriage(M, r). □

Corollary 5.4.1 *Let $M \neq M_H$ be a stable matching and r an arbitrary resident with $M(r) \neq M_H(r)$. Then, no resident $r' \in R$ is ever rejected by $M_H(r')$ in a call to Algorithm Breakmarriage(M, r).*

Lemma 5.4.2 *When Algorithm Breakmarriage(M, r) terminates, the set of assignments M' output by the algorithm is a stable matching.*

Proof We first observe that $M(r)$ is full in M, by Theorem 5.4.1, since $M(r) \neq M_H(r)$. We proceed by showing that every hospital that is full in M is also full in M'. Throughout the execution of Algorithm Breakmarriage(M, r), no hospital that was full in M can become undersubscribed except for $M(r)$, as no other hospital rejects a resident without gaining a better one. Therefore, the hospitals that are undersubscribed at some point in the execution of the algorithm are those that are undersubscribed in every stable matching (by Theorem 5.4.1) and $M(r)$. Suppose that during the algorithm's execution a resident r' were to propose to a hospital h', such that h' is undersubscribed in M. By Theorem 5.4.1, h' is undersubscribed in M_H, and also $(r', h') \notin M_H$, since $(r', h') \notin M$. If r' is unmatched in M_H then (r', h') blocks M_H, a contradiction. Hence r' is matched in M_H. As $h' \neq M_H(r')$, by Corollary 5.4.1, h' precedes $M_H(r')$ on r''s preference list, implying that (r', h') blocks M_H, a contradiction. Hence, no resident may propose to a hospital that is undersubscribed in M at any point in the execution of the algorithm, implying that proposals are only made to those hospitals that are full in M. It follows by a simple counting argument that $M(r)$ is full in M'.

The stability of M' follows from the fact that a resident r has proposed to, and been rejected by,

any hospital h it prefers over $M(r)$. Since hospitals continually improve throughout the course of the algorithm, r cannot be a part of a blocking pair. □

Lemma 5.4.3 *Let M and M' be distinct stable matchings, and suppose that M dominates M'. Then, if r is a resident with $M(r) \neq M'(r)$, Algorithm Breakmarriage(M, r) either returns M' or a stable matching M'' that dominates M' (i.e., $M \succ M'' \succeq M'$).*

Proof This is an immediate consequence of Lemmas 5.4.1 and 5.4.2. □

Lemma 5.4.4 *Any stable matching M can be obtained by a series of calls to Algorithm Breakmarriage from the resident-optimal stable matching M_R in $O(m)$ time, where m is the sum of the lengths of the preference lists.*

Proof The fact that an arbitrary stable matching M can be obtained from M_R by a series of calls to Algorithm Breakmarriage is a straightforward consequence of Lemma 5.4.3. A total of $O(m)$ time is spent, as any arbitrary series of calls to the algorithm constitutes at most one left to right traversal of each resident's preference list, and similar time is spent traversing the hospitals' preference lists. □

So, in light of Lemma $5.4.4$, we can see that the computation of a feasible stable matching (if one exists) can be achieved by first finding the resident-optimal stable matching M_R (which, in general, need not be feasible), and making a suitable selection of calls to Algorithm Breakmarriage. In the next subsection, we will show that because the preference lists in $\mathcal{L}_{c_k}$ are consistent, we can always compute an appropriate sequence of calls to Algorithm Breakmarriage in linear time.

We also note that repeated calls to Algorithm Breakmarriage ultimately yield the hospital-optimal stable matching, in which every resident is, of course, assigned to $M_H(r)$. Thus this is the only stable matching Algorithm Breakmarriage cannot take as input.

5.4.2 The algorithm

Before presenting the main algorithm of this section, we require some preliminary lemmas and definitions. Let M be a stable matching. Recall Theorem 5.4.1, which states that precisely the

$\mathcal{L}_{r_1}$: h_1 h_2 $\underline{h_3}$ h_5
$\mathcal{L}_{r_2}$: h_1 $\underline{h_2}$ h_3
$\mathcal{L}_{r_3}$: h_4 $\underline{h_3}$ h_2
$\mathcal{L}_{r_4}$: h_3 $\underline{h_4}$
$\mathcal{L}_{r_5}$: $\underline{h_4}$ h_2 h_1
$\mathcal{L}_{r_6}$: h_3 h_1 $\underline{h_5}$
$\mathcal{L}_{r_7}$: $\underline{h_2}$ h_3 h_4 h_1
$\mathcal{L}_{r_8}$: $\underline{h_1}$ h_2 h_4

1 : $\mathcal{L}_{h_1}$: $\underline{r_8}$ r_5 r_1 r_7 r_6 r_2
2 : $\mathcal{L}_{h_2}$: r_3 r_8 $\underline{r_2}$ r_5 $\underline{r_7}$ r_1
2 : $\mathcal{L}_{h_3}$: r_7 $\underline{r_1}$ r_2 $\underline{r_3}$ r_6 r_4
2 : $\mathcal{L}_{h_4}$: r_7 r_8 $\underline{r_4}$ $\underline{r_5}$ r_3
1 : $\mathcal{L}_{h_5}$: $\underline{r_6}$ r_1

$\mathcal{L}_{(r_2,r_3)}$: (h_1,h_3) (h_3,h_2)
$\mathcal{L}_{(r_4,r_5)}$: (h_4,h_4) (h_4,h_2) (h_4,h_1)
$\mathcal{L}_{(r_7,r_8)}$: (h_2,h_2) (h_4,h_4)

Figure 5.3: An HRCC instance with a stable but not feasible matching

same set of residents are matched in every stable matching. With this in mind, we define a *matched couple* $c_k = (r_i, r_j)$ to be a couple such that r_i and r_j are matched in M (and hence in every stable matching). Similarly, we define an *unmatched couple* $c_k = (r_i, r_j)$ to be a couple such that one or both of r_i and r_j are unmatched in M (and hence in every stable matching). Let $c_k = (r_i, r_j)$ be a matched couple. We define the *next acceptable pair* on $\mathcal{L}_{c_k}$ (denoted $next_M(c_k)$) to be the first pair of hospitals (h_p, h_q) on $\mathcal{L}_{c_k}$ such that h_p succeeds or is equal to $M(r_i)$ on $\mathcal{L}_{r_i}$ and h_q succeeds or is equal to $M(r_j)$ on $\mathcal{L}_{r_j}$. If no such pair exists, we say $next_M(c_k) = \emptyset$ with slight abuse of notation.

Example To illustrate the notion of the next acceptable pair for a couple, we refer the reader to Figure 5.3. This shows an HRCC instance with 8 residents $r_1, r_2, \ldots, r_8$ and 5 hospitals $h_1, h_2, \ldots, h_5$. There are three couples, namely (r_2, r_3), (r_4, r_5), and (r_7, r_8). A stable (but not feasible) matching M for this instance is denoted by underlining. In M, $next_M(r_2, r_3) = (h_3, h_2)$, $next_M(r_4, r_5) = (h_4, h_4)$, and $next_M(r_7, r_8) = (h_2, h_2)$.

The next two lemmas will help us to develop the algorithm to determine an appropriate sequence of calls to Algorithm Breakmarriage to obtain a feasible stable matching, if one exists.

Lemma 5.4.5 *Let M be a non-feasible stable matching that dominates a feasible stable matching M_f. Let $c_k = (r_i, r_j)$ be any matched couple who are not matched to a pair of hospitals on $\mathcal{L}_{c_k}$. Then, in M_f, (r_i, r_j) is either assigned to $next_M(c_k)$ or a pair of hospitals succeeding $next_M(c_k)$ on $\mathcal{L}_{c_k}$.*

Proof Since M dominates M_f, for each resident r either $M(r) = M_f(r)$ or $M_f(r)$ succeeds $M(r)$ on $\mathcal{L}_r$ by Definition 5.4.1. So, for a couple $c_k = (r_i, r_j)$, their partners in M_f are either their current hospitals or hospitals that appear further down their individual preference lists. It follows that $next_M(c_k)$ is the first pair of hospitals on $\mathcal{L}_{c_k}$ that c_k could be assigned to in M_f. So, in M_f, c_k is either assigned to $next_M(c_k)$ or to a pair of hospitals that succeeds $next_M(c_k)$ on $\mathcal{L}_{c_k}$. □

Lemma 5.4.6 *Let M be a non-feasible stable matching that dominates a feasible stable matching M_f. Let $c_k = (r_i, r_j)$ be any matched couple who are not matched to a pair of hospitals on $\mathcal{L}_{c_k}$. Let $(h_p, h_q) = next_M(c_k)$. Then,*

1. *Either $M(r_i) \neq h_p$ or $M(r_j) \neq h_q$ (or both).*
2. *The stable matching obtained by calling Algorithm Breakmarriage with M and r^* dominates M_f, where r^* is r_i if $M(r_i) \neq h_p$ and is otherwise r_j.*

Proof For the first claim, r_i and r_j cannot both be assigned to h_p and h_q, respectively in M_f, for M_f is feasible, and this pair does not appear on $\mathcal{L}_{c_k}$, by the assumption of the lemma.

For the second claim, since M is not feasible and M dominates M_f, the members of c_k must be assigned to $next_M(c_k)$ or to a pair of hospitals succeeding $next_M(c_k)$ by Lemma 5.4.5. By the nature of the construction of the joint preference list $\mathcal{L}_{c_k}$, and by the fact that M dominates M_f, this implies that r^* is assigned to a hospital succeeding $M(r^*)$ on $\mathcal{L}_{r^*}$ in M_f. Hence by Lemma 5.4.3, calling Algorithm Breakmarriage on the current matching and r^* yields a stable matching that dominates M_f. □

We are now ready to describe the algorithm for finding a feasible stable matching or reporting that none exists. The algorithm begins by computing the resident-optimal stable matching M_R and the hospital-optimal stable matching M_H. By Lemma 5.4.1, M_R dominates all stable matchings in $\mathcal{M}$ – hence it dominates every feasible stable matching (if any exist) as well. If M_R is itself feasible, the algorithm returns M_R. Otherwise, if for any couple (r_i, r_j) it is the case that r_i is assigned and r_j is unassigned, the algorithm halts, correctly reporting that no feasible stable matching exists by Theorem 5.4.1.

Only if no such couple exists do we enter the while loop which maintains the loop condition that the current matching M is not feasible – hence there is some couple $c_k = (r_i, r_j)$ who are not assigned

Compute M_R and M_H

$M \leftarrow M_R$

for each couple $c \in C$:

  if one member of c is assigned in M and the other is unassigned in M:

    report "no feasible stable matching exists"

    HALT

while some couple $c_k = (r_i, r_j)$ is not assigned a pair from $\mathcal{L}_{c_k}$ in M:

  if c_k has $next_M(c_k) = \emptyset$:

    report "no feasible stable matching exists"

    HALT

  $r^* \leftarrow$ a resident in c_k with different partners in M and $next_M(c_k)$

  if $M(r^*) = M_H(r^*)$:

    report "no feasible stable matching exists"

    HALT

  else:

    $M \leftarrow$ Breakmarriage(M, r^*)

return M

Figure 5.4: Algorithm HRCC

to a hospital on $\mathcal{L}_{c_k}$. If $next_M(c_k) = \emptyset$, the algorithm reports failure. Otherwise the algorithm identifies a resident $r^* \in \{r_i, r_j\}$ such that $M(r^*)$ is not equal to r^*'s partner in $next_M(c_k)$. If r^* has the same partner in M and in M_H, the algorithm reports failure. Otherwise, we call Algorithm Breakmarriage with M and r^*. The loop is exited only when the algorithm reports failure or when the current matching M is feasible. The pseudocode for the algorithm is presented in Figure 5.4 as Algorithm HRCC.

5.4.3 Correctness

Suppose that no feasible stable matching exists. Then, Algorithm HRCC will clearly correctly output "no feasible stable matching exists" in one of three places. If, before entering the while loop, there is a couple with one member assigned and the other unassigned, the algorithm correctly halts. Otherwise, the algorithm enters the while loop, and since no feasible stable matching exists,

the algorithm continues to make calls to Algorithm Breakmarriage. This process must eventually halt, either when $next_M(c_k) = \emptyset$ for some couple c_k, or when the successive calls to Algorithm Breakmarriage eventually yield M_H, at either point Algorithm HRCC will correctly output the failure message.

So, instead, let us suppose a feasible stable matching M_f does exist. We claim that Algorithm HRCC maintains the invariant that at each iteration of the while loop the current matching M dominates M_f. The claim is clearly true when $M = M_R$, by Lemma 5.4.1, so let us assume the invariant is true at the end of the i^{th} iteration. Let M_i denote the stable matching at the end of iteration i of the while loop and suppose that M_i is not feasible. Since M_i is not feasible, there is some assigned couple $c_k = (r_s, r_t)$ that is not assigned a pair from $\mathcal{L}_{c_k}$. By Lemma 5.4.6, there is at least one resident $r^* \in \{r_s, r_t\}$ that is not assigned to a hospital in the ordered pair $next_{M_i}(c_k)$, and, further, calling Algorithm Breakmarriage on the current matching M_i and r^* yields a stable matching that dominates M_f. Thus the matching M_{i+1} obtained by this process dominates M_f, and the claim follows.

Thus, at each iteration of the while loop of Algorithm HRCC, the current matching dominates M_f. Hence, the algorithm eventually terminates having encountered M_f or a different feasible stable matching that dominates M_f. Let M_f^* denote the feasible stable matching that is returned by the algorithm. Since M_f is an arbitrary feasible stable matching, we have argued that M_f^* dominates *every* feasible stable matching. Hence, M_f^* is *resident-optimal* amongst the set of feasible stable matchings.

We summarise this section with the following theorem.

Theorem 5.4.2 *Algorithm HRCC finds the resident-optimal feasible stable matching M_f^* if one exists or reports "none exists" in $O(m)$ time, where m is the sum of the lengths of the preference lists of the input.*

Proof We have shown that the algorithm finds the resident-optimal feasible stable matching if it exists, or reports "none exists" correctly. To establish the claimed running time, we observe that the algorithm constitutes essentially a "left to right" sweep of the residents' preference lists. So, by using appropriate data structures (extending those described in [36, Section 1.2.3] for the Extended Gale-Shapley algorithm for SMI to the HR case), we can implement this algorithm to run in $O(m)$

time. □

We end this section with the remark that HRCC under classical stability is a variant of HR that can be solved in polynomial time by a unified approach [19] since this problem exhibits the so-called *independence property* (see [19] for the definition of this property and further details). For completeness and for consistency with the notation and terminology adopted in the remainder of this chapter, we have chosen to present the main result of this section as a standalone algorithm.

5.5 HRS with hospital preference lists of length ≤ 2

In light of the NP-completeness result for HRS presented in Section 5.3, it is natural to ask if, by specialising the problem version, we can identify a "boundary" at which HRS becomes polynomial-time solvable. One option for us to consider is to allow the sizes of the residents to be at most one, rather than two. This restriction would, of course, yield an instance of the classical Hospitals / Residents problem, which is polynomial-time solvable. A different option is to further restrict the lengths of the preference lists for the residents and/or the hospitals. We show that by restricting the length of the preference list of each hospital to be at most two, rather than three, a stable matching always exists, and an extension of the Gale-Shapley algorithm finds a stable matching in polynomial time, even if no restriction is placed on the sizes of the residents, the lengths of the preference lists of the residents, or the capacities of the hospitals. Since NP-completeness for HRS holds even for hospital preference lists of length at most three, the results of this section indicate such a boundary for HRS. We describe the restricted version of HRS in which the lengths of the hospitals' preference lists are at most two and the residents' lists are unbounded as $(\infty, 2)$-HRS.

The procedure for solving $(\infty, 2)$-HRS is as follows. The algorithm can be seen as a sequence of "proposal" operations from the residents to the hospitals. A resident proposes sequentially to each hospital on his list until he becomes assigned or his list becomes empty. When a resident r_i proposes to a hospital h_j, r_i becomes provisionally assigned to h_j. If r_i is that hospital's first choice, and h_j's preference list has another entry, we let r_k denote h_j's second choice. If $s_i + s_k > c_j$, the pair (r_k, h_j) is deleted, meaning that r_k is removed from h_j's preference list, and h_j is removed from r_k's preference list. This is the only time a (resident,hospital) pair is deleted by the algorithm. The algorithm continues this process until each resident is either assigned a hospital or has an empty list. The details of the algorithm are shown in Figure 5.5.

```
assign all residents to be free
while some resident r_i is free and r_i has a nonempty list:
    h_j ← first hospital on r_i's list
    if r_i is h_j's first choice and h_j's list is of length 2:
        r_k ← h_j's second choice
        if s_i + s_k > c_j:
            if r_k is assigned to h_j:
                unassign r_k
            delete (r_k, h_j)
    assign r_i to h_j
```

Figure 5.5: Algorithm $(\infty, 2)$-HRS

Let us now establish the correctness and time complexity of the algorithm presented.

Theorem 5.5.1 *Algorithm* $(\infty, 2)$*-HRS finds the resident-optimal stable matching for an instance of* $(\infty, 2)$*-HRS in* $O(m)$ *time, where* m *is the sum of the lengths of the preference lists.*

Proof It is clear that the provisional assignments at the termination of Algorithm $(\infty, 2)$-HRS form a matching M. We claim that M is stable. To see this, consider an arbitrary resident r_i who is unassigned or prefers a hospital h_j to his assignment in M. Then, since r_i is not assigned to h_j in M and prefers h_j to his current assignment, h_j must have been deleted from r_i's preference list. But this can only happen if r_i is h_j's second choice and h_j was assigned to its first choice at some point in the algorithm and does not have enough spare capacity to accommodate r_i. But h_j's first choice can never become unassigned from h_j at any subsequent step of the algorithm – so in fact r_i cannot block with h_j in M. Since r_i was chosen arbitrarily it follows that no resident is part of a blocking pair in M.

Secondly, we claim that Algorithm $(\infty, 2)$-HRS never deletes a stable pair (i.e., a (resident,hospital) pair that belongs to some stable matching). For, suppose that (r_k, h_j) is the first such pair deleted during an arbitrary execution of the algorithm, and let M' be a stable matching containing (r_k, h_j). Then r_k was deleted because the resident r_i preceding r_k on h_j's preference list became assigned to h_j and $s_i + s_k > c_j$. Now, since no stable pair has been deleted prior to this point, in M', r_i is either assigned to h_j or to a hospital lower than h_j, or is unassigned. Since r_i and r_k cannot both be assigned to h_j in M', it follows that (r_i, h_j) blocks M', a contradiction.

