Matching Under Preferences

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Deeksha Adil
20121071

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2017

Supervisor: Prof. Saket Saurabh

Certificate

This is to certify that this dissertation entitled Matching Under Preferences towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Deeksha Adil at The Institute of Mathematical Sciences, Chennai and The Institute of Informatics, University of Bergen under the supervision of Prof. Saket Saurabh, Professor, Institute of Mathematical Sciences, Chennai , during the academic year 2016-2017.

Prof. Saket Saurabh

Committee:
Prof. Saket Saurabh
Dr. Soumen Maity

Dedicated to Mount Fløyen

Declaration

I hereby declare that the matter embodied in the report entitled Matching Under Preferences are the results of the work carried out by me at the Institute of Mathematical Sciences, Chennai and The Department of Informatics,University of Bergen under the supervision of Prof. Saket Saurabh and the same has not been submitted elsewhere for any other degree.

20 March 2017 Deeksha Adil

Acknowledgements

I would like to express my sincere gratitude towards Prof. Saket Saurabh for all his constant help and guidance. I am really grateful for all the opportunities and exposure that I was provided with as a part of Prof. Saket's group. I highly appreciate the amount of time and resources devoted towards me during the project. I would have to mention how thankful I am for having been given the opportunity to visit the University of Bergen. The visit has had a huge impact on me academically and I am glad to have gotten such an exposure.

I would next like to thank Dr. Sushmita Gupta for having spent so much time and patiently discussing every bit starting from the basics with me. I was completely new to the area but her constant energy, enthusiasm has been a huge encouragement towards learning and understanding the field.

I thank Dr. Meirav Zehavi and Sanjukta Roy for their time and I would like to admit that I did enjoy the discussions.

I thank Dr. Soumen Maity, for his constant interest and help.

I would like to specially thank Prof. Raghuram for his constant support in the years I spent in IISER and even now. I am really grateful towards him for suggesting IMSc as an option for my Master's Thesis.

I thank the Department of Mathematics, IISER Pune and IISER Pune for providing me with such an opportunity to carry out such a project.

Lastly I would like to thank my parents and sister, all my friends from IMSc,

Lawqueen, Roohani, Pratik, Sanjukta, Oorna and Aditi, my friends from IISER Pune Prachi, Shipra, Anirban and my math classmates from IISER for all the emotional and mental support throughout. It would not have been possible to complete this project without this support.

Abstract

The *Stable Marriage Problem* is a classic problem of matching under preferences. It involves two sets of agents, classically refered to as *men* and *women* each having a *preference list* over the other set. A matching is a disjoint collection of man woman pairs. We define a blocking edge with respect to a matching to be a (man,woman) pair, say (m, w), such that both m and w prefer each other over their matched partners in the matching. A *Stable Matching* is a matching with no blocking pairs. In the 1960s Gale and Shapley gave an algorithm, now known as the Gale-Shapley algorithm, that solves the stable marriage problem in polynomial time.

Various extentions of the Stable Marriage problem have been studied. The simplest and most natural extension is to *incomplete lists*, called a the Stable Marriage problem with incomplete lists, or SMI. As the name suggests an instance of this problem would involve the preference lists to be incomplete. The GS algorithm can be easily extended to this problem and again it has been shown that a stable matching always exists and can be found in polynomial time. If *indifference* is introduced in the problem instance, that is the preference lists may involve ties and are incomplete as well, the problem is referred to as *The Stable Marriage problem with Ties and Incomplete Lists* or SMTI in short. With ties there are three different notions of stablity - super stability, strong stability and weak stability. We are only interested in *weakly stable matchings*. A matching in an SMTI instance is weakly stable if there is no pair (m, w) such that m and w both strictly prefer each other over their respective matched partners.

It has been shown that arbitrarily breaking the ties in an SMTI problem instance $\mathcal{I}$ we get a corresponding SMI instance $\mathcal{I}'$ such that a stable matching in $\mathcal{I}'$ is a weakly stable matching in $\mathcal{I}$. Depending on how we break the ties the size of the stable matching obtained may vary. The problem of finding a maximum sized or a minimum sized weakly stable matching is shown to be NP-hard. When the underlying graph is not a bipartite graph, the problem is commonly known as the Stable Roommates problem. We can define versions of SMI and SMTI for the Stable Roommates setting as well and we call them SRI and SRTI respectively. Ronn showed that in an SRTI instance the problem of deciding whether a weakly stable matching exists or not is NP-complete.