Thus we have shown that M is stable and that each resident is assigned to his optimal partner in M. Let us now show that a $O(m)$ implementation is easily achieved with the use of simple data structures. If we maintain a stack S of free residents, then each iteration of the loop involves a subset of the following operations: (i) pop a resident r_i off of, S (ii) examining the first entry h_j of r_i's list, (iii) examining the length of h_j's list (it is either one or two), (iv) simple comparisons and arithmetic, (v) assigning and/or unassigning at most two residents to h_j, (vi) deleting the first entry of a resident's list, (vii) pushing a resident onto S. If each preference list is stored as a linked list, each of these operations clearly can be performed in $O(1)$ time, and thus a single iteration of the loop takes $O(1)$ time. Since each resident proposes to each hospital on his list at most once, the number of iterations of the loop is $O(m)$, and therefore the running time of the algorithm is $O(m)$. □

5.6 Conclusion and open problems

Our stability definition for HRS allows a resident r_i to displace a group of inferior residents of a given total size, so long as this frees up enough space for r_i. This could, of course, include a situation whereby a resident of size ten is displaced in order to make way for a resident of size one, for example. Our definition assumes that the quality of the assignees takes precedence over the size. However it may be the case that a hospital's primary concern is to ensure that its occupancy is as high as possible. Thus it would not participate in a blocking pair if its occupancy were to be reduced as a result of rejecting the inferior residents and taking on the new resident. This gives rise to an alternative stability definition which is obtained from the one given for HRS in Section 5.2 by modifying Condition (2) as follows:

2. $O_j^M + s_i \leq c_j$, or h_j prefers r_i to residents $r_{k_1}, \ldots r_{k_t} \in M(h_j)$ such that

$$s_i \geq \sum_{p=1}^{t} s_{k_p} \quad \text{and} \quad O_j^M + s_i - \sum_{p=1}^{t} s_{k_p} \leq c_j.$$

It remains open to investigate the algorithmic complexity of the problem of finding a matching that satisfies this new version of stability, for a given HRS instance.

Chapter 6

Three dimensional stable matching

6.1 Introduction

Knuth [69] initiated the study of three dimensional stable matching problems by asking if the stable marriage problem could be extended to three sets, so that we have not only men and women, but a third set, which he called dogs. Knuth's question is (perhaps intentionally) somewhat open-ended. He did not suggest a new stability criterion or specify what the agent's preference lists would be like.

Over the years, a handful of researchers have explored three dimensional stable matching problems, in an effort to answer Knuth's question. In Section 2.2.8, we surveyed the relevant literature and known results regarding three dimensional stable matchings. In this chapter we are particularly interested in the study of the so-called *cyclic* three dimensional stable matching problem, in which men care about only the women, women care about only the dogs, and the dogs care about only the men. As an open question, Ng and Hirschberg [83] asked for a polynomial-time algorithm to find stable matchings in this setting (we define cyclic stable matchings and stability formally in Section 6.2). Boros et al [11] showed that if there are at most three agents in each set of men, women and dogs, then a stable matching always exists. Eriksson et al [26] proved that this also holds if there are four agents in each set and conjectured that a stable matching exists for every instance of cyclic 3DSM.

In this chapter we continue the study of cyclic three dimensional stable matching problems under

two natural definitions of stability given by Eriksson et al [26], called weak and strong stability, respectively. We describe a special instance of the three dimensional stable matching problem with incomplete lists called the 9-Sun, and show that it is particularly problematic. Specifically, we show that the 9-Sun admits no weakly stable matching, and use this instance as an instrumental gadget in showing that weakly stable matchings are NP-hard to find when agents are allowed to have incomplete lists, and that strongly stable matchings are also NP-hard to find, regardless of the length of the preference lists (complete or incomplete). For brevity we have chosen to omit the word 'cyclic' when referring to the cyclic three dimensional stable matching problem in this chapter.

6.2 Formal definitions

The *three-dimensional stable matching problem* (3DSM) consists of a set of n men, n women, and n dogs. Associated with each agent is a preference list which strictly ranks all of the members of one of the other sets. Specifically, each man has a strict preference list ranking all of the women, each woman has a strict preference list over all of the dogs, and every dog has a strict preference list over all of the men. When preference lists are allowed to be *incomplete*, so that an agent ranks only a subset of the appropriate set of agents, we obtain an instance of the *three-dimensional stable matching problem with incomplete lists* (3DSMI). In keeping with the common terminology of this thesis, if agent b appears on agent a's preference list, then a finds b *acceptable*. Notice that in the 3DSMI setting there is no analogous notion of *mutually acceptable pair* as there is in the stable marriage or stable roommates setting. An *acceptable triple* is a triple (m, w, d) such that m finds w acceptable, w finds d acceptable, and d finds m acceptable.

A matching M for a 3DSMI instance is a disjoint set of acceptable triples. If a triple (m, w, d) is in M, then we let $M(m) = w$, $M(w) = d$, and $M(d) = m$. $M(a)$ is undefined for an unmatched agent a.

There are at least two natural definitions of stability which arise in the context of 3DSMI. A matching M is said to be *weakly stable* if there is no *strongly blocking triple*, i.e. an acceptable triple $(m, w, d) \notin M$ such that (i) m is unmatched or prefers w to $M(w)$, (ii) w is unmatched or prefers d to $M(d)$, and (iii) d is unmatched or prefers m to $M(d)$. A matching is *strongly stable* if there is no *weakly blocking triple* $(m, w, d) \notin M$ such that (i) m is unmatched, $M(m) = w$ or m strictly prefers w to $M(m)$ (ii) w is unmatched, $M(w) = d$, or w strictly prefers d to $M(w)$, and (iii) d

is unmatched, $M(d) = m$, or d strictly prefers m to $M(d)$. Note that the definition of a weakly blocking triple crucially depends on the fact that $(m, w, d) \notin M$. Hence at least one agent strictly improves in a weakly blocking triple, while the other two agents are at least as happy. Observe that every strongly stable matching is also weakly stable, hence if no weakly stable matching exists for an instance I, then no strongly stable matching can exist for I either.

The underlying directed graph $D_I = (V, A)$ of an instance I of 3DSMI consists of a vertex for each agent of I, and a directed arc (a, b) for each pair of agents (a, b) such that a finds b acceptable. Clearly, a matching for I corresponds to a disjoint subset of directed 3-cycles in D_I. It is sometimes convenient for us to think of matchings in this graph-theoretic context, so we sometimes refer to a matching as a set of disjoint *cycles* (rather than triples). We also occasionally refer to an entry on a preference list as an *edge*, or refer to an agent as a *vertex*. The meaning should always be clear from the context.

We use the notation $\mathcal{L}_a$ to denote the preference list of an agent a, and $\mathcal{L}_a^i$ to denote the i^{th} entry of agent a's preference list.

6.3 The 9-Sun: a problematic subgraph

At the core of all the results in this chapter is a particular 3DSMI instance that admits no weakly stable matching (and thus no strongly stable matching). The preference lists and underlying directed graph of this instance are given in Figure 6.1. The thickness of the arcs of the directed graph illustrate the preference of a given vertex in that thicker arcs represent higher preferences. For obvious reasons, we refer to this 3DSMI instance as the 9-Sun (denoted S_9) . Note that, rather paradoxically, the 9-Sun has six agents of each kind, hence 18 agents in total.

$m_1 : w_1 w_1'$	$w_1 : d_1 d_1'$	$d_1 : m_2 m_2'$
$m_2 : w_2 w_2'$	$w_2 : d_2 d_2'$	$d_2 : m_3 m_3'$
$m_3 : w_3 w_3'$	$w_3 : d_3 d_3'$	$d_3 : m_1 m_1'$
$m_1' : w_3$	$w_1' : d_3$	$d_1' : m_1$
$m_2' : w_1$	$w_2' : d_1$	$d_2' : m_2$
$m_3' : w_2$	$w_3' : d_2$	$d_3' : m_3$

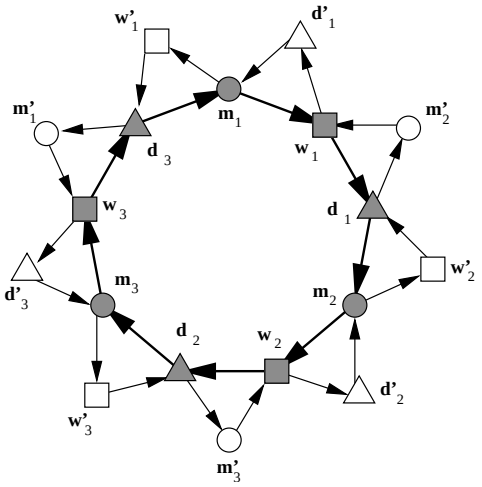

Figure 6.1: The 9-Sun

We refer to the agents $\{m_i, w_i, d_i : 1 \leq i \leq 3\}$ as the *inner agents* of S_9 and the agents $\{m_i', w_i', d_i' : 1 \leq i \leq 3\}$ as the *outer agents* of S_9. In what follows we will show that the 9-Sun admits no weakly stable matching, and, moreover, that the preference lists of the 9-Sun can be completed to a 3DSM instance which admits no strongly stable matching. These observations are crucially important in the NP-hardness results of the subsequent sections of this chapter, as 9-Suns play a vital role in the reductions.

Lemma 6.3.1 *The 9-Sun admits no weakly stable matching.*

Proof By inspection of the underlying graph of the 9-Sun, we can observe that the only acceptable triples are of the form (m_i, w_i', d_{i-1}), (m_i, w_i, d_i') and (m_i', w_{i-1}, d_{i-1}), so that any acceptable triple contains exactly two inner agents. In any matching M, at least one inner agent is unmatched. By the symmetry of the instance we may suppose, without loss of generality, that this unmatched inner agent is m_1. Then, the triple (m_1, w_1', d_3) is a blocking triple for M. □

For an inner agent a_i, we let $S_9 \setminus a_i$ denote the 3DSMI instance obtained by removing a_i (and all incident edges). In the following lemma, we show that the instance obtained by removing an arbitrary inner agent from the 9-Sun does, in fact, admit a (unique) weakly stable matching.

Lemma 6.3.2 *Let a_i be an inner agent of the 9-Sun. Then, the instance obtained by removing a_i ($S_9 \setminus a_i$) admits a unique weakly stable matching M_u. Moreover, M_u is strongly stable.*

Proof Suppose without loss of generality that the inner agent m_1 is removed. Then, $M_u = \{(m'_2,\ w_1,\ d_1),\ (m_2,\ w_2,\ d'_2),\ (m_3,\ w'_3,\ d_2),\ (m'_1,\ w_3,\ d_3)\}$ is a weakly stable matching. To see that M_u is the unique weakly stable matching, notice that the triple (m'_1, w_3, d_3) must be in any stable matching M', otherwise d_3 and m'_1 are unmatched, and form a blocking triple with w_3. This implies that the triple (m_3, w'_3, d_2) must be in M' as well, for otherwise m_3 and w'_3 are unmatched, and will form a blocking triple with d_2. This argument continues in such a way that M' must necessarily contain (m_2, w_2, d'_2) and (m'_2, w_1, d_1). Hence $M' = M_u$.

Verifying that M_u is also strongly stable is a trivial task achieved by inspecting the preference lists. □

For a given agent a_i in the 9-Sun, we denote the unique weakly stable matching made possible by removing a_i by $M_{S_9 \setminus a_i}$.

Corollary 6.3.1 *$M_{S_9 \setminus a_i}$ is the only weakly or strongly stable matching for $S_9 \setminus a_i$.*

We next show that by completing the preference lists of the 9-Sun in an arbitrary way (so that each man ranks every woman, every woman ranks every dog, etc), the resulting instance of 3DSM, denoted by $\overline{S_9}$, does not admit any strongly stable matching. For ease of exposition, we call the triples of S_9 *original triples*.

Lemma 6.3.3 *The instance $\overline{S_9}$ of cyclic 3DSM admits no strongly stable matching.*

Proof Suppose, for a contradiction, that M is a strongly stable matching. As the nine inner agents form a 9-cycle in the underlying directed graph, the nine original triples have a natural cyclic order. We show that if an arbitrary original triple, say (m_1, w_1, d'_1), is not in M, then the "successor" original triple (m'_2, w_1, d_1) must be in M, which would imply a contradiction given that the number of these original triples is odd. To this end, suppose without loss of generality that $(m_1, w_1, d'_1) \notin M$. Then, $M(w_1) = d_1$, for otherwise (m_1, w_1, d'_1) would be weakly blocking. Similarly, $(m'_2, w_1, d_1) \notin M$ implies $M(d_1) = m_2$. But this means that $(m_2, w_1, d_1) \in M$, so (m_2, w'_2, d_1) is weakly blocking. □

Let $\overline{S_9} \setminus a_i$ denote the instance created by removing an inner agent a_i from $\overline{S_9}$. Consider the unique strongly stable matching $M_{S_9 \setminus a_i}$ for $S_9 \setminus a_i$. The next fact we wish to establish is that $M_{S_9 \setminus a_i}$ is in fact the only strongly stable matching for $\overline{S_9} \setminus a_i$.

Lemma 6.3.4 *Let M be a matching for $\overline{S_9} \setminus a_i$ such that $M \neq M_{S_9 \setminus a_i}$. Then, M is not strongly stable.*

Proof Suppose that M is a matching of $\overline{S_9} \setminus a_i$. As in the proof of Lemma 6.3.3, we use the fact that if an arbitrary original triple is not in M, then the successor original triple is either in M, or is weakly blocking. Therefore, if we do not include four of the seven original triples of $\overline{S_9} \setminus a_i$ in a matching then one of them would be weakly blocking. There is only one way to select four of the seven original triples of $\overline{S_9} \setminus a_i$, hence the lemma. □

6.4 NP-completeness of 3DSMI under weak-stability

6.4.1 The reduction

This section describes the polynomial-time reduction that establishes NP-completeness for the problem of deciding whether a weakly stable matching exists for an arbitrary 3DSMI instance. The reduction is from a restricted variant of MAX-SMTI (see Section 2.2.5 and Chapter 3 for background and definitions) called $(3, 3)$-COM-SMTI. We used this same starting point for a reduction presented in Section 5.3. To make this chapter self-contained, define $(3, 3)$-COM-SMTI to be the problem of deciding whether a complete stable matching exists (i.e., a stable matching that matches every agent), given an instance of SMTI in which each preference list is of length at most three, every woman's preference list is strictly ordered, and each man's preference list is either strictly ordered or is a tie of length two (all of these conditions holding simultaneously). In the Appendix we prove that $(3, 3)$-COM-SMTI is NP-complete. Of course, this hardness result holds if the roles of the men and women are reversed, which, for convenience, we assume in the following reduction. The remainder of this section is devoted to describing a polynomial-time reduction from $(3, 3)$-COM-SMTI to cyclic 3DSMI.

Given an arbitrary $(3, 3)$-COM-SMTI instance I, the *underlying graph* $G = (A \cup B, E)$ of I consists of the set $A = \{a_1, a_2, \ldots, a_n\}$ of men a_i, all of whom have strictly ordered preference

lists, and the set B of women which is partitioned into two sets $B_1 \cup B_2 = \{b_1,\ldots, b_{n_1}\} \cup \{b_1^T,\ldots, b_{n_2}^T\}$ where $n_1 + n_2 = n$. Each woman $b_j \in B_1$ has a strictly ordered preference list, and each woman $b_j^T \in B_2$ has a preference list consisting solely of a tie of length two. We denote a woman who may be a member of either B_1 or B_2 by $b_i^{(T)}$.

Let I be an instance of $(3,3)$-COM-SMTI with the underlying graph $G = (A \cup B, E)$. We construct an instance I' of cyclic 3DSMI which initially consists of the sets Q, R, and S of men, women, and dogs, respectively, as follows.

Step 1: the proper part

The sets of men and women of I' we create in this step are in direct correspondence to the men and women in I. The dogs of I' are created to capture the preference lists of the women of I.

Create men $Q = \{m_1,\ldots, m_n\}$ and women $R = W_1 \cup W_2 = \{w_1,\ldots, w_{n_1}\} \cup \{w_1^T,\ldots, w_{n_2}^T\}$. The set of dogs of I' consists of two parts $S_1 \cup S_2 = S$, defined by creating a dog $d_{j,i}$ in S_1 for each i such that $a_i \in \mathcal{L}_{b_j}$ $(1 \leq j \leq n_1)$, and creating dogs d_j^T $(1 \leq j \leq n_2)$ in S_2.

Recall that $\mathcal{L}_a^i$ denotes the i^{th} entry on agent a's preference list. A tie in the preference list of an agent (in the given instance I) is indicated by parentheses. The (strictly ordered) preference lists of the agents in Q, R, and S are constructed by the following cases:

1. If $\mathcal{L}_{a_i}^l = b_j^{(T)}$ then let $\mathcal{L}_{m_i}^l = w_j^{(T)}$ $(1 \leq l \leq r$, where r is the length of a_i's list).
2. If $\mathcal{L}_{b_j}^l = a_i$ then let $\mathcal{L}_{w_j}^l = d_{j,i}$ and $\mathcal{L}_{d_{j,i}} = m_i$ $(1 \leq l \leq r$, where r is the length of b_j's list).
3. If $\mathcal{L}_{b_j^T} = (a_p, a_q)$ then let $\mathcal{L}_{w_j^T} = d_j^T$ and $\mathcal{L}_{d_j^T} = m_p\ m_q$ (in arbitrary order).

The collection of agents and preferences created in this step of the reduction is the *proper part* of the instance.

Step 2: the additional part (add 9-Suns)

We construct the *additional part* of I' by creating n ($|Q|$) copies of S_9. The t^{th} copy of S_9 (denoted S_9^t consists of the inner agents $\{m_{t_i}, w_{t_i}, d_{t_i} : 1 \leq i \leq 3\}$ and outer agents $\{m'_{t_i}, w'_{t_i}, d'_{t_i} : 1 \leq$

$i \leq 3\}$ with preference lists as described in Figure 6.1. We add these n copies of S_9 to I' in the following way. Replace the inner agent m_{t_1} in S_9^t with man $m_t \in Q$ by replacing each occurrence of m_{t_1} in the preference lists of each agent in S_9^t with m_t. Also, let m_{t_1}'s acceptable partners in S_9^t, namely w_{t_1} and w'_{t_1} be appended in this order to the end of m_t's list. The final preference list of man m_t along with S_9^t is shown below. The portion of m_t's preference list consisting of women from the proper part of the instance is denoted by P_t.

m_t	:	$P_t\ w_{t_1}\ w'_{t_1}$	w_{t_1}	:	$d_{t_1}\ d'_{t_1}$	d_{t_1}	:	$m_{t_2}\ m'_{t_2}$
m_{t_2}	:	$w_{t_2}\ w'_{t_2}$	w_{t_2}	:	$d_{t_2}\ d'_{t_2}$	d_{t_2}	:	$m_{t_3}\ m'_{t_3}$
m_{t_3}	:	$w_{t_3}\ w'_{t_3}$	w_{t_3}	:	$d_{t_3}\ d'_{t_3}$	d_{t_3}	:	$m_t\ m'_{t_1}$
m'_{t_1}	:	w_{t_3}	w'_{t_1}	:	d_{t_3}	d'_{t_1}	:	m_t
m'_{t_2}	:	w_{t_1}	w'_{t_2}	:	d_{t_1}	d'_{t_2}	:	m_{t_2}
m'_{t_3}	:	w_{t_2}	w'_{t_3}	:	d_{t_2}	d'_{t_3}	:	m_{t_3}

This ends the reduction, which can be computed in polynomial time. Now, we prove that there is a one-to-one correspondence between the complete stable matchings in I and the stable matchings in I'.

First we show that there is a one-to-one correspondence between the matchings of I and the matchings in the proper part of I'. This comes from the natural one-to-one correspondence between the edges of I and the triples in the proper part of I'. More precisely, if M is a matching in I, then a corresponding matching M_p in the proper part of I is created as follows: $(a_i, b_j) \in M \iff (m_i, w_j, d_{j,i}) \in M_p$ and $(a_i, b_j^T) \in M \iff (m_i, w_j^T, d_j^T) \in M_p$. Next, we show that stability is preserved by this correspondence.

Lemma 6.4.1 *A matching M of I is weakly stable if and only if the corresponding matching M_p in the proper part of I' is weakly stable.*

Proof It is enough to show that an edge (a_i, b_j) is blocking in I if and only if the corresponding triple $(m_i, w_j, d_{j,i})$ is also (strongly) blocking in I'; and similarly, an edge (a_i, b_j^T) is blocking in I if and only if the corresponding triple (m_i, w_j^T, d_j^T) is also blocking in I'.

Suppose first that (a_i, b_j) is blocking in I, which means that a_i is either unmatched or prefers b_j to $M(a_i)$ and b_j is either unmatched or prefers a_i to $M(b_j)$. This implies that m_i prefers w_j to $M_p(m_i)$, w_j prefers $d_{j,i}$ to $M(w_j)$, and $d_{j,i}$ is unmatched in M_p, i.e. $(m_i, w_j, d_{j,i})$ is blocking in I'. Similarly, if (a_i, b_j^T) is blocking then a_i is either unmatched or prefers b_j^T to $M(a_i)$ and b_j^T is unmatched in M. This implies that m_i prefers w_j^T to $M_p(m_i)$, w_j^T and d_j^T are both unmatched in M_p, and hence (m_i, w_j^T, d_j^T) is blocking in I'.

In the other direction, if $(m_i, w_j, d_{j,i})$ is blocking in I', then m_i prefers w_j to $M_p(m_i)$, w_j prefers $d_{j,i}$ to $M_p(w_j)$, and $d_{j,i}$ is unmatched in M_p. This implies that a_i is either unmatched or prefers b_j to $M(a_i)$ and b_j is either unmatched or prefers a_i to $M(b_j)$, so (a_i, b_j) is blocking in I. Similarly, if (m_i, w_j^T, d_j^T) is blocking in I', then w_j^T and d_j^T are both unmatched in M_p and m_i prefers w_j^T to $M_p(m_i)$. This implies that a_i is either unmatched or prefers b_j^T to $M(a_i)$ and b_j^T is unmatched in M, so (a_i, b_j^T) is blocking in I. □

Furthermore, if the matching M is complete, then we can enlarge the corresponding matching to the additional part of I' by matching every $S_9^t \setminus m_t$ in the unique stable way. So by adding this matching $M_{S_9^t \setminus m_t}$ to M_p for every t, this leads to the one-to-one correspondence between the complete stable matchings of I and the stable matching of I'.

Lemma 6.4.2 *The instance I admits a complete stable matching M if and only if the reduced instance I' admits a stable matching M_p, where M_p is the corresponding matching of M.*

Proof The stability of M implies that M_p is stable in the proper part of I' by Lemma 6.4.1. The completeness of M and Lemma 6.3.2 implies that M_p is also stable in the additional part of I'.

In the other direction, if M_p is stable then every man in M_p must be matched in a proper triple. For, if a proper man m_t does not have a proper partner in M then S_9^t would contain a blocking triple, by Lemma 6.3.1. This implies that the corresponding matching M, defined in Lemma 6.4.1, is complete. The stability of M is a consequence of Lemma 6.4.1. Finally, we note that the additional part has a unique stable matching, since every $S_9^t \setminus a_t$ must be matched in the unique stable way indicated by Lemma 6.3.2, which implies the one-to-one correspondence. □

The following theorem is a direct consequence of Lemma 6.4.2, and by observing that no agent's preference list in I' exceeds the length of five.

Theorem 6.4.1 *Determining the existence of a stable matching in a given instance of cyclic* 3DSMI *is NP-complete, even if the preference list of each agent is of length at most five.*

6.5 NP-completeness of 3DSM under strong stability

In this section we prove that 3DSM is NP-hard under strong stability. As in Section 6.4, our 9-Sun gadgets play an instrumental role. The reduction in this section is somewhat more complex than the other reductions in this thesis. We have therefore provided an example in Section 6.7 to illustrate some of the more involved steps of the transformation.

6.5.1 The reduction

The reduction we describe in this section again begins with an instance of $(3,3)$-COM-SMTI, only this time we assume that ties are allowed on the men's, rather than the women's preference lists. To be precise, we assume the underlying graph of a $(3,3)$-COM-SMTI instance I to have a vertex set $((A_1 \cup A_2) \cup B)$ that consists of a set $A_1 = \{a_1, a_2, \ldots, a_{n_1}\}$ of men with strictly ordered preference lists, and a set with $A_2 = \{a_1^T, a_2^T, \ldots, a_{n_2}^T\}$ of men with preference lists consisting of a single tie of length two, and $n_1 + n_2 = n$. We let $A = A_1 \cup A_2$. The set $B = \{b_1, b_2, \ldots, b_n\}$ consists entirely of women with strictly ordered preference lists. As previously stated, all agents of I have a preference list of length at most three.

Given an instance I of $(3,3)$-COM-SMTI as defined above, we create an instance I' of 3DSM as follows.

Step 1: the proper instance

The *proper instance* I_p of cyclic 3DSMI is a subinstance of I' with agents Q_p, R_p, and S_p of men, women, and dogs, respectively, with each set being of size n.

The preference list of woman $w_j \in R_p$ is the single entry $d_j \in S_p$. The preference list of $d_j \in S_p$ is such that if $\mathcal{L}_{b_j}^l = a_i$, then $\mathcal{L}_{d_j}^l = m_i$. Otherwise, if $\mathcal{L}_{b_j}^l = a_i^T$, then $\mathcal{L}_{d_j}^l = m'_{i,j}$ for $1 \leq l \leq r$,

where r is the length of b_j's list. So the preference list of dog d_j is essentially the "same" as that of woman b_j, only with men in Q_p rather than A.

The preference list of a man $m_i \in Q_p$ created in correspondence to man $a_i \in A_1$ is given as follows. If $\mathcal{L}^l_{a_i} = b_j$, then $\mathcal{L}^l_{m_i} = w_j$ for $1 \leq l \leq r$, where r is the length of a_i's list. So the preference list of man m_i is essentially the "same" as that of man a_i. For each man m_i created in correspondence to man $a_i^T \in A_2$, with a preference list consisting of a single tie of length two, say (b_r, b_s), we create five men m_i^T, $m'_{i,r}$, $m''_{i,r}$, $m'_{i,s}$, $m''_{i,s}$, four women $w'_{i,r}$, $w''_{i,r}$, $w'_{i,s}$, $w''_{i,s}$ and four dogs $d'_{i,r}$, $d''_{i,r}$, $d'_{i,s}$, $d''_{i,s}$. The preference list of m_i^T contains $w'_{i,r}$ and $w'_{i,s}$ in an arbitrary order, and the other preference lists are as shown below.

$$
\begin{array}{llll}
m'_{i,r} : w'_{i,r} \quad w_r & w'_{i,r} : d'_{i,r} \quad d''_{i,r} & d'_{i,r} : m''_{i,r} \quad m_i^T \\
m''_{i,r} : w''_{i,r} & w''_{i,r} : d''_{i,r} \quad d'_{i,r} & d''_{i,r} : m'_{i,r} \quad m''_{i,r} \\
m'_{i,s} : w'_{i,s} \quad w_s & w'_{i,s} : d'_{i,s} \quad d''_{i,s} & d'_{i,s} : m''_{i,s} \quad m_i^T \\
m''_{i,s} : w''_{i,s} & w''_{i,s} : d''_{i,s} \quad d'_{i,s} & d''_{i,s} : m'_{i,s} \quad m''_{i,s}
\end{array}
$$

These agents are added to the sets Q_p, R_p and S_p, respectively. Note that in I_p, every set of agents has the same cardinality: $n_p = |Q_p| = |R_p| = |S_p| = n + 4n_2$. The notions of *proper agent*, *proper partner* and *proper triple* are defined in the obvious way, i.e., they all belong to the proper instance.

Step 2: the additional part (add 9-Suns)

The *additional part* of I' is the disjoint union of $3n_p$ copies of S_9, such that the i^{th} copy of S_9, denoted S_9^i, incorporates the i^{th} agent of I_p, as described in Step 2 of the previous reduction for the proof of Theorem 6.4.1 (we omit the full description of this process again). The new agents are referred to as *additional agents*.

Let $M_s = \cup_{i \in \{1,\ldots,3n_p\}} M_{S_9^i \setminus a_i}$ be the unique strongly stable matching of the additional part, as described in Section 6.3 and Lemma 6.3.2, where a_i is the proper agent of S_9^i. We sometimes call M_s the *additional matching*.

We call $C = \cup_{i \in \{1,\ldots,3n_p\}} C_{S_9^i \setminus a_i}$ the set of *covered additional agents*, as these additional agents are covered by M_s, and we call $U = \cup_{i \in \{1,\ldots,3n_p\}} U_{S_9^i \setminus a_i}$ the set of *uncovered additional agents*, as these additional agents are not covered by M_s.

Step 3: pad the instance

Note that U has equal numbers of men, women and dogs. The *fitting part* of I' is constructed on U by creating disjoint triples that cover U. This is done in such a way that every agent has exactly one agent in his/her/its list, i.e. the fitting part is a complete matching of U, denoted by M_f. There is a certain amount of nondeterminism in this step, as there are a number of ways this step can be accomplished.

Finally, the *dummy part* is obtained by an arbitrary extension of the preference lists to ensure that all preference lists are complete. Note that this does not involve adding any additional agents. By putting together the three subinstances – the proper, additional, and fitting parts – we have constructed the complete instance I'. The preferences of the agents over partners in different parts respect the order in which we defined these parts: the list of a proper agent contains the proper partners first, then the additional partners, and finally the dummy partners; the list of a covered additional agent contains the additional partners first, then the dummy partners; the list of an uncovered additional agent contains the additional partners first, then the fitting partner, and finally the dummy partners.

Thus we have reduced an instance I of $(3,3)$-COM-SMTI to an instance I' of 3DSM in polynomial time.

We show that there is a one-to-one correspondence between the complete stable matchings of I and the complete strongly stable matchings of I_p. The stability is preserved via the following one-to-one correspondence between the complete matchings of I and complete matchings of I':

$$(a_i, b_j) \in M \iff (m_i, w_j, d_j) \in M_p$$

$$(a_i^T, b_s) \in M \iff (m_i^T, w'_{i,s}, d'_{i,s}), (m''_{i,s}, w''_{i,s}, d''_{i,s}), (m'_{i,s}, w_s, d_s) \in M_p$$

$$(a_i^T, b_s) \notin M \iff (m'_{i,s}, w'_{i,s}, d''_{i,s}), (m''_{i,s}, w''_{i,s}, d'_{i,s}) \in M_p$$

Lemma 6.5.1 *A complete matching M of I is stable if and only if the corresponding complete matching M_p of I_p is strongly stable.*

Proof As a man a_i^T cannot belong to a blocking pair in I, it may be verified that his corresponding

copy m_i^T cannot belong to a weakly blocking triple in I_p either. Therefore, it is enough to show that a pair (a_i, b_j) is blocking for M if and only if the corresponding triple (m_i, w_j, d_j) is blocking for M_p. But this is obvious, because the preference lists of a_i and m_i are essentially the same, and the preference lists of b_j and d_j are also essentially the same. □

Now, given a matching M of I let us create the corresponding matching M of I' by adding M_s and M_f to M_p, so $M = M_p \cup M_s \cup M_f$.

Lemma 6.5.2 *The instance I admits a complete stable matching M if and only if the reduced instance I' admits a strongly stable matching M', where M' is the corresponding matching of M.*

Proof Suppose that we have a complete stable matching M of I, and M' is the corresponding matching in I'. Lemma 6.5.1 implies that every proper agent has a proper partner in M' and no proper triple is weakly blocking. Therefore, no proper agent can be involved in any weakly blocking triple either. Recall that M_s is the union of the unique strongly stable matchings of the suitable part. By construction of M_s, every covered additional agent has an additional partner in M' and by Lemma 6.3.4, no additional triple is weakly blocking. Therefore, no such agent can be part of any weakly blocking triple. Finally, every uncovered additional agent has a fitting partner in M', so these agents cannot form a weakly blocking triple either, since an uncovered additional agent prefers only additional partners to fitting partners, which cannot be involved in a weakly blocking triple. Hence M' is strongly stable.

In the other direction, suppose that M' is a strongly stable matching of I'. Every proper agent must have a proper partner, since otherwise if a_t had no proper partner in M', then $\overline{S_9^t}$ would contain an additional weakly blocking triple, by Lemma 6.3.3. So the corresponding matching M in I is complete. The stability of M is a consequence of Lemma 6.5.1. Finally, we note that the additional agents must be matched in the unique strongly stable way in M', namely, the covered additional agents must be covered by matching M_s, by Lemma 6.3.4, and the uncovered additional agents must be covered by M_f (recall this is the matching created during the fitting part), since otherwise a fitting triple would weakly block M'. Therefore, we have a one-to-one correspondence as was claimed. □

Theorem 6.5.1 *Determining the existence of a strongly stable matching in a given instance of* 3DSM *is NP-complete.*

6.6 Conclusion and open questions

The 9-Sun described in Section 6.3 is the smallest example that we can find of a 3DSMI instance with no weakly stable matching. Is there a smaller example? In the case of strong stability, one can construct smaller examples (with $n = 4$) that admit no strongly stable matching. However, we have not found a use for such instances as gadgets for NP-hardness proofs.

It is an intriguing question to determine if there exists an instance of 3DSM that admits no weakly stable matching. A natural place to start would be to try to complete the preference lists of the 9-Sun in a way that does not introduce a weakly stable matching. However, we conjecture that this is not possible. A larger question is whether there is a polynomial-time algorithm to find a weakly stable matching or report that none exists, given an instance of 3DSM.

It is very uncommon to find a matching problem with preferences for which there is no clear way to extend a hardness result to complete preference lists (in fact, we know of no other such problem). Could it really be the case that when one attempts to complete the preference lists of a 3DSMI instance, one cannot avoid introducing a weakly stable matching?

6.7 Example

We consider an example reduction from a simple $(3, 3)$-COM-SMTI instance I with five men $\{a_1, a_2, a_3, a_4, a_5\}$ and five women $\{b_1, b_2, b_3, b_4, b_5\}$. The instance is given in Figure 6.2. The men a_4 and a_5 have preference lists consisting of a single tie of size two, and all the other agents of the instance have strictly ordered preference lists. This instance is a "yes" instance, for $M = \{(a_1, b_1), (a_2, b_2), (a_3, b_5), (a_4, b_4), (a_5, b_3)\}$ is a complete stable matching for I.

Even for this small instance, the derived instance I' has a total of 663 agents (221 of each set), and the sum of the lengths of the preference lists of I' exceeds 100,000. Ideally, we would present all the details, but constructing a complete example seems a bit daunting. We will instead illustrate the second step of the reduction, which involves creating a large number of agents, from the perspective of just a single agent. For the first step, however, we give a complete construction.

The result of Step 1 of the reduction is given in Figure 6.3. Each man a_i with a strictly ordered preference list has been transformed into a man m_i, whereas a man a_j whose preference list is a

$$
\begin{array}{llll}
a_1: & b_1 & & \\
a_2: & b_2 & b_1 & b_4 \\
a_3: & b_5 & b_1 & \\
a_4: & (b_4 & b_5) & \\
a_5: & (b_3 & b_4) &
\end{array}
\qquad
\begin{array}{llll}
b_1: & a_2 & a_3 & a_1 \\
b_2: & a_5 & a_2 & \\
b_3: & a_5 & & \\
b_4: & a_4 & a_5 & a_2 \\
b_5: & a_4 & a_3 &
\end{array}
$$

Figure 6.2: The given instance I of $(3,3)$-COM-SMTI

$$
\begin{array}{llll}
m_1: & w_1 & & \\
m_2: & w_2 & w_1 & w_4 \\
m_3: & w_5 & w_1 & \\
m_4^T: & w'_{4,4} & w'_{4,5} & \\
m_5^T: & w'_{5,3} & w'_{5,4} &
\end{array}
\qquad
\begin{array}{lll}
w_1: & d_1 & \\
w_2: & d_2 & \\
w_3: & d_3 & \\
w_4: & d_4 & \\
w_5: & d_5 &
\end{array}
\qquad
\begin{array}{llll}
d_1: & m_2 & m_3 & m_1 \\
d_2: & m_5 & m_2 & \\
d_3: & m_5 & & \\
d_4: & m_4 & m_5 & m_2 \\
d_5: & m_4 & m_3 &
\end{array}
$$

$$
\begin{array}{lll}
m'_{4,4}: & w'_{4,4} & w_4 \\
m''_{4,4}: & w''_{4,4} & \\
m'_{4,5}: & w'_{4,5} & w_5 \\
m''_{4,5}: & w''_{4,5} &
\end{array}
\qquad
\begin{array}{lll}
w'_{4,4}: & d'_{4,4} & d''_{4,4} \\
w''_{4,4}: & d''_{4,4} & d'_{4,4} \\
w'_{4,5}: & d'_{4,5} & d''_{4,5} \\
w''_{4,5}: & d''_{4,5} & d'_{4,5}
\end{array}
\qquad
\begin{array}{lll}
d'_{4,4}: & m''_{4,4} & m_4^T \\
d''_{4,4}: & m'_{4,4} & m''_{4,4} \\
d'_{4,5}: & m''_{4,5} & m_4^T \\
d''_{4,5}: & m'_{4,5} & m''_{4,5}
\end{array}
$$

$$
\begin{array}{lll}
m'_{5,3}: & w'_{5,3} & w_3 \\
m''_{5,3}: & w''_{5,3} & \\
m'_{5,4}: & w'_{5,4} & w_4 \\
m''_{5,4}: & w''_{5,4} &
\end{array}
\qquad
\begin{array}{lll}
w'_{5,3}: & d'_{5,3} & d''_{5,3} \\
w''_{5,3}: & d''_{5,3} & d'_{5,3} \\
w'_{5,4}: & d'_{5,4} & d''_{5,4} \\
w''_{5,4}: & d''_{5,4} & d'_{5,4}
\end{array}
\qquad
\begin{array}{lll}
d'_{5,3}: & m''_{5,3} & m_5^T \\
d''_{5,3}: & m'_{5,3} & m''_{5,3} \\
d'_{5,4}: & m''_{5,4} & m_5^T \\
d''_{5,4}: & m'_{5,4} & m''_{5,4}
\end{array}
$$

Figure 6.3: The proper instance of I' resulting from Step 1 of the reduction

tie has been transformed into a man m_j^T. Each dog d_i of the proper instance has been created in correspondence to the woman b_i, and the women of the proper instance are created with a preference list containing a single dog. The block of agents beginning with $m'_{4,4}$ was created in correspondence to a_4, and the block of agents beginning with $m'_{5,3}$ was created from a_5.