In this dissertation we study the problem of finding a maximum (minimum) sized weakly stable matching in SMTI and SRTI instances in the realm of parameterized

complexity. We show that these problems are *fixed parameter tractable* by giving polynomial kernels. Also when the underlying graph is planar we show that we can improve the kernel further to a linear kernel thus giving us better algorithms.

Another extension of the Stable Marriage problem gives us the well known Hospital/Residents Problem (HR Problem). This problem is a many-one stable matching problem. An instance of Hospital/Residents Problem is again a bipartite graph, $H \cup R$, where H is the set of hospitals and R the set of residents. Each hospital has a strict preference over a subset of residents and each resident has a strict preference over a subset of hospitals. With every hospital we have an associated number called its *upper quota*. Our goal is to assign residents to hospitals such that for every hospital not more than its upper quota residents are assigned to it, and the assignment is *'stable'*. Again a stable assignment is an assignment with no blocking pairs, but the definition of a blocking pair is slightly different than the one defined for a stable marraige instance. The GS algorithm can be used in this setting as well to find a stable matching and again a stable matching always exists.

When we associate a lower quota with every hospital and add an additional constraint to the problem of assigning at least lower capacity number of residents to each hospital. Now we want to find such a feasible matching which is as stable as possible. A stable matching found by GS may not be feasible and by the Rural Hospital Theorem which states that all stable matchings match the same set of vertices, we are assured that no stable matching would be feasible. So we look for a feasible matching as *'stable'* as possible. In other words we look for a matching with either fewest number of blocking edges or fewest number of blocking residents. Both these problems of finding a feasible matching minimizing the number of blocking edges or blocking residents are shown to be NP-hard. We study the Hospital/Residents problem with lower quotas in the realm of parameterized complexity. We give an *XP* Algorithm with parameter *blocking residents*. For other parameters *sum of lower quotas* and *colleges with positive lower quota* we show that the problem is W[1] hard.

Contents

Abstract **xi**

1 Stable Matching **1**
1.1 Formal Definitions and Properties 1
1.2 Stable Marriage Problem with Incomplete Lists and/or Ties 5
1.3 Hospital/Residents Problem 7
1.4 Stable Roommates Problem 8

2 Parameterized Complexity **13**
2.1 Defining a Parameterized Problem 13
2.2 W-Hierarchy 14
2.3 Kernelization 16

3 Parameterized Algorithms for the SM Problem **17**
3.1 Introduction 17
3.2 Formulation and Problem Definition 18
3.3 Our Contribution 19
3.4 Preliminaries 20
3.5 Algorithms for SMTI 21
3.6 Algorithms for SRTI 27
3.7 SMTI and SRTI on planar graphs 29

4 Parameterized Algorithms for the HR Problem **35**
4.1 Introduction 35
4.2 An XP-Algorithm 37
4.3 W[1]-Hardness Results 41

5 Results and Conclusions **45**

Chapter 1

The Stable Marriage Problem

Gale and Shapley's seminal paper in 1962 contained the first algorithm which solved the classical *Stable Marriage Problem*. This problem along with its variants have been extensively studied in mathematics, economics and computer science. The classical stable marriage problem involves two sets of agents, referred to as *men* and *women*. Each man ranks all the women in a certain strict order of preference. Similarly each woman ranks every man in a strict order of preference. The task is to find a set of *man-woman* pairs, which we can call a set of marriages such that they are *stable* i.e., there are no *extra-marrital affairs*. In other words, no man and woman together like each other over their current assigned partners. Such a set of pairs is called a *stable matching*.

In this chapter, we begin with formally defining the *Stable Marriage Problem*, Gale-Shapley Algorithm and state some known results. In section 1.2 we give a generalisation of the problem when the preference lists are not complete and/or may include ties. In section 1.3 we will define the *many to one* matching extension of the Stable Marriage problem called the Hospital/Residents Problem. Section 1.4 gives certain known results for non-bipartite, general graphs.

1.1 Formal Definitions and Properties

The *Stable Marriage Problem* is a classic problem of matching under preferences. It involves two sets of agents, classically refered to as *men* and *women* each having a *preference list* over the other set. Formally, the preference list of a man $m \in M$ is

a total ordering of the set W and similarly the preference list of a woman is a total ordering of the set M. This can be seen as a bipartite graph $G = M \cup W$ with a set of preference lists $\mathcal{L}^M, \mathcal{L}^W$ of men and women, respectively. A matching is a disjoint collection of man-woman pairs. With respect to a matching μ we define a blocking edge.