The numbers of agents created in Step 2, in which we add 9-Suns, is quite large. We illustrate this step on a single agent, man m_3, chosen arbitrarily. Recall that the preference list of m_3 after the end of Step 1 consists of the woman w_5 followed by w_1.

$$
\begin{array}{lcllll}
m_3 & : & w_5 & w_1 & w_{3_1} & w'_{3_1} \\
m_{3_2} & : & \underline{w_{3_2}} & w'_{3_2} & & \\
m_{3_3} & : & w_{3_3} & \underline{w'_{3_3}} & & \\
m'_{3_1} & : & \underline{w_{3_3}} & & & \\
m'_{3_2} & : & \underline{w_{3_1}} & & & \\
m'_{3_3} & : & w_{3_2} & & &
\end{array}
\qquad
\begin{array}{lcll}
w_{3_1} & : & \underline{d_{3_1}} & d'_{3_1} \\
w_{3_2} & : & d_{3_2} & \underline{d'_{3_2}} \\
w_{3_3} & : & \underline{d_{3_3}} & d'_{3_3} \\
w'_{3_1} & : & d_{3_3} & \\
w'_{3_2} & : & d_{3_1} & \\
w'_{3_3} & : & \underline{d_{3_2}} &
\end{array}
\qquad
\begin{array}{lcll}
d_{3_1} & : & m_{3_2} & \underline{m'_{3_2}} \\
d_{3_2} & : & \underline{m_{3_3}} & m'_{3_3} \\
d_{3_3} & : & m_t & \underline{m'_{3_1}} \\
d'_{3_1} & : & m_3 & \\
d'_{3_2} & : & \underline{m_{3_2}} & \\
d'_{3_3} & : & m_{3_3} &
\end{array}
$$

Figure 6.4: Step 2 illustrated on man m_3.

We now illustrate the notion of the *covered* and *uncovered agents*, as described in Step 2 in Section 6.5.1. If we consider man m_3 to be the "third" agent of I', then this 9-Sun is the third copy of S_9 to be added to I', and is therefore denoted S_9^3. The covered agents $C_{S_9^3 \setminus m_3}$ are denoted by underlining in Figure 6.4. The uncovered agents $U_{S_9^3 \setminus m_3}$ are those in the set $\{m'_{3_3}, w'_{3_1}, w'_{3_2}, d'_{3_1}, d'_{3_3}\}$.

Notice that this particular set of uncovered agents consists of one man, two women, and two dogs. When we apply this process to, say, w_1 and d_1, where w_1 and d_1 are the sixth and eleventh agents of the proper instance, respectively, we would add the sixth and eleventh copies of the 9-Sun, denoted S_9^6 and S_9^{11}, respectively, to I'. The uncovered agents $U_{S_9^6 \setminus w_1}$ consist of one woman, two men, and two dogs. The uncovered agents $U_{S_9^{11} \setminus d_1}$ consist of one dog, two men, and two women. Thus it is easy to see that the numbers of men, women, and dogs of the uncovered agents are equal.

In Step 3, when the fitting part is constructed, we could, for example, create three arbitrary triples from the sets $U_{S_9^3 \setminus m_3}$, $U_{S_9^6 \setminus w_1}$, and $U_{S_9^{11} \setminus d_1}$, as these sets contain a total of six men, six women, and six dogs. Clearly there are a number of ways in which this can be accomplished. Any of these ways will do.

Finally, in Step 4, the dummy part completes the preference lists of the existing agents by arbitrarily completing each agent's preference list in any way that adheres to the rule that the list of a proper agent contains the proper partners first, then the additional partners, and finally everyone else; the list of a covered additional agent contains the additional partners first, then everyone else; and the list of an uncovered additional agent contains the additional partners first, then the unique fitting partner, then everyone else.

We illustrate this process on the proper agent m_3, the covered additional agent m_{3_2}, and the uncovered additional agent m'_{3_3} in Figure 6.5. Notice the woman w^* in the list of m'_{3_3} – this is the fitting partner found in Step 3.

m_3 : $w_5\ w_1\ w_{3_1}\ w'_{3_1}$ ... other proper women ... other women in S_9^3 ... all others ...
m_{3_2} : $w_{3_2}\ w'_{3_2}$ ... other women in S_9^3 ... all others ...
m'_{3_3} : $w_{3_2}\ w^*$ ... other women in S_9^3 ... all others ...

Figure 6.5: The completion of the preference lists in Step 4

Chapter 7

Popular matchings: structure and algorithms

7.1 Introduction

We consider the popular matching problem (POP-M) in the setting of the post allocation problem (PA). All the relevant concepts and terminology for PA and POP-M were introduced in Section 2.3.3. Our goal in this chapter is to characterize the structure of the set of popular matchings for an instance of POP-M. This characterization is in terms of a novel data structure which we call a *switching graph*. We will show that this structure can be exploited to enable the design of efficient algorithms for a range of extensions of the basic popular matching problem, such as counting and enumerating popular matchings, generating a popular matching uniformly at random, and finding popular matchings that satisfy various additional optimality criteria. In particular, we improve on the algorithm of Kavitha and Nasre [60] by showing how minimum-cost popular matchings can be found in $O(n + m)$ time, and rank-maximal and generous popular matchings in $O(n \log n + m)$ time (these terms are defined in Section 7.3.5).

7.1.1 Preliminaries

For convenience, a unique *last-resort post*, denoted by $l(a)$, is created for each applicant a, and placed last on a's preference list. As a consequence, in any popular matching, every applicant is

matched, although some may be matched to their last-resort post. Note that this technique was also used by Abraham et al [5]. Let $f(a)$ denote the first-ranked post on a's preference list; any post that is ranked first by at least one applicant is called an *f-post*. Let $s(a)$ denote the first non-f-post on a's preference list. (Note that $s(a)$ must exist, for $l(a)$ is always a candidate for $s(a)$). Any such post is called an *s-post*. By definition, the sets of f-posts and s-posts are disjoint.

The following fundamental result, proved in [5], completely characterizes popular matchings, and is key in establishing the structural results that follow.

Theorem 7.1.1 *(Abraham et al [5]) A matching M for an instance of POP-M is popular if and only if (i) every f-post is matched in M, and (ii) for each applicant a, $M(a) \in \{f(a), s(a)\}$.*

In light of Theorem 7.1.1, given a POP-M instance I we define the *reduced instance* of I to be the instance obtained by removing from each applicant a's preference list every post except $f(a)$ and $s(a)$. It is immediate that the reduced instance of I can be derived from I in $O(n + m)$ time, i.e., in time that is linear in the size of the input. Henceforth, unless explicitly stated, it is assumed that a given instance of POP-M is a reduced instance. For a (reduced) instance I of POP-M, let M be a popular matching, and let a be an applicant. Denote by $O_M(a)$ the post on a's (reduced) preference list to which a is not assigned in M. Note that since I is a reduced instance, $O_M(a)$ is well defined. So, if a is matched to $f(a)$ in M, then $O_M(a) = s(a)$, whereas if a is matched to $s(a)$ in M, then $O_M(a) = f(a)$.

Throughout this chapter, we refer to a working example found in Section 7.5, which illustrates many of the important concepts surrounding popular matchings and their structure.

Example As an illustration of the reduced instance of a POP-M instance, consider the full POP-M instance I of Figure 7.2 in Section 7.5 with applicants $a_1 \ldots a_{16}$ and posts $p_1 \ldots p_{18}$. The reduced instance of I is shown in Figure 7.3. It may be inferred that the matching shown in Figure 7.3 satisfies the conditions of Theorem 7.1.1, and is thus a popular matching for this instance.

7.2 The structure of popular matchings – the switching graph

The key concept that underlies the characterization of the structure of popular matchings is the *switching graph*, a directed graph which captures all the ways in which applicants may form different popular matchings by switching between the two posts on their reduced preference lists. Given a popular matching M for an instance I of POP-M, the *switching graph* G_M of M is a directed graph with a vertex for each post, and a directed edge (p_i, p_j) for each applicant a, where $p_i = M(a)$ and $p_j = O_M(a)$. A vertex v is called an ***f-post vertex*** (respectively ***s-post vertex***) if the post it represents is an f-post (respectively s-post). Each vertex (respectively edge) is labelled with the post (respectively applicant) that it represents. In fact, we refer to posts and vertices of G_M interchangeably, and likewise to applicants and edges of G_M. A *component* of G_M is any maximal weakly connected subgraph of G_M. An applicant (respectively post) is said to be *in* a component, or path, or cycle of G_M if the edge (respectively vertex) representing it is in that component, path or cycle.

A very similar graph was defined by Mahdian [Lemma 2][71]. However, Mahdian used this structure solely to investigate the existence of popular matchings in random instances of POP-M.

Some simple properties of switching graphs are spelled out in the following lemma.

Lemma 7.2.1 *Let M be a popular matching for an instance I of POP-M, and let G_M be the switching graph of M. Then*
(i) Each vertex in G_M has outdegree at most 1.
(ii) The sink vertices of G_M are those vertices corresponding to posts that are unmatched in M, and are all s-post vertices.
(iii) Each component of G_M contains either a single sink vertex or a single cycle.

Proof (i) A vertex v in G_M has an outgoing edge for each applicant who is matched in M to the post represented by v, and there can be at most one such applicant because M is a matching.
(ii) A vertex has no outgoing edge if and only if it represents an unmatched post, and by Theorem 7.1.1 (i) any such post is an s-post.
(iii) This is an easy consequence of (i). □

Every component of the switching graph is therefore either a tree or a "tree plus one edge", and

is called a *tree component* or a *cycle component* according as it contains a sink or a cycle. Each cycle in G_M is called a *switching cycle*, and must have even length, as the posts of such a cycle are alternately f- and s-posts. If T is a tree component in G_M with sink p, and if q is another s-post vertex in T, the (unique) path from q to p is called a *switching path*. So each cycle component of G_M has a unique switching cycle, but each tree component may have zero or more switching paths; to be precise it has one switching path for each s-post vertex that it contains, other than the sink vertex. It is immediate that the cycle components and tree components of G_M can be identified, say using depth-first search, in linear time.

Example Figure 7.4 of Section 7.5 provides an illustrative example of the switching graph of a popular matching M in the POP-M instance described in Figures 7.2 and 7.3. The switching graph of this instance contains one cycle component and two tree components.

Let C be a switching cycle of G_M. To *apply* C to M is to assign each applicant a in C to $O_M(a)$, while leaving all other applicants assigned as in M. We denote by $M \cdot C$ the matching obtained by applying the switching cycle C to M.

Similarly, let P be a switching path of G_M. To *apply* P to M is to assign each applicant a in P to $O_M(a)$, while leaving all other applicants assigned as in M. We denote by $M \cdot P$ the matching obtained by applying the switching path P to M. Note that, if p is the sink vertex in G_M and the path P begins at vertex q, then in $M \cdot P$, the post p is matched but the post q is unmatched (whereas in M, q is matched and p is unmatched). In general, if we apply a switching cycle or switching path that contains the edge representing applicant a, and this edge connects post q to post p, then applicant a is switched from post q to post p as a result.

Note that the switching graph is uniquely determined by a particular popular matching M, but different popular matchings for the same instance yield different switching graphs. However, all switching graphs for an instance of POP-M have the same number of vertices (one for each post), and the same number of edges (one for each applicant).

The significance of switching paths and switching cycles begins to emerge in the following theorem.

Theorem 7.2.1 *Let M be a popular matching for an instance I of POP-M, and let G_M be the switching graph of M.*

(i) If C is a switching cycle in G_M then $M \cdot C$ is a popular matching for I.
(ii) If P is a switching path in G_M then $M \cdot P$ is a popular matching for I.

Proof (i) Let $M' = M \cdot C$. By Theorem 7.1.1, it is sufficient to argue that (a) every f-post is matched in M' and (b) for each applicant a, $M'(a) \in \{f(a), s(a)\}$. It is clear, from the cyclic nature of the reassignments that take place on applying C, that each post that is matched in M is also matched in M'. Hence all f-posts are matched in M', and condition (a) is established. Furthermore, each applicant $a_i \notin C$ is assigned to the same post in M' as in M, and each applicant $a_i \in C$ is assigned to $O_M(a_i)$ in M', which is clearly either $f(a_i)$ or $s(a_i)$, establishing (b).
(ii) Condition (b) follows by a similar argument to that of (i), since every applicant a is still assigned to either $f(a)$ or $s(a)$ in $M \cdot P$. Also, the only post that is "vacated" by applying P is the s-post corresponding to the initial vertex of P. Each f-post in P is filled by a different applicant, and all f-posts not in P are filled by the same applicant as in M, so that condition (a) is satisfied. □

Theorem 7.2.1 shows that, given a popular matching M for an instance I of POP-M, and the switching graph of M, we can potentially find other popular matchings. Our next step is to establish that this is essentially the only way to find other popular matchings. More precisely, we show that if M' is an arbitrary popular matching for I, then M' can be obtained from M by applying a sequence of switching cycles and switching paths, at most one per component of G_M. First we state a simple technical lemma, the proof of which is an easy consequence of the definition of the switching graph.

Lemma 7.2.2 *Let M be a popular matching for an instance I of POP-M, let G_M be the switching graph of M, and let M' be an arbitrary popular matching for I. If the edge representing applicant a in G_M connects the vertex p to the vertex q, then*
(i) a is assigned to p in M;
(ii) if $M'(a) \neq M(a)$ then a is assigned to q in M'.

Lemmas 7.2.3 and 7.2.4 deal with switching cycles and switching paths respectively.

Lemma 7.2.3 *Let M be a popular matching for an instance I of POP-M, let T be a cycle component with cycle C in the switching graph G_M of M, and let M' be an arbitrary popular matching for I.*
(i) Either every applicant a in C has $M'(a) = M(a)$, or every such applicant a has $M'(a) =$

$O_M(a)$.

(ii) Every applicant a in T that is not in C has $M'(a) = M(a)$.

Proof (i) Let $a_{i_0}, \ldots, a_{i_{r-1}}$ be the sequence of applicants in C, and suppose that $M'(a_{i_j}) \neq M(a_{i_j})$ for some a_{i_j} in C. Then, by Lemma 7.2.2, a_{i_j} must be assigned in M' to $O_M(a_{i_j}) = M(a_{i_{j+1}})$ (where $j+1$ is taken mod r). It follows that $a_{i_{j+1}}$ must also be assigned to a different post in M' as compared to M, and that this post must be $O_M(a_{i_{j+1}}) = M(a_{i_{j+2}})$ (where $j+2$ is taken mod r). Inductively, this implies that every applicant in C is assigned different posts in M and M' if any one of them is.

(ii) Suppose, for a contradiction, that an applicant a who is in T but not in C has $M'(a) \neq M(a)$. Let the sequence of distinct edges on the path in T that begins with edge a be $(a =)a_{j_1}, \ldots, a_{j_t}, \ldots, a_{j_s}$ where a_{j_t} is the last edge in this path that is not in the cycle C. Then, by an argument similar to that in (i) above, we must have $M'(a_{j_t}) = M(a_{j_{t+1}})$. But, by the same reasoning, we must have $M'(a_{j_s}) = M(a_{j_{t+1}})$, since the edge $a_{j_{t+1}}$ follows the edge a_{j_s} in the cycle. This implies that a particular post, namely $M(a_{j_{t+1}})$, has two applicants, a_{j_t} and a_{j_s}, assigned to it in M', a contradiction. □

Lemma 7.2.4 *Let M be a popular matching for an instance I of POP-M, let T be a tree component in the switching graph G_M of M, and let M' be an arbitrary popular matching for I. Then either every applicant a in T has $M'(a) = M(a)$, or there is a switching path P in T such that every applicant a in P has $M'(a) = O_M(a)$ and every applicant a in T that is not in P has $M'(a) = M(a)$.*

Proof Suppose that $M'(a) \neq M(a)$ for some applicant a in T. By an argument similar to that of part (i) of Lemma 7.2.3, the same must be true of every applicant on the path from a to the sink vertex of G_M. Suppose that two applicants in T whose edges have a common end point, say p, are both matched to different posts in M' as compared to M. Then, by Lemma 7.2.2, both would have to be assigned in M' to p, a contradiction. Hence the applicants in T who are assigned different posts in M and M' form a path ending at the sink vertex. Moreover, this path must begin at an s-post vertex, otherwise the f-post at the start of the path would be unfilled in M', contradicting Theorem 7.1.1, so the path is a switching path. □

Suppose that M is a popular matching for an instance I of POP-M, and that T and T' are distinct

components of the switching graph G_M of M. If we apply the switching cycle in T (if T is a cycle component) or a switching path in T (if T is a tree component) to obtain a different popular matching, then the assignments of the applicants in T' are unaffected. Hence, the component T' is present in the switching graph corresponding to the new matching. Intuitively, this means that the application of switching cycles and paths are independent processes when in different components of the switching graph. This notion of independence is captured in the following lemma.

Lemma 7.2.5 *Let T and T' be components of a switching graph G_M for a popular matching M, and let Q be either the switching cycle (if T is a cycle component) or a switching path (if T is a tree component) in T. Then, T' is a component in the switching graph $G_{M \cdot Q}$.*

We can now characterize fully the relationship between any two popular matchings for an instance of POP-M.

Theorem 7.2.2 *Let M and M' be two popular matchings for an instance I of POP-M. Then M' may be obtained from M by successively applying the switching cycle in each of a subset of the cycle components of G_M together with one switching path in each of a subset of the tree components of G_M.*

Proof We describe a procedure for obtaining M' from M in a way that will establish the claim. By Lemma 7.2.5, we can describe this procedure in terms of its separate effect on each component of the switching graph.

For each cycle component T of G_M, we know by Lemma 7.2.3 that either the applicants a in T all have $M(a) = M'(a)$, or those applicants in the unique cycle of T have $M'(a) = O_M(a)$. In the former case, we leave T unchanged, and in the latter case, we apply the switching cycle in T, so that every applicant a in T becomes matched to $M'(a)$.

For each tree component T' of G_M, we know by Lemma 7.2.4 that either every applicant a in T' has $M(a) = M'(a)$, or there is a single switching path P in T' such that every applicant a_j in P has $M'(a_j) = O_M(a_j)$, and all applicants a_k in T' but not in P must have $M(a_k) = M'(a_k)$. Hence, by applying P, every applicant a in T' is matched to $M'(a)$. Thus we obtain M' from M by successively applying at most one switching cycle per cycle component of G_M, and at most one

switching path per tree component of G_M. Moreover, the order in which these switching cycles and paths are applied is arbitrary. □

An immediate corollary of this theorem is a characterization of the set of popular matchings for a POP-M instance.

Corollary 7.2.1 *Let I be a POP-M instance, and let M be an arbitrary popular matching for I with switching graph G_M. Let the tree components of G_M be $X_1, \ldots, X_k$, and the cycle components of G_M be $Y_1, \ldots, Y_l$. Then, the set of popular matchings for I consists of exactly those matchings obtained by applying at most one switching path in X_i for each i ($1 \leq i \leq k$) and by either applying or not applying the switching cycle in Y_i for each i ($1 \leq i \leq l$).*

Example Taken all together, the figures and textual description of the example in Section 7.5 contain a POP-M instance, its reduced instance, the switching graph of a particular popular matching M, and an indication of how the application of switching paths and cycles leads to different popular matchings with different, but closely related, switching graphs.

7.3 Algorithms that exploit the structure

In this section we show how the characterization of the structure of the set of popular matchings for an instance of POP-M allows the construction of efficient algorithms to solve a number of extensions of the basic problem, namely to compute the number of popular matchings, to generate a popular matching uniformly at random, to enumerate the set of all popular matchings, to find all applicant-post pairs that can occur in a popular matching, and to find popular matchings that are optimal in one of a number of natural ways.

Each of these algorithms begins in the same way – by constructing the reduced instance, finding an arbitrary popular matching M (if one exists) with the $O(n+m)$ time algorithm given by Abraham et al [5], building the switching graph G_M, and identifying the cycle components and tree components of this graph using, say, depth-first search. Clearly all of this can be achieved in $O(n + m)$ time, where n is the number of applicants and posts and m is the sum of the lengths of the original preference lists, in other words in time that is linear in the input size. This sequence of steps is referred to as the *preprocessing phase*.

7.3.1 Counting popular matchings

Recall that a tree component having q s-posts has exactly $q-1$ switching paths. For a tree component X_i, denote by $\mathcal{S}(X_i)$ the number of s-posts in X_i. The following theorem is an immediate consequence of Corollary 7.2.1.

Theorem 7.3.1 *Let I be a POP-M instance, and let M be an arbitrary popular matching for I with switching graph G_M. Let the tree components of G_M be $X_1, \ldots, X_k$, and the cycle components of G_M be $Y_1, \ldots, Y_l$. Then, the number of popular matchings for I is $2^l * \prod_{i=1}^{k} \mathcal{S}(X_i)$.*

Thus, in light of Theorem 7.3.1, it is easy to see that an algorithm for counting the number of popular matchings for an instance I of POP-M first carries out the preprocessing phase, during which the number l of cycle components and $\mathcal{S}(X_i)$ for each tree component X_i in G_M are determined. Once these values are known, the algorithm then returns the product $2^l * \prod \mathcal{S}(X_i)$.

Theorem 7.3.2 *The number of popular matchings for an arbitrary instance of POP-M can be computed in linear time.*

7.3.2 Random popular matchings

Let I be an arbitrary POP-M instance, and let $\mathcal{M}$ denote the set of popular matchings for I. Corollary 7.2.1 facilitates the generation of a popular matching from $\mathcal{M}$ uniformly at random in linear time. The procedure again begins with the preprocessing phase, during which the cycle components $Y_1, \ldots, Y_l$ and the tree components $X_1, \ldots, X_k$ of G_M are identified. Next, for each cycle component Y_i a bit b is generated uniformly at random (fair coin). The unique switching cycle in Y_i is applied if and only if $b = 1$. Likewise, for each tree component X_i, the s-posts other than the sink are numbered $1, 2, \ldots, q-1$ where $q = \mathcal{S}(X_i)$, and a value r is chosen uniformly at random from the set $\{0, 1, \ldots, q-1\}$. If $r = 0$, no switching path from this component is applied, otherwise, the switching path beginning at the s-post numbered r in X_i is applied. The algorithm returns the popular matching obtained by applying this choice of switching cycles and switching paths.

Theorem 7.3.3 *Let I be an instance of POP-M, and let $\mathcal{M}$ denote the set of popular matchings for*

I. There is an algorithm to generate a popular matching from $\mathcal{M}$ uniformly at random in linear time.

Proof It is immediate that any one of the popular matchings for the instance is equally likely to be returned by the algorithm. To establish the complexity, it suffices to observe that the preprocessing phase is linear, and the total time spent applying the switching paths and cycles is also linear. □

7.3.3 Enumerating popular matchings

An algorithm for enumerating the set of popular matchings can be obtained by first computing an arbitrary popular matching M along with the switching graph G_M and then generating all possible popular matchings by applying switching paths and switching cycles as described in Corollary 7.2.1.

The algorithm begins with the preprocessing phase. During this phase, for each tree component X_i, $\mathcal{S}(X_i)$ is computed, and the s-posts of this component other than the sink are numbered $1, \ldots, q-1$, where $q = \mathcal{S}(X_i)$. Let $j = l + k$. Next, a vector $V = (v_1, \ldots v_j)$ is defined, where $v_i \in \{0, 1\}$ ($1 \leq i \leq l$) and $v_{l+i} \in \{0, 1, \ldots, \mathcal{S}(X_i) - 1\}$ ($1 \leq i \leq k$).

At this point the matching M is output, and V is initialized to be $(0, 0, \ldots, 0)$. The algorithm then loops through all possible values of the vector V. At each iteration, the switching cycle in Y_i is applied to M if and only if $v_i = 1$ ($1 \leq i \leq l$). For each k ($l < k \leq j$), if $v_k \neq 0$, the switching path beginning with the s-post numbered v_k in component X_k is applied to M. Otherwise, if $v_k = 0$, no switching path in X_k is applied. The popular matching so generated is then output and control passes to the next loop iteration.

Theorem 7.3.4 *Let I be a POP-M instance, and let $\mathcal{M}$ denote the set of popular matchings for I. There is an algorithm that enumerates $\mathcal{M}$ in $O(n + m + n|\mathcal{M}|)$ time.*

Proof The preprocessing phase occupies $O(n+m)$ time. The generation of each popular matching requires the identification and application of a set of switching paths and switching cycles. Within each component this can be done in time linear in the size of the component, so overall in time linear in the size of the switching graph, namely $O(n)$. □

7.3.4 Popular pairs

A *popular pair* for an instance I of POP-M, is an applicant-post pair (a_i, p_j) such that there exists a popular matching M with $(a_i, p_j) \in M$. We show that the popular pairs can be determined in linear time. The following lemma is the key.

Lemma 7.3.1 *Let M be a popular matching for an instance I of POP-M, and let G_M be the switching graph of M. Then, (a_i, p_j) is a popular pair if and only if (i) (a_i, p_j) is in M, or (ii) a_i is an incoming edge to p_j in G_M, and a_i and p_j are in a switching cycle or switching path in G_M.*

Proof The proof of sufficiency is easy, for if (a_i, p_j) is in M, it is by definition a popular pair. If instead, a_i is an incoming edge to p_j in G_M and a_i is in a switching cycle or path in G_M, we know by Theorem 7.2.1 that applying this switching cycle or path matches a_i and p_j in a popular matching.

On the other hand, suppose that $(a_i, p_j) \notin M$. If a_i is not an incoming edge to p_j, then $O_M(a_i) \neq p_j$, implying p_j is not on a_i's reduced preference list, so a_i can never be matched to p_j in a popular matching. Suppose that a_i is an incoming edge to p_j in G_M, but a_i is not in a switching cycle or path in G_M. If we suppose that (a_i, p_j) is in a popular matching M', then, by Theorem 7.2.2, there is a sequence of switching cycles and paths that can be applied to transform M to M'. But clearly for a_i to become matched to p_j, a_i must be in one of these switching cycles or paths, giving a contradiction. □

Theorem 7.3.5 *There is a linear time algorithm to generate all of the popular pairs for a POP-M instance.*

Proof After the preprocessing phase, all pairs in the resulting popular matching (if any) are output. The switching graph G_M is then traversed to find all applicants a_i in a switching path or switching cycle, and for each such a_i the pair (a_i, p_j) is output, where p_j is the end vertex of edge a_i. Lemma 7.3.1 guarantees that the correct set of pairs has been generated. The preprocessing phase and traversal of the switching graph can both be accomplished in linear time. □

7.3.5 Optimal popular matchings

Kavitha and Nasre [60] recently studied the following problem: suppose we wish to compute a matching that is not only popular, but is also optimal with respect to some additional well-defined criterion. They defined a natural optimality criterion and described an augmenting path-based algorithm for computing an optimal popular matching. In this section we will describe faster algorithms that exploit the switching graph of the instance to find an optimal popular matching with respect to certain optimality criteria. We first define two particular optimality criteria, in terms of the profile of the matching, which we discussed in detail in Section 2.3.1. We briefly recall here the necessary definitions regarding profiles.

For a POP-M instance with n_1 applicants and n_2 posts, we define the *profile* of M to be the (n_2+1)-tuple $(x_1, \ldots, x_{(n_2+1)})$ where, for each i $(1 \leq i \leq n_2 + 1)$, x_i is the number of applicants who are matched in M with their i^{th}-choice post. An applicant who is matched to his last-resort post is considered to be matched to his $(n_2 + 1)^{th}$-choice post, regardless of the length of his preference lists.

Total orders $\succ_L$ and $\prec_G$ on profiles are defined as follows. Suppose that $x = (x_1, \ldots, x_{n_2+1})$ and $y = (y_1, \ldots, y_{n_2+1})$ are profiles. Then

- $x \succ_L y$ if, for some j, $x_i = y_i$ for $1 \leq i < j$ and $x_j > y_j$;
- $x \prec_G y$ if, for some j, $x_i = y_i$ for $j < i \leq n_2 + 1$ and $x_j < y_j$.

A *rank-maximal* popular matching is a popular matching whose profile is maximal with respect to $\succ_L$. A *generous* popular matching is a popular matching whose profile is minimal with respect to $\prec_G$. Note that, since the number of $(n_2 + 1)^{th}$ choices is minimised, a generous popular matching is inevitably a maximum cardinality popular matching.

If a *weight* $w(a_i, p_j)$ is defined for each applicant-post pair with p_j acceptable to a_i, then the *weight* $w(M)$ of a popular matching M is $\sum_{(a_i,p_j)\in M} w(a_i, p_j)$. We call a popular matching *optimal* if it is of maximum or minimum weight depending on the context.

Some examples of optimal popular matchings include, but are not limited to:

- *Maximum cardinality* popular matchings: assign a weight of 0 to each pair involving a last resort post, and a weight of 1 to all other pairs, and find a maximum weight popular matching. (A linear time algorithm to find a maximum cardinality popular matching was already given by Abraham et al [5].)

- *Minimum cost maximum cardinality* popular matchings: assign a large weight, say n^2, to each pair involving a last resort post, and a weight of k to each pair (a_i, p_j) such that p_j is a_i's k^{th} choice in the original instance, and find a minimum weight popular matching.

- *Rank-maximal* popular matchings: assign a weight of 0 to each pair involving a last resort post, and a weight of n^{n-k+1} to each pair (a_i, p_j) where p_j is the kth choice of a_i, and find a maximum weight popular matching.

- *Generous* popular matchings : assign a weight of n^{n-k+1} to each pair (a_i, p_j) where p_j is the $(n-k)^{th}$ choice of a_i, and find a minimum weight maximum cardinality popular matching.

Kavitha and Nasre [60] described an $O(n^2 + m)$-time algorithm for finding minimum cost, generous (which they called *fair*), and rank-maximal popular matchings. In what follows, we give an $O(n + m)$-time algorithm for finding minimum cost maximum cardinality popular matchings and $O(n \log n + m)$-time algorithms for finding generous and rank-maximal popular matchings.

From the above description of generous and rank-maximal popular matchings, it is apparent that we may wish to assign very large weights to the applicant-post pairs, so we cannot assume that weights can be compared or added in O(1) time. We assume that these weights occupy $O(f(n))$ space for some function f, so that this is also the time for comparison or addition of such values.

Given an instance of POP-M and a particular allocation of weights, let M be a popular matching, and M_{opt} an optimal popular matching (maximum or minimum weight, as appropriate). By Theorem 7.2.2, M_{opt} can be obtained from M by applying a choice of at most one switching cycle or switching path per component of the switching graph G_M. The algorithm for computing M_{opt} will compute an arbitrary popular matching M, and make an appropriate choice of switching paths and switching cycles to apply in order to obtain an optimal popular matching. The next step is to show how to decide exactly which switching cycles and paths need be applied. In the following, for simplicity of presentation, we assume that "optimal" means "maximum". Analogous results hold in the "minimum" case.

If T is a cycle component of G_M, an *orientation* of T is either the set of pairs $\{(a, M(a)) : a \in T\}$, or the set $\{(a, M \cdot C(a)) : a \in T\}$, where C is the switching cycle in T. Likewise, if T is a tree component of G_M, an *orientation* of T is either the set of pairs $\{(a, M(a)) : a \in T\}$, or the set $\{(a, M \cdot P(a)) : a \in T\}$, for some switching path P in T. The weight of an orientation is the sum of the weights of the pairs in it, and an orientation of a component is *optimal* if its weight is at least as great as that of any other orientation. Intuitively, an optimal orientation of a component T of G_M assigns the applicants in T so that T contributes its optimal weight to the popular matching.

The following lemma establishes the relationship between M, M_{opt}, and optimal orientations of components of G_M.

Lemma 7.3.2 *If M is an arbitrary popular matching, T is a component of G_M, and M_{opt} is an optimal popular matching, then the set of pairs $\{(a, M_{opt}(a)) : a \in T\}$ is an optimal orientation of T.*

Proof Recall that M_{opt} may be generated from M by applying at most one switching cycle in each cycle component of G_M and at most one switching path in each tree component of G_M. Suppose that the set $\{(a, M_{opt}(a)) : a \in T\}$ is not an optimal orientation of T. Then if we were to generate a matching M' from M in exactly the same way as M_{opt}, except that we deal with the component T according to an optimal orientation, matching M' would have a weight greater than M_{opt}, a contradiction. □

In light of Lemma 7.3.2, an algorithm for computing an optimal popular matching can be constructed as follows. The algorithm begins, as always, with the preprocessing phase. The next step is to find an optimal orientation for each component in G_M. An optimal orientation of each cycle component T with switching cycle C can be found by comparing $\sum_{a \in C} w(a, M(a))$ with $\sum_{a \in C} w(a, M \cdot C(a))$. This is easily done in $O(f(n)|T|)$ time, and the outcome tells us whether or not the switching cycle C should be applied to give an optimal popular matching.

In the case of a tree component T, we would like to find an optimal orientation also in $O(f(n)|T|)$ time. This cannot necessarily be achieved by independent evaluation of each switching path in T. Instead, a depth-first traversal of T can be carried out, starting from the sink, and traversing edges in reverse direction. For an s-post vertex v, let P_v be the switching path beginning at v. To find the weight of the orientation of T resulting from the application of P_v, suppose that the switching path

starting at v has a and b as its first two edges. This evaluation involves subtracting the weight $w(a, v)$ from, and adding the weight $w(b, u)$ to, the weight of the orientation resulting from application of P_u, where u is the nearest s-post ancestor of v in the depth-first spanning tree. So the weight of each orientation can be computed in $O(f(n))$ time. By this means we can determine an optimal orientation of each tree component T in $O(f(n)|T|)$ time.

These considerations establish the main theorem of this section.

Theorem 7.3.6 *There is an algorithm to compute an optimal popular matching in $O(m + nf(n))$ time, where n is the number of posts, m is the sum of the lengths of the original preference lists, and $f(n)$ is the maximum time needed for a single comparison of two given weights.*

We make the not unreasonable assumption that an arithmetic or comparison operation on numbers of size $O(n)$ can be carried out in constant time, but that the complexity of such an operation on numbers of size $O(n^n)$ is no better than $O(n)$. In the case of a mincost popular matching, all weights are $O(n^2)$, so that we can take $f(n) = O(1)$. However, for rank-maximal or generous matchings, we can only assume that the weights are $O(n^n)$, so that $f(n) = O(n)$. Hence we have the following corollary.

Corollary 7.3.1 *(i) A mincost popular matching can be found in linear time. (ii) A rank-maximal popular matching and a generous popular matching can be found in $O(m + n^2)$ time.*

Improving the running time

When computing a rank-maximal or generous popular matching, the complexity of our algorithm is dominated by the time required to compute an optimal orientation of a component of G_M. To improve the complexity for these specific problems, we discard the weights and work directly with matching profiles. This enables us to compute the optimal orientation of a tree component in $O(t \log t)$ time, where t is the number of edges in the given tree component. For simplicity of presentation, this improved algorithm is described in terms of computing rank-maximal popular matchings, then we indicate the changes that need to be made to compute a generous popular matching.