Definition 1. *(Blocking edge) An edge $e = (a, b) \in G$ blocks a matching μ in graph G if vertex a prefers b over it's matched partner $\mu(a)$ and vertex b prefers a over it's matched partner $\mu(b)$.*

We can now formally define a stable matching.

Definition 2. *(Stable Matching) Matching μ in G is a stable matching iff it has no blocking pairs.*

The Stable Marriage Problem can now be defined.

The Stable Marriage Problem **Input:** A bipartite graph G with preference lists $\mathcal{L}^M$ and $\mathcal{L}^W$. **Task:** Find (if exists) a stable matching in $(G, \mathcal{L}^M, \mathcal{L}^W)$.

The Gale-Shapley Algorithm (GS algorithm) gives a stable matching in a stable marriage instance in $\mathcal{O}(n^2)$ time where n is the size of the instance. In the *Man-Optimal* GS algorithm, every man proposes one by one to women and women accept or decline the proposal. Initially all men and women are assigned a state free. We start with a man $m \in M$ who proposes to his most preferred woman w. The woman w accepts the proposal and we call (m, w) to be engaged. Now we repeat this process for the next man m' in M. Man m' proposes to its most preferred neighbor it has not yet proposed to, say w'. If w' is engaged to a man it prefers more than m' it declines the offer and m' proposes to its next most preferred neighbor. If w' is free or engaged to a man it prefers less than m' it accepts the offer and breaks its engagement with its previous partner setting that man to be free. The process of proposals continues as long as there is a man who is free. The algorithm terminates when every man is engaged or every man has proposed to all of his neighbors.

Algorithm 1 Man optimal Gale-Shapley Algorithm

procedure GS($M \cup W, \mathcal{L}^M, \mathcal{L}^W$) Assign every $m \in M$ and $w \in W$ to be free
  while $\exists m \in M$ that is free **do**
    $w = m$'s most preferred neighbour it has not yet proposed to
    if w is free **then**
      set m to be engaged to w
    else
      if w is engaged to m' such that w prefers m more than m' **then**
        set w to be engaged to m and set m' to be free
      else
        m remains free
      end if
    end if
  end while
  return the engaged pairs
end procedure

We state the following theorem without proof which yields the correctness of the GS Algorithm. The proof can be found in [2]

Theorem 1. *[2] The Gale-Shapley algorithm always terminates and the matching returned by it is stable.*

Corollary 1. *[2] An instance of the stable marriage problem always admits a stable matching.*

The above algorithm can also be used when the roles of men and women are exchanged. When women propose and men dispose, we again get a stable matching, referred to as the *women optimal matching*. Looking at this algorithm there might seem to be certain non determinism about the order in which the men (or women) propose. The following theorem proven in [2] resolves this.

Theorem 2. *[2] All possible executions of the Gale-Shapley algorithm (with the men as proposers) yield the same stable matching, and in this stable matching, each man has the best partner that he can have in any stable matching.*

The above theorem also says that in the men proposing Gale-Shapley M vertices get the best partners they can in any stable matching. The following are some results

which we state without proof about about stable matchings. The proofs can be found in [2].

Theorem 3. *[2]*

1. *In the M optimal stable matching each vertex in W gets the worst partner it can get in any stable matching.*
2. *All stable matchings of a given instance, have the same size and match the same set of agents.*

Observe that in the Gale-Shapley algorithm, in any iteration if a man m proposes to a woman w, then w cannot finally be matched to a partner it prefers less than m in the stable matching returned. Also m cannot get a partner it prefers more than w. This is because, when m proposes to w, m has been rejected by every agent ranked higher than w in its list in some previous iteration of the algorithm. Similarly when w receives a proposal from m it accepts it only if it was engaged to someone it ranks lower than m. Otherwise it is already engaged to someone better. So once m proposes to some w every agent in its list before w can be deleted and this will have no effect on the output of the algorithm. Similarly every agent behind m in w's list can be deleted. This is basically deleting an edge from the original graph that we know is not a part of the stable matching. This gives the following extended version of the M optimal or man optimal Gale-Shapley Algorithm.

Algorithm 2 Extended M optimal Gale-Shapley

```
1: procedure GS(M ∪ W, L^M, L^W) Assign every m ∈ M and w ∈ W to be free
2:     while ∃m ∈ M that is free do
3:         w = m's most prefered neighbour it has not yet proposed to
4:         if w is engaged to m' then
5:             set m to be engaged to w and set m' to be free
6:         end if
7:         ∀m' such that w prefers m more than m',delete the pair (m', w)
8:     end while
9:     return the engaged pairs
10: end procedure
```

Notice that unlike the GS algorithm here the woman w always accepts the proposal. This is because if the man was not ranked higher than her current partner the

edge would have been deleted when her current partner had proposed. The cut short lists at the end of the man proposing extended GS is called **MGS-lists**. Similarly we define **WGS-lists**. The intersection of these two lists is defined as **GS-list**. This idea of a GS list will be used to find stable matchings in general graphs and also bipartite graphs which are not complete. The following theorem from [2] gives some properties of GS-lists.

Theorem 4. *[2] For a given instance of the stable marriage problem,*

1. *all stable matchings are contained in the GS-lists.*
2. *no matching contained in the GS-lists can be blocked by a pair that is not in the GS-lists.*
3. *in the man-optimal(respectively woman-optimal) stable matching, each man is partenered by the first(respectively last) woman on his GS-list, and each woman by the last (respectively first) man on hers.*

1.2 Stable Marriage Problem with Incomplete Lists and/or Ties

The original Stable Marriage problem involves the preference lists to be complete (men rank all women and women rank all men) as well as strictly ordered. Practical applications do not always have complete preference lists. There can be some partners that can be unacceptable. Also it is possible that some agent is indifferent towards a set of agents. This motivates studying the stable marriage problem when the lists are incomplete and/or have ties.

1.2.1 Stable Marriage Problem with Incomplete Lists

The preference lists now are not complete i.e., the underlying graph is not a complete bipartite graph anymore. Such an instance is called an instance of *Stable Marriage Problem with Incomplete Lists (SMI).* The notion of a blocking pair and stability still remains the same for this problem as well. It can be shown that GS algorithm and the extended GS Algorithm both can be used to give a stable matching in an instance of SMI.

1.2.2 Stable Marriage Problem with Ties

When we introduce ties in the preference lists, we have three notions of stability - super stable, strongly stable and weakly stable.

Definition 3. *(Super Stable Matching [2]) A matching μ is super stable if there does not exist any pair $(m, w) \in E(G) \setminus \mu$ such that m and w prefer each other atleast as much as their partners in μ.*

Definition 4. *(Stronly Stable Matching[2]) A matching μ is strongly stable if there does not exist any pair $(m, w) \in E(G) \setminus \mu$ such that one of m and w strictly prefers the other over his/her partner in μ and the other is atleast indifferent.*

Definition 5. *(Weakly Stable Matching [2]) A matching μ is weakly stable if there does not exist any pair $(m, w) \in E(G) \setminus \mu$ such that both m and w strictly prefer each other over their matched partners in μ.*

Strongly Stable and Super Stable matchings need not always exist in an instance of SMT, but there are polynomial time algorithms that decide whether such matchings exist or not [6],[7]. On the other hand, weakly stable matchings always exist in an instance of SMT. Arbitrarily breaking the ties a Stable Marriage instance is created, a stable matching in which is a weakly stable matching in the original instance. Depending on how the ties are broken we may get different matchings. In this dissertation we are interested in weakly stable matchings.

1.2.3 Stable Marriage Problem with Incomplete Lists and Ties

A Stable Marriage Problem with Incomplete Lists and Ties or SMTI instance, allows unacceptable partners and also ties in the lists of agents. Since there are ties, we again have the above defined notions of stability. We are mostly interested in weakly stable matchings. As we have seen, a weakly stable matching always exists in an SMT instance. Similarly, if we arbitrarily break ties in an instance of SMTI, we get an instance of SMI. A stable matching in the so found SMI instance is a weakly stable matching in our original SMTI instance. The difference from SMT, is that depending on how we break the ties, we get stable matchings of different sizes. The problem of finding a maximum sized weakly stable matching in an instance of SMTI is shown to

be NP-hard [5]. In our work we have studied the parameterized complexity of this problem which is presented in chapter 3.