Let Z be a tree component of the switching graph with sink z, let $u \neq z$ be an s-post vertex in Z, and let $v \neq u$ be a vertex such that there is a path $P(u, v)$ in Z from u to v (hence v lies on the path $P(u, z)$). Any such path $P(u, v)$ is the initial part of the switching path $P(u, z)$ starting at u.

The concept of *profile change* $C(u, v)$ along a path $P(u, v)$ quantifies the effect on the profile of applying the switching path from u, but only as far as v – we call this a *partial switching path*. (It is a genuine switching path if and only if $v = z$). Note that, unless $v = z$, applying such a partial switching path does not yield a matching, since two applicants would be matched to the post represented by v. More precisely, $C(u, v)$ is the sequence of ordered pairs $\langle (i_1, j_1), \ldots, (i_r, j_r) \rangle$, where $j_1 < j_2 < \ldots < j_r$, $i_k \neq 0$ for all k, and, for each k, there is a net change of i_k in the number of applicants assigned to their j_k^{th} choice post when the partial switching path $P(u, v)$ is applied.

Example As an illustration of the notion of profile change, consider the switching graph given in Figure 7.5 in Section 7.5. Applying the path $P(l_{16}, p_{11})$ causes a_{15} to move from his nineteenth (last resort) choice to his first choice, a_{14} from his first to his second choice, and a_{12} from his fifth to his first choice, so the resulting profile change is $\langle (1, 1), (1, 2), (-1, 5), (-1, 19) \rangle$.

We define a total order $\succ$ on profile changes (to reflect rank-maximality) in the following way. If $x = \langle (p_1, q_1), (p_2, q_2), \ldots, (p_k, q_k) \rangle$ and $y = \langle (r_1, s_1), (r_2, s_2), \ldots, (r_l, s_l) \rangle$ are profile changes ($x \neq y$), and j is the maximum index for which $(p_j, q_j) = (r_j, s_j)$, we write $x \succ y$ if and only if

- $k > l$, $j = l$, and $p_{j+1} > 0$; or
- $k < l$, $j = k$ and $r_{j+1} < 0$; or
- $j < \min(k, l)$, $q_{j+1} < s_{j+1}$ and $p_{j+1} > 0$; or
- $j < \min(k, l)$, $q_{j+1} > s_{j+1}$ and $r_{j+1} < 0$; or
- $j < \min(k, l)$, $q_{j+1} = s_{j+1}$ and $p_{j+1} > r_{j+1}$.

An *improving profile change* (with respect to $\succ_L$) is a profile change $\langle (i_1, j_1), \ldots, (i_r, j_r) \rangle$ with $i_1 > 0$. So an improving profile change leads to a better profile with respect to $\succ_L$. Moreover, if x and y are profile changes with $x \succ y$, and if applying x and y to the same profile ρ yields profiles ρ_x and ρ_y respectively, then $\rho_x \succ_L \rho_y$. The converse is also true: if x and y are profile changes, and if applying x and y to the same profile ρ yields profiles $\rho_x \succ_L \rho_y$, then $x \succ y$.

As a next step, we define the following arithmetic operation, which captures the notion of adding an ordered pair to a profile change. For a profile change $C = \langle (i_1, j_1), \ldots, (i_r, j_r) \rangle$ and ordered pair (i, j) $(i \neq 0, j > 0)$, define $C + (i, j)$ as follows:

- If $j = j_k$ and $i_k + i \neq 0$, then
  $C + (i, j) = \langle (i_1, j_1), \ldots, (i_k + i, j_k), \ldots, (i_r, j_r) \rangle$.
- If $j = j_k$ and $i_k + i = 0$, then
  $C + (i, j) = \langle (i_1, j_1), \ldots, (i_{k-1}, j_{k-1}), (i_{k+1}, j_{k+1}) \ldots, (i_r, j_r) \rangle$.
- If $j_{k-1} < j < j_k$, then
  $C + (i, j) = \langle (i_1, j_1), \ldots, (i_{k-1}, j_{k-1}), (i, j), (i_k, j_k) \ldots, (i_r, j_r) \rangle$.

The algorithm computes an optimal orientation of a tree-component Z by means of a post-order traversal, viewing Z as a tree rooted at the sink. During this traversal, *processing* a vertex v means determining the best improving profile change C_v obtainable by applying a partial switching path that ends at v, together with the starting vertex u_v of a path $P(u_v, v)$ corresponding to C_v. If no path ending at v has an improving profile change then C_v is null and u_v is undefined.

For a leaf vertex v, C_v is trivially null. For a branch node v, C_v and u_v are computed using the best improving profile change C_w for each child w of v in the tree (excluding any such w that is an f-post leaf, since no switching path can begin in such a subtree of v). Let w be a child of v, and let a be the applicant represented by the edge (w, v) of Z. Let posts v and w be the j_w^{th} and l_w^{th} choices, respectively, of applicant a, so that if a were to be re-assigned from post w to post v the profile would gain a j_w^{th} choice and lose an l_w^{th} choice. It follows at once that C_v is determined by the formula

$$C_v = \max_{w \in V_c} \{(C_w + (1, j_w)) + (-1, l_w)\}$$

where the maximum is with respect to $\succ$, and V_c is the set of children of v.

A pseudocode version of the algorithm appears in Figure 7.1.

On termination of the traversal, we have determined C_z, the best improving profile change, if any, of a switching path in Z, together with the starting point of such a path. Application of this switching

```
/* Traverse(v) returns the optimum profile change C_v and corresponding
   starting vertex u_v of a partial switching path ending at vertex v */
Traverse(v):
    if v is a leaf:
        return null
    else:
        best = null
        start = null
        for (each child w of v that is not an f-post leaf):
            (C_w, u_w) = Traverse(w)
            C = C_w + (1, j_w)) + (-1, l_w)        (1)
            if (C ≻ best):                          (2)
                best = C_w
                start = u_w
        return (best, start)
```

Figure 7.1: The postorder traversal of a tree component

path yields an optimum orientation of Z, or, in case $null$ is returned, we know that Z is already optimally oriented.

From the pseudocode in Figure 7.1, we see that the complexity of the algorithm is determined by the total number of operations involved in steps (1) and (2).

To deal with step (1), we represent a profile change by a balanced binary tree B whose nodes contain the pairs (i, j), ordered by the second member of the pair. The $+$ operation on profile changes involves amendment, insertion, or deletion of a node in B, which can be accomplished in time logarithmic in the size of B. Since the number of pairs in a profile change cannot exceed the number of edges in Z, this is $O(\log t)$, and since step (1) is executed at most t times, the total number of operations carried out by step (1), summed over all iterations, is $O(t \log t)$.

As far as step (2) is concerned, we first note that two profile changes, involving c_1 and c_2 pairs, with $c_1 < c_2$, can be compared in $O(c_1)$ time. So the cost of a comparison is linear in the size of each of the balanced trees involved. Once a profile change is the 'loser' in such a comparison, the

balanced tree representing it is never used again. Hence the cost of all such comparisons is linear in s, the sum of the sizes of all of the balanced trees constructed by the algorithm. But each node in a balanced tree originates from one or more edges in Z, and each edge in Z contributes to at most one node in one balanced tree. So s is bounded by the number of edges in Z, and hence the total number of operations in step (2), summed over all iterations, is $O(t)$.

It follows that the postorder traversal of a tree component Z with t edges can be completed in $O(t \log t)$ time, and once the optimal switching path is found it can be applied in $O(t)$ time. Hence, since the total number of edges in all tree components is $O(n)$, this process can be applied to all tree components in $O(n \log n)$ time.

Finally, we observe that the optimal orientation of each cycle component can be computed efficiently. For a cycle component Y with switching cycle C, we need only check if the profile change obtained by applying C is an improving profile change, and, if so, C is applied, otherwise, Y is already optimally oriented. Hence, the optimal orientation of a cycle component Y with y edges can be computed in $O(y)$ time. This process can therefore be applied to each cycle component in $O(n)$ time. Bearing in mind that the preprocessing phase of the algorithm requires $O(n+m)$ time, we conclude that a rank-maximal popular matching can be found in $O(n \log n + m)$ time.

The algorithm can be amended to find a generous popular matching by making appropriate changes to the definition of the order relation on profile changes, as follows.

We define a total order $\prec$ on profile changes (to reflect generosity) in the following way. If $x = \langle (p_1, q_1), (p_2, q_2), \ldots, (p_k, q_k) \rangle$ and $y = \langle (r_1, s_1), (r_2, s_2), \ldots, (r_l, s_l) \rangle$ are profile changes ($x \neq y$), we write $x \prec y$ if and only if

- $q_k > s_k$ and $p_k < 0$; or
- $q_k < s_l$ and $r_l > 0$; or
- j is the maximum value such that $(p_{k-j}, q_{k-j}) = (r_{l-j}, s_{l-j})$, and
  - $q_{k-j-1} > s_{l-j-1}$ and $p_{k-j-1} < 0$; or
  - $q_{k-j-1} < s_{l-j-1}$ and $r_{l-j-1} > 0$; or
  - $q_{k-j-1} = s_{l-j-1}$ and $p_{k-j-1} < r_{l-j-1}$.

An *improving profile change* (with respect to $\prec_G$) is a profile change $\langle (i_1, j_1), \ldots, (i_r, j_r) \rangle$ with $i_r < 0$. So an improving profile change leads to a better profile with respect to $\prec_G$. Moreover, if x and y are profile changes with $x \prec y$, and if applying x and y to the same profile ρ yields profiles ρ_x and ρ_y respectively, then $\rho_x \prec_G \rho_y$.

It is now straightforward to verify that an amended version of the postorder traversal algorithm that uses $\prec$ rather than $\succ$ to compare profile changes will determine a switching path in a tree component of the switching graph that is optimal with respect to generosity. All other aspects of the algorithm, and its analysis, are identical to the rank-maximal case. It follows that a generous popular matching can be computed in $O(n \log n + m)$ time.

7.4 Conclusions and open problems

This chapter has characterized the structure of the set of popular matchings for a POP-M instance in terms of the so-called switching graph. This characterization leads to efficient algorithms for a range of extensions of the basic problem.

We have assumed throughout that the applicants' preference lists are strictly ordered. Abraham et al [5] considered also the case where the preference lists may contain ties, and gave a $O(\sqrt{n}m)$ time algorithm to determine a popular matching in that case. It is an open question to provide a neat characterization of the structure for this more general problem, and to exploit any such structure to give an analogous set of efficient algorithms. Note, however, that algorithms for, say, counting or enumerating all popular matchings would necessarily subsume such algorithms for arbitrary bipartite graphs. Consider, for example, a POP-M instance in which every applicant's preference list is a single tie – the set of popular matchings is exactly the set of maximum matchings. Therefore, this is likely to be considerably more involved than in the no-ties case.

7.5 Example

To illustrate the notion of the switching graph and the implications of applying switching paths and cycles, we provide a detailed example. Figure 7.2 shows the full preference lists (without last resorts) for a POP-M instance I. The instance consists of 16 applicants $a_1, a_2, \ldots, a_{16}$ and 18 posts

$p_1, p_2, \ldots, p_{18}$. The set of f-posts of the instance is $\{p_1, p_4, p_5, p_6, p_{10}, p_{11}, p_{15}, p_{17}\}$ and, after the inclusion of the last-resort posts, the set of s-posts is $\{p_2, p_3, p_7, p_8, p_9, p_{12}, p_{13}, p_{14}, l_{15}, p_{18}\}$. Note that post l_{15} is a_{15}'s last resort post. This is the only last resort post that is also an s-post for this instance.

Figure 7.3 shows the reduced instance of I, obtained by removing all posts from an applicant a's preference list except for $f(a)$ and $s(a)$. This instance admits several popular matchings; one such popular matching M is denoted by underlining.

The switching graph G_M for M is given in Figure 7.4. The graph consists of three components, one cycle component and two tree components containing posts p_{18} and p_9 as sinks, respectively. The cycle component has a switching cycle containing posts p_1, p_2, p_4, p_3. The switching graph has a total of 4 switching paths, all of which are contained in the larger tree component. These switching paths can easily be identified by recalling that the path to the sink from any other s-post vertex is a switching path. Hence, the paths to p_9 starting at p_{12}, p_{14}, p_{13}, and l_{15} are the switching paths for this instance. The other tree component has no switching path; it represents a fixed pair – applicant a_{16} must be matched to post p_{17} (and post p_{18} is unfilled) in every popular matching. Note that the (applicant,post) pairs in the "tails" of the cycle component, namely the pairs (a_7, p_7), (a_6, p_6), (a_8, p_8), and (a_5, p_5), are also fixed pairs.

If the switching cycle is applied, then a_1 becomes matched to p_2, a_2 to p_4, a_3 to p_3, and a_4 to p_1, giving a second popular matching M'. All other applicants are matched to the same posts in M' as in M. Figure 7.5 shows the change in the switching graph when the switching cycle is applied; the direction of each arc in the cycle is reversed while all other arcs in G_M are unchanged.

If, instead of applying the switching cycle, we apply a switching path, say the path beginning at post p_{13}, applicant a_{12} becomes matched to p_{11}, and applicant a_{10} to post p_9 (which was previously unoccupied in M), giving a third popular matching M''. All other applicants remain matched to the same posts in M'' as in M. Figure 7.6 shows the change in the switching graph resulting from the application of this switching path; the direction of each arc on the path from p_{13} to p_9 is reversed, so that p_{13} becomes the new sink vertex in this tree component.

As an illustration of Theorem 7.3.1, this problem instance has a total of ten popular matchings, resulting from the five possible orientations of the larger tree component and the two possible orientations of the cycle component.

a_1 :	p_1	p_4	p_{10}	p_{11}	p_2	p_6	p_8		
a_2 :	p_4	p_6	p_{11}	p_{17}	p_2	p_5	p_{12}	p_{13}	p_{10}
a_3 :	p_4	p_1	p_3	p_{15}	p_8	p_{16}			
a_4 :	p_1	p_{11}	p_6	p_3	p_{15}				
a_5 :	p_5	p_1	p_{11}	p_4	p_2	p_6	p_{14}		
a_6 :	p_6	p_{10}	p_{11}	p_3	p_1	p_6			
a_7 :	p_6	p_7							
a_8 :	p_6	p_5	p_8	p_{16}					
a_9 :	p_{10}	p_{11}	p_4	p_9	p_2	p_1	p_{18}		
a_{10} :	p_{11}	p_9	p_7	p_1	p_7	p_{12}			
a_{11} :	p_{10}	p_4	p_{17}	p_6	p_{12}	p_7	p_{13}		
a_{12} :	p_{11}	p_5	p_6	p_{15}	p_{13}	p_1	p_9	p_{18}	
a_{13} :	p_{11}	p_{15}	p_4	p_{14}	p_3				
a_{14} :	p_{15}	p_{13}	p_1	p_4	p_9	p_8			
a_{15} :	p_{15}	p_6							
a_{16} :	p_{17}	p_{15}	p_5	p_4	p_{18}	p_8	p_9	p_{13}	

Figure 7.2: A popular matching instance I

a_1 :	$\underline{p_1}$	p_2
a_2 :	p_4	$\underline{p_2}$
a_3 :	$\underline{p_4}$	p_3
a_4 :	p_1	$\underline{p_3}$
a_5 :	$\underline{p_5}$	p_2
a_6 :	$\underline{p_6}$	p_3
a_7 :	p_6	$\underline{p_7}$
a_8 :	p_6	$\underline{p_8}$
a_9 :	$\underline{p_{10}}$	p_9
a_{10} :	$\underline{p_{11}}$	p_9
a_{11} :	p_{10}	$\underline{p_{12}}$
a_{12} :	p_{11}	$\underline{p_{13}}$
a_{13} :	p_{11}	$\underline{p_{14}}$
a_{14} :	$\underline{p_{15}}$	p_{13}
a_{15} :	p_{15}	$\underline{l_{15}}$
a_{16} :	$\underline{p_{17}}$	p_{18}

Figure 7.3: The reduced instance of I with popular matching M denoted by underlining

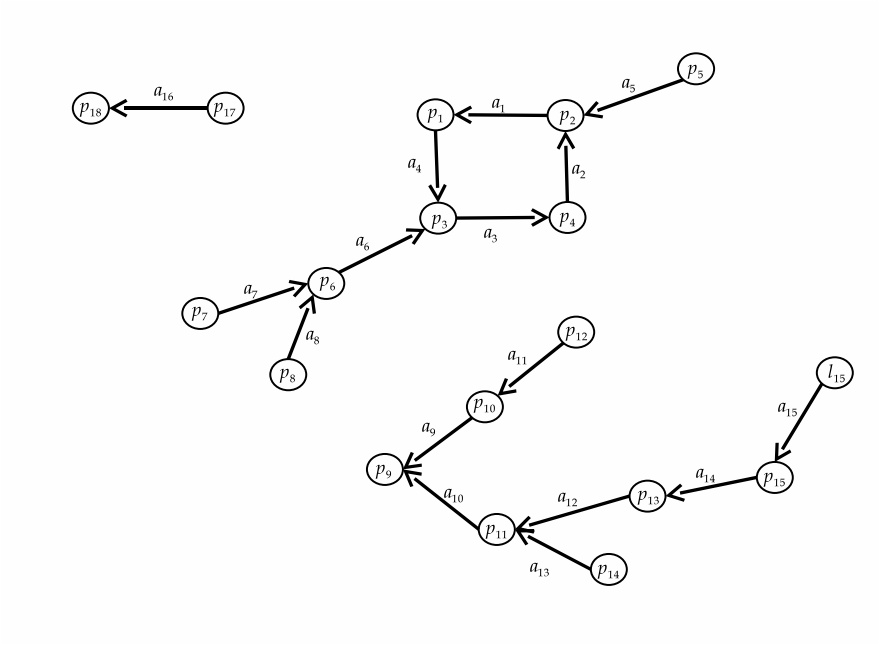

Figure 7.4: The switching graph G_M for popular matching M

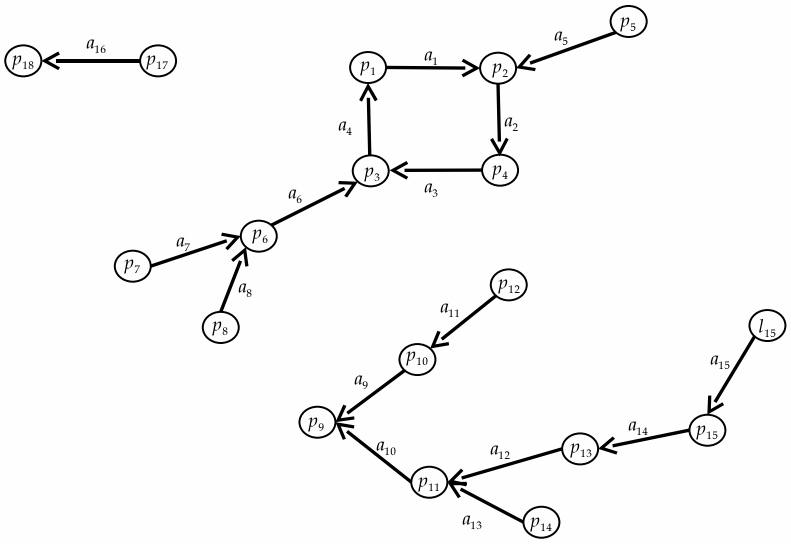

Figure 7.5: The switching graph for the popular matching M' obtained by applying the switching cycle in G_M

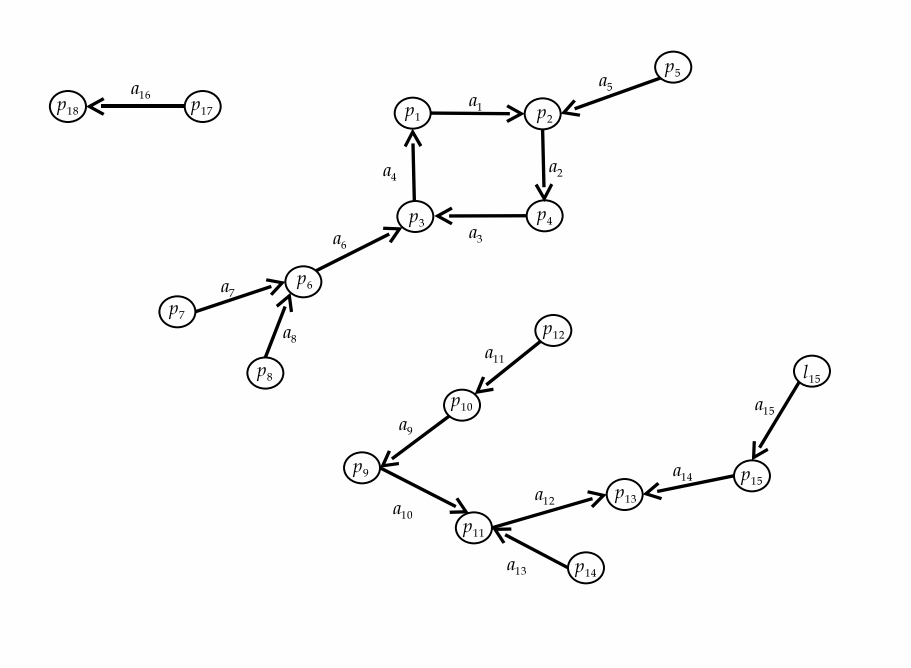

Figure 7.6: The switching graph for the popular matching M'' obtained by applying the switching path beginning at p_{13} in G_M

Chapter 8

Conclusions and Future Work

8.1 Introduction

This thesis has presented a number of new algorithmic results for five specific problem domains from the realm of both full and partial preference information. We emphasized a *structural* approach to each individual problem in the sense that each new algorithmic result presented relied crucially on some structural property implicitly embedded in the problem.

This final chapter is a summary of the results presented in this thesis, along with a collection of open problems and ideas of future work arising from the problems we have studied.

8.2 Approximation algorithms for MAX-SMTI

In Chapter 3, we considered the well-studied NP-hard variant of the stable marriage problem with ties and incomplete lists in which we seek to find stable matchings that are as large as possible (MAX-SMTI).

Roughly speaking, our performance guarantee is based on the following argument: if M_{opt} is a maximum cardinality stable matching, and M the stable matching returned by our algorithm, then the symmetric difference $M_{opt} \oplus M$ can have no augmenting paths of length at most three with the two end edges belonging to M_{opt}. We then conclude that the ratio of M_{opt} edges to M edges

is therefore at most $3/2$ in every component of $M_{opt} \oplus M$. The performance guarantee of $3/2$ follows.

A next step, therefore, could be to devise an approximation algorithm that somehow also elminates the presence of at least 'some' augmenting paths of length at most *five*. For example, perhaps there is an algorithm for which it can be argued that $M_{opt} \oplus M$ can have no augmenting paths of length at most three with the two end edges belonging to M_{opt}, as in our algorithm in Chapter 3, but also has the property that at most, say, $1/10$ of the components cannot be augmenting paths of length at most five. This would yield a constant performance strictly better than $3/2$.

On the practical side, it would be interesting to see the performance of this algorithm in practice. There has been some work done in the comparison of such approximation algorithms in practice [50] – how does this algorithm measure up? We conjecture that our algorithm performs very well when each of the men's preference lists consist of a few large ties. The following is some intuition as to why this may be the case. Notice that for an SMTI instance in which every man's preference list consists of a single tie, an arbitrary maximum matching is a maximum cardinality stable matching. Given such an input, our algorithm will go immediately to Phase 2 (assuming every man's preference list has length at least two). The algorithm will return a maximum cardinality matching, which is the optimal solution. It would also be interesting to know if there is a formal argument to show that the algorithm does indeed perform well when the men's ties have this property.

8.3 Sex-equal stable matchings

In Chapter 4, we studied the strongly NP-hard problem of finding a sex-equal stable matching (SESM). We gave a complete characterisation of the parameterized complexity of (α, β)-SESM (defined in Chapter 4). When the preference lists on one side are of length at most two, the problem is solvable in polynomial time, but, if the preference lists on either side are allowed instead to be of length three or greater, then the problem is $W[1]$-hard. Additionally, we presented an exact low-order exponential-time algorithm for SESM in which the men's preference lists are bounded in length by a constant l, and the lengths of the women's preference lists are unrestricted.

As far as we know, our exponential-time algorithm is the first 'moderately' exponential-time algorithm for any computationally hard stable marriage variant. Since the stable matching literature

is rife with hard stable matching problems, we believe that this opens the door to the exploration of the existence of other exponential-time algorithms with 'good' theoretical running times for a whole host of stable marriage problems. As an obvious next step, one could consider searching for an exact algorithm for SESM with no bound on the lengths of the preference lists.

A similar problem to SESM is a problem described in the PhD thesis of Feder [27], called the *balanced stable matching problem*, the goal of which is to find a stable matching M that minimizes

$$\max\Big\{\sum_{(m,w)\in M} p_m(w), \sum_{(m,w)\in M} p_w(m)\Big\}$$

over all $M \in \mathcal{M}$. Intuitively, a balanced stable matching minimizes the unhappiness of the most unhappy group of people (the men or the women). Feder [27] proved that this problem is NP-hard. Is the balanced stable matching problem solvable by similar techniques to that presented in Chapter 4?

A corollary to the dynamic programming algorithm given in Section 4.9 is that SESM is solvable in polynomial-time whenever the underlying graph of the rotation posed is series-parallel. A next step could be to ask if it is the case that SESM can be solved in polynomial-time whenever the underlying graph has bounded *treewidth* (see, for example, one of the many surveys by Bodlaender [10] for the relevant background on treewidth).

This leads to an even larger question: what stable matching problems can be solved efficiently (or more efficiently) when the underlying graph of the rotation poset has bounded treewidth? One particularly interesting problem could be the median stable matching problem (discussed in Section 2.2.4). This problem is not even known to be in NP. Does the bounded treewidth property make this problem easier?

Finally, we remark that very recently an improvement in the Edwards and Farr theorem has been made [24]. This probably implies an improvement in the upper bound of the running time described in Theorem 4.10.1. In fact, any further improvement upon the bound given in Theorem 4.10.1 will likely imply an improved result for the exponential-time algorithm given in Chapter 4.

8.4 Keeping couples together

Chapter 5 studies the hospitals / residents problem with couples (HRC) in which pairs of residents are allowed to form *couples*, with some form of joint preference lists that express the couples' preference for where the members of the couple should both be matched. We considered a natural restriction of HRC in which the members of a couple have individual preference lists over hospitals, and the couples form joint preference lists that are, in a formal sense, *consistent* with these individual preference lists. We gave an appropriate stability definition and showed that the problem of deciding whether a stable matching exists is NP-complete, even if each resident's preference list has length at most three and each hospital has capacity at most two. In contrast to this result, we gave a linear-time algorithm to find a stable matching (or report that none exists) when stability is defined in terms of the classical Gale-Shapley concept. This algorithm makes no assumptions about the preference lists or capacities of the hospitals. Finally, for an alternative formulation of our restriction of HRC, which we call the *hospitals / residents problem with sizes* (HRS), we gave a linear-time algorithm that always finds a stable matching for the case that hospital preference lists are of length at most two, and where hospital capacities can be arbitrary.

There are a couple of immediate open questions that arise from our results. Firstly, what is the complexity of $(2, \infty)$-HRS? We conjecture that this problem is solvable in polynomial-time, but the solution appears to be non-trivial.

Another area of approach could be to reconsider our definition of HRS. As it stands, our definition favors 'quality over quantity' in that a single resident could displace a large group of inferior residents. An alternative defintion of HRS would be to instead allow for 'quantity over quality', so that a hospital's primary concern is to ensure that its occupancy is as high as possible. This gives rise to an alternative stability definition which is obtained from the one given for HRS in Section 5.2 by modifying Condition (2) as follows:

2. $O_j^M + s_i \leq c_j$, or h_j prefers r_i to residents $r_{k_1}, \ldots r_{k_t} \in M(h_j)$ such that

$$s_i \geq \sum_{p=1}^{t} s_{k_p} \quad \text{and} \quad O_j^M + s_i - \sum_{p=1}^{t} s_{k_p} \leq c_j.$$

Is this problem solvable in polynomial-time?

Lastly, in the spirit of'keeping couples together', we could consider a different definition of a blocking pair where a resident r and a hospital h do not form a blocking pair if moving r to h would force the couple involving r to break apart. Intuitively, this means that r makes keeping his couple together a priority over his own personal benefit of moving to hospital h. We leave it as an open question to give a proper formalization of this problem, and to determine its complexity [1].

8.5 3D-stable matchings

In Chapter 6, we explored the generalisation of the stable marriage problem to *three* sets, so that we attempt to match men, women, and *dogs* into triples (3DSM). We examined 3DSMI under so-called weak- and strong-stability, and show that, in general, finding a strongly stable matching is NP-hard in the 3DSM setting (and hence is also NP-hard for 3DSMI), and that finding a weakly stable matching is NP-hard in the 3DSMI setting.

The key ingredient in our constructions was a specially constructed instance of 3DSM that we call a 9-*Sun*. When we view this instance in terms of its underlying graph, it is easy to see that the special structural nature of this instance makes the creation of a stable matching impossible.

The 9-Sun is the smallest example that we can find of a 3DSMI instance with no weakly stable matching. Is there a smaller example? In the case of strong stability, one can construct smaller examples (with $n = 4$) that admit no strongly stable matching.

There is, in fact, no known instance of 3DSM (i.e. complete preference lists) that does not admit at least one weakly stable matching. A natural place to try to construct a counterexample would be to try to complete the preference lists of the 9-Sun in a way that does not introduce a weakly stable matching. However, we conjecture that this is not possible for this particular example. Is there a 3DSM instance with no weakly stable matching? A larger question is to determine whether there is a polynomial-time algorithm to find a weakly stable matching or report that none exists, given an instance of 3DSM.

It is very uncommon to find a matching problem with preferences for which there is no clear way to extend a hardness result to complete preference lists (in fact, we know of no other such problem).

[1] We thank Paul Goldberg for suggesting this problem.

Could it really be the case that when one attempts to complete the preference lists of a 3DSMI instance, one cannot avoid introducing a weakly stable matching?

8.6 Popular matchings

Finally, Chapter 7 considers the popular matching problem (POP-M) in the context of matching a set of applicants with preference lists to a set of posts without preferences. Our contribution was to provide a characterization of the set of popular matchings for an arbitrary POP-M instance in terms of a new structure called the *switching graph*, a directed graph computable in linear time from the preference lists. We showed that this structure can be exploited to yield efficient algorithms for a range of associated problems, including the counting and enumeration of the set of popular matchings, generation of a popular matching uniformly at random, finding all applicant-post pairs that can occur in a popular matching, and computing popular matchings that satisfy various additional optimality criteria.

An obvious open question would be to consider the case in which the preference lists of the applicants may be allowed to have ties. Is there a generalization of the switching graph (or a similar structure) to characterize the set of popular matchings in the presence of ties? We suspect that this is a particularly difficult problem, as such a structure would subsume a structure for the set of all maximum matchings of an arbitrary bipartite graph. Consider: when each applicant's list is a single tie, then the set of popular matchings is exactly the set of maximum matchings. Therefore, we suspect that this problem is considerably more difficult than that of the no-ties case.

A different approach could be to consider the structure of the set of popular matchings without ties, but with capacities (associated with the posts) and/or weights (associated with the applicants). Is there a neat structural characterization of the set of popular matchings in this case?

On a different note, it would be interesting to know if there are faster algorithm than our $O(n \log n + m)$-time method for finding rank-maximal or generous popular matchings. One possible approach from the negative side could be to attempt a reduction from a problem such as Element Uniqueness, with a known time complexity lower bound of $\Omega(n \log n)$.

Appendix A

NP-completeness of (3,3)-COM-SMTI

We prove that MAX-SMTI (defined in Section 2.2.5) remains NP-complete in a particularly restricted setting. We are interested in this result is because it is particularly useful to us in some of the NP-hardness reductions we present in Chapters 5 6. Define $(3, 3)$-COM-SMTI to be the problem of deciding whether a complete weakly stable matching exists (i.e., a weakly stable matching that matches every agent), given an instance of SMTI in which each preference list is of length at most three, every woman's preference list is strictly ordered, and each man's preference list is either strictly ordered or is a tie of length two (all of these conditions holding simultaneously). Our proof of this theorem is largely based on a reduction of Irving et al [51] with some small but non-trivial modifications.

Theorem A.0.1 $(3, 3)$*-COM-SMTI is NP-complete*

Proof We reduce from a a restricted version of SAT. Let (2,2)-E3-SAT denote the problem of deciding, given a Boolean formula B in CNF in which each clause contains exactly 3 literals and, for each variable v_i, each of literals v_i and $\bar{v_i}$ appears exactly twice in B, whether B is satisfiable. Berman et al. [8] showed that (2,2)-E3-SAT is NP-complete.

Hence let B be an instance of (2,2)-E3-SAT. Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ be the set of variables and clauses respectively in B. Then for each $v_i \in V$, each of literals v_i and $\bar{v_i}$ appears exactly twice in B. (Hence $m = \frac{4n}{3}$.) Also $|c_j| = 3$ for each $c_j \in C$. We form an instance I of (3,3)-COM-SMTI as follows. The set of men in I is $X \cup P \cup U \cup Q$, where

$$
\begin{array}{rll}
x_{4i}: & y_{4i}\ c(x_{4i})\ y_{4i+1} & (0 \le i \le n-1) \\
x_{4i+1}: & y_{4i+1}\ c(x_{4i+1})\ y_{4i+2} & (0 \le i \le n-1) \\
x_{4i+2}: & y_{4i+3}\ c(x_{4i+2})\ y_{4i+2} & (0 \le i \le n-1) \\
x_{4i+3}: & y_{4i}\ c(x_{4i+3})\ y_{4i+3} & (0 \le i \le n-1) \\
p_j^s: & z_j^s\ c_j^s & (1 \le j \le m \wedge 1 \le s \le 3) \\
u_j^s: & z_j^s\ w_j & (1 \le j \le m \wedge 1 \le s \le 3) \\
q_j: & c_j^1\ c_j^2\ c_j^3 & (1 \le j \le m) \\
 & & \\
y_{4i}: & (x_{4i}\ x_{4i+3}) & (0 \le i \le n-1) \\
y_{4i+1}: & (x_{4i}\ x_{4i+1}) & (0 \le i \le n-1) \\
y_{4i+2}: & (x_{4i+1}\ x_{4i+2}) & (0 \le i \le n-1) \\
y_{4i+3}: & (x_{4i+2}\ x_{4i+3}) & (0 \le i \le n-1) \\
c_j^s: & p_j^s\ x(c_j^s)\ q_j & (1 \le j \le m \wedge 1 \le s \le 3) \\
w_j: & u_j^1\ u_j^2\ u_j^3 & (1 \le j \le m) \\
z_j^s: & (p_j^s\ u_j^s) & (1 \le j \le m \wedge 1 \le s \le 3)
\end{array}
$$

Figure A.1: Preference lists in the constructed instance of (3,3)-COM-SMTI

$X = \cup_{i=0}^{n-1} X_i$, $X_i = \{x_{4i+r} : 0 \le r \le 3\}$ $(0 \le i \le n-1)$, $P = \cup_{j=1}^{m} P_j$, $P_j = \{p_j^1, p_j^2, p_j^3\}$ $(1 \le j \le m)$, $U = \cup_{j=1}^{m} U_j$, $U_j = \{u_j^1, u_j^2, u_j^3\}$ $(1 \le j \le m)$, and $Q = \{q_j : 1 \le j \le m\}$. The set of women in I is $Y \cup C' \cup W \cup Z$, where $Y = \cup_{i=0}^{n-1} Y_i$, $Y_i = \{y_{4i+r} : 0 \le r \le 3\}$ $(0 \le i \le n-1)$, $C' = \{c_j^s : c_j \in C \wedge 1 \le s \le 3\}$, $W = \{w_j : 1 \le j \le m\}$, $Z = \cup_{j=1}^{m} Z_j$ and $Z_j = \{z_j^1, z_j^2, z_j^3\}$ $(1 \le j \le m)$.

The preference lists of the men and women in I are shown in Figure A.1. In a given preference list, entries within round brackets are tied. In the preference list of an agent $x_{4i+r} \in X$ ($0 \le i \le n-1$ and $r \in \{0, 1\}$), the symbol $c(x_{4i+r})$ denotes the woman $c_j^s \in C'$ such that the $(r+1)$th occurrence of literal v_i appears at position s of c_j. Similarly if $r \in \{2, 3\}$ then the symbol $c(x_{4i+r})$ denotes the woman $c_j^s \in C'$ such that the $(r-1)$th occurrence of literal $\bar{v}_i$ appears at position s of c_j. Also in the preference list of an agent $c_j^s \in C'$, if literal v_i appears at position s of clause $c_j \in C$, the symbol $x(c_j^s)$ denotes the man x_{4i+r-1}, where $r = 1, 2$ according as this is the first or second occurrence of literal v_i in B. Otherwise if literal $\bar{v}_i$ appears at position s of clause $c_j \in C$, the symbol $x(c_j^s)$ denotes the man x_{4i+r+1}, where $r = 1, 2$ according as this is the first or second occurrence of literal $\bar{v}_i$ in B. Clearly each preference list is of length at most 3, the men's lists are strictly ordered, and each woman's list is either strictly ordered or is a tie of length 2.