1.3 Hospital/Residents Problem

Consider the following practical problem of assigning Residents to Hospitals. There are certain number of hospitals available in a particular town and all the residents of that town have to be assigned to one hospital that will take care of them. Now practical constraints would bound the maximum number of residents each hospital can accommodate (call this the upper quota of the hospital). This motivated a many-to-one extension of the classical Stable Marriage problem which in literature is known as the *Hospital/Residents Problem*(HR Problem). Formally in an HR problem instance, we have a set H of hospitals, R of residents. With each hospital $h_i \in H$ we have an associated upper quota u_i. Each hospital and resident has a strict preference list over the other (may or may not be complete). Our goal is to assign residents to hospitals such that for each hospital h_i not more than u_i residents are assigned and each resident is assigned to at most one hospital. A *blocking pair* for an assignment μ is a pair $(h, r), h \in H, r \in R$ such that h prefers r more than one of its assigned residents or h has not filled its upper quota number of residents, and r is unassigned or r prefers h over its current hospital. The GS algorithm can be used to get a stable assignment in this setting (details in chapter 4) and a stable assignment always exists [8].

Sometimes, it might happen that a certain hospital needs a minimum number of assigned residents in order to function. Such a scenaio can be modelled by assigning a number $l_i, l_i \leq u_i$ with each hospital h_i, called its lower quota. We now have an additional constraint to satisfy i.e., every hospital must get at least its lower quota number of residents. An assignment satisfying this lower quota bound is called a *feasible* assignment. The stable assignment we get from GS may not be a feasible assignment and by the Rural Hospital Theorem we know that the same set of hospitals and residents will be assigned in every stable assignment. So no stable assignment is feasible. In such cases we aim to find an assignment which is 'as stable as possible'. In other words we try to minimize the number of blocking edges or blocking residents (the residents involved in a blocking edge). Both these problems of finding a feasible

matching minimizing the number of blocking edges or blocking residents are shown to be NP-hard [8]. We have studied the HR problem with lower quotas in the realm of parameterized complexity. Details are presented in chapter 4.

1.4 Stable Roommates Problem

In the previous sections we have looked at variants of the classical Stable Marriage problem on bipartite graphs. When the underlying graph is not bipartite, the original stable marriage problem is called as the *Stable Roommates* Problem (SR Problem). An instance $\mathcal{I} = (G, \mathcal{L}^V)$ of SR consists of a graph $G = (V, E)$, V being the vertex set and E the edge set and a set of preference lists $\mathcal{L}^V$. A preference list of a vertex $v \in V$ is an ordering of the vertices in $V \setminus \{v\}$. A matching is a set of disjoint vertex pairs. The aim is to find a stable matching in this setting. A matching is stable if there are no blocking pairs i.e., no two vertices prefer each other over their matched partners. The name *'Stable Roommates'* is associated to this problem as this can be related to the problem of assigning rooms to students. The rooms should be assigned in such a way that there are no conflicts between rooms.

Unlike the SM problem instance, an instance of SR might not always have a stable matching. The following example from [9] can be checked to have no stable matching. The instance consists of 6 people with preference lists as shown in table 1.1.

1	2	6	4	3	5
2	3	5	1	6	4
3	1	6	2	5	4
4	5	2	3	6	1
5	6	1	3	4	2
6	4	2	5	1	3

Table 1.1: Example of an SR instance with no stable matching

Irving has given an algorithm that in $\mathcal{O}(n^2)$ time decides whether or not an SR instance has a stable matching and finds one if exists. The algorithm has two phases, phase one reduces the instance and phase two eliminates certain *cycles* from the

graph at the end of which we are left with a matching (if exists). We now give Irving's algorithm without proof.

Algorithm 3 Phase 1: Irving's Algorithm for SR

```
1: procedure SR(G = (V, E), L^V) Assign every v ∈ V to be free
2:     while ∃v ∈ V that is free do
3:         u = v's most preferred neighbour that has not rejected v
4:         if u is not holding anyone then
5:             u holds v
6:             u rejects all v' ∈ Nbr(u) such that u prefers v over v'.
7:             Delete these edges (u, v') from G.
8:         else
9:             u holds v and rejects all v' ∈ Nbr(u) such that u prefers v over v'.
10:            Delete these edges (u, v') from G.
11:        end if
12:    end while
13:    return G, L^V
14: end procedure
```

In Algorithm 3, a vertex being free means it has currently not proposed to anyone who has held its proposal. While we have some free vertex, the algorithm goes on. A free vertex proposes to its most preferred neighbour that has not rejected it. The vertex receiving the proposal can either hold the proposal (if it is free or currently holds someone it prefers less) or reject the proposal (if it is holding someone it prefers more). Once a vertex receives some proposal, it rejects all neighbours it prefers less than the proposing vertex. We delete these edges from the graph. In the end we are left with a reduced graph and preference lists. In this new graph if there is a degree 0 vertex we can immediately say that the instance has no stable matching (proven in [9]). Also if in the reduced graph G we have every vertex having degree one then that is a stable matching.