For each i $(0 \le i \le n-1)$, let $T_i = \{(x_{4i+r}, y_{4i+r}) : 0 \le r \le 3\}$ and $F_i = \{(x_{4i+r}, y_{4i+r+1})\} : 0 \le r \le 3\}$, where addition is taken modulo 4.

We claim that B is satisfiable if and only if I admits a complete stable matching.

For, let f be a satisfying truth assignment of B. Define a complete matching M in I as follows. For each variable $v_i \in V$, if v_i is true under f, add the pairs in T_i to M, otherwise add the pairs in F_i to M. Now let $c_j \in C$. As c_j contains a literal that is true under f, let $s \in \{1, 2, 3\}$ denote the position of c_j in which this literal occurs. Add the pairs (p_j^k, c_j^k) and (u_j^k, z_j^k) ($1 \leq k \neq s \leq 3$), (p_j^s, z_j^s), (q_j, c_j^s) and (u_j^s, w_j) to M.

As M is a complete matching in I, clearly no woman in $Y \cup Z$ can be involved in a blocking pair of M in I. Nor can a man in $P \cup U$ (since he can only potentially prefer a woman in Z), nor a man in Q (since he can only potentially prefer a woman in C, who ranks him last), nor a woman in W (since she can only potentially prefer a man in U, who ranks him last). Now suppose that $(x_{4i+r}, c(x_{4i+r}))$ blocks M, where $0 \leq i \leq n-1$ and $0 \leq r \leq 3$. Let $c_j^s = c(x_{4i+r})$, where $1 \leq j \leq m$ and $1 \leq s \leq 3$. Then $(q_j, c_j^s) \in M$. If $r \in \{0, 1\}$ then $(x_{4i+r}, y_{4i+r+1}) \in M$, so that v_i is false under f. But literal v_i occurs in c_j, a contradiction, since literal v_i was supposed to be true under f by construction of M. Hence $r \in \{2, 3\}$ and $(x_{4i+r}, y_{4i+r}) \in M$, so that v_i is true under f. But literal $\bar{v_i}$ occurs in c_j, a contradiction, since literal $\bar{v_i}$ was supposed to be true under f by construction of M. Hence M is stable in I.

Conversely suppose that M is a complete stable matching in I. We form a truth assignment f in B as follows. For each i ($0 \leq i \leq n-1$), if $M \cap (X_i \times Y_i) = T_i$, set v_i to be true under f. Otherwise $M \cap (X_i \times Y_i) = F_i$, in which case we set v_i to be false under f.

Now let c_j be a clause in C ($1 \leq j \leq m$). There exists some s ($1 \leq s \leq 3$) such that $(q_j, c_j^s) \in M$. Let $x_{4i+r} = x(c_j^s)$ for some i ($0 \leq i \leq n-1$) and r ($0 \leq r \leq 3$). If $r \in \{0, 1\}$ then $(x_{4i+r}, y_{4i+r}) \in M$ by the stability of M. Thus variable v_i is true under f, and hence clause c_j is true under f, since literal v_i occurs in this clause. If $r \in \{2, 3\}$ then $(x_{4i+r}, y_{4i+r+1}) \in M$ (where addition is taken modulo 4) by the stability of M. Thus variable v_i is false under f, and hence clause c_j is true under f, since literal $\bar{v_i}$ occurs in this clause. Hence f is a satisfying truth assignment of B.

□

Bibliography

[1] A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[2] D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter-exchange markets: Enabling nationwide kidney exchanges. *In Proceedings of ACM-EC 2007: the Eighth ACM Conference on Electronic Commerce*, 2007.

[3] D.J. Abraham, P. Biró, and D.F. Manlove. "Almost stable" matchings in the Roommates problem. In *Proceedings of WAOA '05: the 3rd Workshop on Approximation and Online Algorithms*, volume 3879 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[4] D.J. Abraham, K. Cechlárová, D.F. Manlove, and K. Mehlhorn. Pareto optimality in house allocation problems. In *Proceedings of ISAAC 2004: the 15th Annual International Symposium on Algorithms and Computation*, volume 3341 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2004.

[5] D.J. Abraham, R.W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37:1030–1045, 2007.

[6] D.J. Abraham and T. Kavitha. Dynamic matching markets and voting paths. In *Proceedings of SWAT 2006: the 10th Scandinavian Workshop on Algorithm Theory*, volume 4059 of *Lecture Notes in Computer Science*, pages 65–76. Springer, 2006.

[7] A. Alkan. Non-existence of stable threesome matchings. *Mathematical Social Sciences*, 16:207–209, 1988.

[8] P. Berman, M. Karpinski, and Alexander D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. Electronic Colloquium on Computational Complexity Report, number 49, 2003.

[9] P. Biró and K. Cechlárová. Inapproximability of the kidney exchange problem. *Inf. Process. Lett.*, 101(5):199–202, 2007.

[10] H. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2), 1993.

[11] E. Boros, V. Gurvich, S. Jaslar, and D. Krasner. Stable matchings in three-sided systems with cyclic preferences. *Discrete Mathematics*, 289:1–10, 2004.

[12] `http://www.bostonpublicschools.org/assignment/` (boston public schools website.

[13] D. Cantala. Matching markets: the particular case of couples. *Economics Bulletin*, 3(45):1–11, 2004.

[14] K. Cechlárová, T. Fleiner, and D. Manlove. The kidney exchange game. In *Proceedings of SOR '05: the 8th International Symposium on Operations Research in Slovenia*, pages 77–83, 2005. IM Preprint series A, no. 5/2005, PJ Šafárik University, Faculty of Science, Institute of Mathematics.

[15] K. Cechlárová and V. Lacko. The kidney exchange game: How hard is to find a donor? *IM Preprint*, 4/2006, 2006.

[16] P. Chebolu. Personal communication, 2009.

[17] A. Checker. The national intern and resident matching program, 1966-1972. *Journal of Medical Education*, 48:107–109, 1973.

[18] C. Cheng. The generalized median stable matchings: Finding them is not that easy. In *Proceedings of LATIN 2008: the 8th Latin-American Theoretical INformatics Symposium*, volume 4957 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2008.

[19] C. Cheng, E. McDermid, and I. Suzuki. A unified approach to finding good stable matchings in the hospitals/residents setting. *Theoretical Computer Science*, 400(1-3):84–99, 2008.

[20] V.I. Danilov. Existence of stable matchings in some three-sided systems. *Mathematical Social Sciences*, 46:145–148, 2003.

[21] B.C. Dean, M.X. Goemans, and N. Immorlica. The unsplittable stable marriage problem. In *Proceedings of IFIP TCS 2006: the Fourth IFIP International Conference on Theoretical*

Computer Science, volume 209 of *IFIP International Federation for Information Processing*, pages 65–75. Springer, 2006.

[22] R. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, 1999.

[23] B. Dutta and J. Massó. Stability of matchings when individuals have preferences over colleagues. *Journal of Economic Theory*, 75:464–475, 1997.

[24] K. Edwards. Personal communication, 2010.

[25] K. Edwards and G. Farr. Planarization and fragmentability of some classes of graphs. *Discrete Mathematics*, 308:2396–2406, 2008.

[26] K. Eriksson, J. Sjostrand, and P. Strimling. Three dimensional stable matching with cyclic preferences. *Mathematical Social Sciences*, 52:77–87, 2006.

[27] T. Feder. *Stable Networks and Product Graphs*. PhD thesis, Stanford University, 1990. Published in *Memoirs of the American Mathematical Society*, vol. 116, no. 555, 1995.

[28] T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45:233–284, 1992.

[29] H.N. Gabow and R.E. Tarjan. Faster scaling algorithms for general graph-matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

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

[31] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

[32] P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

[33] I.P. Gent, R.W. Irving, D.F. Manlove, P. Prosser, and B.M. Smith. A constraint programming approach to the stable marriage problem. In *Proceedings of CP '01: the 7th International Conference on Principles and Practice of Constraint Programming*, volume 2239 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2001.

[34] D. Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987.

[35] 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.

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

[37] M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Improved approximation of the stable marriage problem. *ACM Transactions on Algorithms*, 3(3), 2007.

[38] M.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(1-3):431–447, 2003.

[39] M.M. Halldórsson, K. Iwama, S. Miyazaki, and H. Yanagisawa. Randomized approximation of the stable marriage problem. *Theoretical Computer Science*, 325(3):439–465, 2004.

[40] C.-C. Huang. Two's company, three's a crowd: stable family and threesome roommate problems. In *Proceedings of ESA '07: the 15th Annual European Symposium on Algorithms*, volume 4698 of *Lecture Notes in Computer Science*, pages 558–569. Springer, 2007.

[41] C-C. Huang, T. Kavitha, D. Michail, and M. Nasre. Bounded unpopularity matchings. In *Proceedings of SWAT 2008, the 12th Scandinavian Workshop on Algorithm Theory*, volume 5124 of *Lecture Notes in Computer Science*, pages 127–137. Springer, 2008.

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

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

[44] 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, 1998.

[45] R.W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.

[46] R.W. Irving and P. Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

[47] R.W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 34(3):532–543, 1987.

[48] R.W. Irving and D. Manlove. Approximation algorithms for hard variants of the stable marriage and hospitals/residents problem. *Journal of Combinatorial Optimization*, 16(3):279–292, 2008.

[49] R.W. Irving and D.F. Manlove. The Stable Roommates Problem with Ties. *Journal of Algorithms*, 43:85–105, 2002.

[50] R.W. Irving and D.F. Manlove. Finding large stable matchings. *ACM Journal of Experimental Algorithmics*, vol. 14 section 1 article no. 2, 30 pages, 2009.

[51] R.W. Irving, D.F. Manlove, and G. O'Malley. Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms*, 7(2):213–219, 2009.

[52] R.W. Irving and S. Scott. The stable fixtures problem. *Discrete Applied Mathematics*, 155(2118-2129), 2007.

[53] K. Iwama, S. Miyazaki, and K. Okamoto. A $\left(2 - c\frac{\log n}{n}\right)$-approximation algorithm for the stable marriage problem. In *Proceedings of SWAT 2004: the 9th Scandinavian Workshop on Algorithm Theory*, volume 3111 of *Lecture Notes in Computer Science*, pages 349–361. Springer, 2004.

[54] K. Iwama, S. Miyazaki, and N. Yamauchi. A 1.875–approximation algorithm for the stable marriage problem. In *Proceedings of SODA 2007: the Eighteenth ACM/SIAM Symposium on Discrete Algorithms*, pages 288–297, 2007.

[55] K. Iwama, S. Miyazaki, and N. Yamauchi. A $\left(2 - c\frac{1}{\sqrt{n}}\right)$-approximation algorithm for the stable marriage problem. *Algorithmica*, 51(3):342–356, 2008.

[56] K. Iwama, S. Miyazaki, and H. Yanagisawa. Approximation algorithms for the sex-equal stable marriage problem. In *Proceedings of WADS 2007: the Tenth International Workshop on Algorithms and Data Structures*, volume 4619 of *Lecture Notes in Computer Science*, pages 201–213. Springer, 2007.

[57] D.S. Johnson and K.A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

[58] A. Kato. Complexity of the sex-equal stable marriage problem. *Japan Journal of Industrial and Applied Mathematics*, 10:1–19, 1993.

[59] T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 574–584, Berlin, Heidelberg, 2009. Springer-Verlag.

[60] T. Kavitha and M. Nasre. Optimal popular matchings. *Discrete Applied Mathematics*, 157(14):3181–3186, 2009.

[61] T. Kavitha and M. Nasre. Popular matchings with variable job capacities. In *ISAAC '09: Proceedings of the 20th International Symposium on Algorithms and Computation*, pages 423–433, Berlin, Heidelberg, 2009. Springer-Verlag.

[62] K. M. Keizer, M. de Klerk, B. J. J. M. Haase-Kromwijk, and W. Weimar. The Dutch algorithm for allocation in living donor kidney exchange. *Transplantation Proceedings*, 37:589–591, 2005.

[63] S. Kijima and T. Nemoto. Finding a level ideal of a poset. In *Proceedings of COCOON '09 : the 15th International Computing and Combinatorics Conference*, Lecture Notes in Computer Science, pages 317–327. Springer, 2009.

[64] Z. Király. Better and simpler approximation algorithms for the stable marriage problem. In *Proceedings of ESA '08: the 16th Annual European Symposium on Algorithms*, Lecture Notes in Computer Science, pages 623 – 634, 2008.

[65] B. Klaus and F. Klijn. Stable matchings and preferences of couples. *Journal of Economic Theory*, 121:75–106, 2005.

[66] B. Klaus and F. Klijn. Paths to stability for matching markets with couples. *Games and Economic Behavior*, 58:154–171, 2007.

[67] B. Klaus, F. Klijn, and J. Massó. Some things couples always wanted to know about stable matchings (but were afraid to ask). *Journal Review of Economic Design*, 11(3):175–184, 2006.

[68] B. Klaus, F. Klijn, and T. Nakamura. Corrigendum: Stable matchings and preferences of couples. *Journal of Economic Theory,* to appear, 2009.

[69] D.E. Knuth. *Mariages Stables*. Les Presses de L'Université de Montréal, 1976.

[70] L. Lovász and M.D. Plummer. *Matching Theory*. Number 29 in Annals of Discrete Mathematics. North-Holland, 1986.

[71] M. Mahdian. Random popular matchings. In *Proceedings of EC '06: the 7th ACM Conference on Electronic Commerce*, pages 238–242. ACM, 2006.

[72] D.F. Manlove. Stable marriage with ties and unacceptable partners. Technical Report TR-1999-29, University of Glasgow, Department of Computing Science, January 1999.

[73] D.F. Manlove. The structure of stable marriage with indifference. *Discrete Applied Mathematics*, 122(1-3):167–181, 2002.

[74] 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.

[75] D.F. Manlove and C.T.S. Sng. Popular matchings in the Capacitated House Allocation problem. In *Proceedings of ESA '06: the 14th Annual European Symposium on Algorithms*, volume 4168 of *Lecture Notes in Computer Science*, pages 492–503. Springer, 2006.

[76] D. Marx and I. Schlotter. Stable assignment with couples: Parameterized complexity and local search. In *Proceedings of IWPEC '09: the 4nd International Workshop on Parameterized and Exact Computation*, pages 200–311. Springer, 2009.

[77] R.M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of LATIN 2008: the 8th Latin-American Theoretical INformatics symposium*, volume 4957 of *Lecture Notes in Computer Science*, pages 593–604. Springer, 2008.

[78] D. McVitie and L.B. Wilson. The stable marriage problem. *Communications of the ACM*, 14:486–490, 1971.

[79] K. Mehlhorn and D. Michail. Network problems with non-polynomial weights and applications. Unpublished manuscript, 2006.

[80] J. Mestre. Weighted popular matchings. In *Proceedings of ICALP '06: the 33rd International Colloquium on Automata, Languages and Programming*, volume 4051 of *Lecture Notes in Computer Science*, pages 715–726. Springer, 2006.

[81] Dimitrios Michail. Reducing rank-maximal to maximum weight matching. *Theoretical Computer Science*, 389(1-2):125–132, 2007.

[82] R. Neidermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[83] C. Ng and D.S. Hirschberg. Three-dimensional stable matching problems. *SIAM Journal on Discrete Mathematics*, 4:245–252, 1991.

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

[85] A. E. Roth, T. Sönmez, and U. M. Ünver. A kidney exchange clearinghouse in New England. *American Economic Review, Papers and Proceedings*, 95(2):376–380, 2005.

[86] A. E. Roth, T. Sönmez, and U. M. Ünver. Pairwise kidney exchange. *J. Econom. Theory*, 125(2):151–188, 2005.

[87] A. E. Roth, T. Sönmez, and U. M. Ünver. Coincidence of wants in markets with compatibility based preferences. *American Economic Review*, 97(3):828–851, 2007.

[88] 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.

[89] A.E. Roth. On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica*, 54:425–427, 1986.

[90] A.E. Roth, T. Sönmez, and M. Utku Ünver. Kidney exchange. *Quarterly Journal of Economics*, 119:457–488, 2004.

[91] A.E. Roth and M.A.O. Sotomayor. *Two-sided matching: a study in game-theoretic modeling and analysis*, volume 18 of *Econometric Society Monographs*. Cambridge University Press, 1990.

[92] S. L. Saidman, A. E. Roth, T. Sönmez, U. M. Ünver, and S. L. Delmonico. Increasing the opportunity of live kidney donation by matching for two and three way exchanges. *Transplantation*, 81(5):773–782, 2006.

[93] S. Scott. *A study of stable marriage problems with ties*. PhD thesis, University of Glasgow, Department of Computing Science, 2005.

[94] S. L. Segev, S. E. Gentry, D. S. Warren, B. Reeb, and R. A. Montgomery. Kidney paired donation and optimizing the use of live donor organs. *J. Am. Med. Assoc.*, 293:1883–1890, 2005.

[95] B. Spieker. The set of super-stable marriages forms a distributive lattice. *Discrete Applied Mathematics*, 58:79–84, 1995.

[96] A. Subramanian. A new approach to stable matching problems. *SIAM Journal on Computing*, 23(4):671–700, 1994.

[97] J.J.M. Tan. A maximum stable matching for the roommates problem. *BIT*, 29:631–640, 1990.

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

[99] C. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Math. Oper. Res.*, 23(4):874–891, 1998.

[100] UK Transplant. *http://www.uktransplant.org.uk*.

[101] G. J. Woeginger. Exact algorithms for np-hard problems: a survey. pages 185–207, 2003.

[102] G.J. Woeginger. Open problems around exact algorithms. *Discrete Appl. Math.*, 156(3):397–405, 2008.

[103] H. Yanagisawa. *Approximation Algorithms for Stable Marriage Problems*. PhD thesis, Kyoto University, School of Informatics, 2007.

[104] H. Yanagisawa. Personal communication, 2008.

[105] `http://www.nrmp.org` (National Resident Matching Program website).

[106] `http://www.carms.ca` (Canadian Resident Matching Service website).

[107] `http://www.nes.scot.nhs.uk/sfas` (Scottish Foundation Allocation Scheme website).

[108] `http://www.nepke.org` (New England Program for Kidney Exchange website).

Index to first usage of major terminology and notation