Lemma 1. *[9] If the first phase of the algorithm terminates with one person having been rejected by all of the others, then no stable matching exists.*

Lemma 2. *[9] If in the reduced preference lists, every list contains just one person, then the lists specify a stable matching.*

For phase 2 of the algorithm, we require the following definition of a *preference cycle.*

Definition 6. *(preference cycle)[9] A sequence of vertices $a_1, a_2, a_3...a_r$ such that,*

(i) for $i = 1, ..., i-1$ the second person in a_i's reduced preference list is the first person in a_{i+1}'s list; we will denote this person by b_{i+1}.

(ii) the second person in a_r's reduced presference list is the first in a_1's; we will denote this person by b_1.

Definition 7. *(cycle elimination)[9] A cycle elimination is applied to a given set of reduced preference lists and a preference cycle. This reduction forces every b_i to reject the proposal it currently holds from a_i, forcing a_i to propose to b_{i+1}. As a result all successors of a_i in b_{i+i}s reduced list can be deleted, and b_{i+1} can be deleted from their lists.*

We can now give the phase 2 algorithm.

Algorithm 4 Phase 2: Irving's Algorithm for SR

```
1: procedure GS(Reduced G = (V, E), L^V)
2:     while ∃ a preference cycle ρ do
3:         apply cycle elimination to ρ.
4:         if A preference list becomes empty then
5:             return No Stable Matching
6:         else
7:             if Each preference list is of size 1 then
8:                 return The pairs corresponding to this graph
9:             end if
10:        end if
11:    end while
12: end procedure
```

Algorithm 4 keeps eliminating preference cycles from the instance till it gets a clear no instance or a solution. The further analysis in Chapter 4 of [9] shows the correctness of the algorithm and running time bound. Lets see how this algorithm shows that the example in Table 1.1 has no stable matching.

Phase 1 of the algorithm, proceeds as follows where the rightarrow between person i and j shows that person i has proposed to j:

1	$\rightarrow$	2,	2 holds 1, delete edges (2,6),(2,4)
2	$\rightarrow$	3,	3 holds 2, delete edges (3,5),(3,4)
3	$\rightarrow$	1,	1 holds 3, delete edges (1,5)
4	$\rightarrow$	5,	5 holds 4, delete edges (5,2)
5	$\rightarrow$	6,	6 holds 5, delete edges (6,1),(6,3)
6	$\rightarrow$	4,	4 holds 6, delete edges (4,1)

Table 1.2: Execution of Phase 1 of the algorithm

The reduced table after phase 1 looks as follows:

1	2	3
2	3	1
3	1	2
4	5	6
5	6	4
6	4	5

Table 1.3: Reduced Table after phase 1

The above table has the following preference cycle : $a_1 = 1, a_2 = 2, a_3 = 3$ and $b_1 = 2, b_2 = 3, b_3 = 1$. Eliminating this cycle, 2 rejects 1, 3 rejects 2 and 1 rejects 3 forcing the lists of $1, 2, 3$ to be empty. Thus this instance has no stable matching.

In this section we saw the SR problem and Irving's algorithm to find a stable matching if exists. Similar to Stable Marriage instance, we can extend this problem to have ties in the list as well as incomplete lists. We similarly define problems SRI, SRT and SRTI. Ronn in [10] shows that in an SRTI instance merely deciding whether a weakly stable matching exists is also NP-complete. Recall that in a SMTI instance we had the guarantee that a weakly stable matching exists. This is because, when we arbitrarily break the ties creating an SMI instance, we are guaranteed to find a stable matching in the SMI instance. For the roommates setting, every SRI instance does

not have a stable matching, thus, depending on how we break the ties we might end up in an instance of SRI which has a stable matching or which does not have one. So, as in the case of finding a maximum sized stable matching in an SMTI instance, the main issue was with how to break the ties, in an SRTI instance even for deciding the existence of such a matching we face the same problem of how to break the ties. This results in the hardness.

The following chapters will contain a detailed analysis of the problems introduced in this chapter. In chapter 2 we first define some tools from parameterized complexity that we use further. In chapter 3 we give an analysis of the parameterized algorithms for the Stable Marriage Problem with ties and incomplete lists. Chapter 4 contains some results for the Hospital/Residents problem with lower quotas.

Chapter 2

Parameterized Complexity

In this chapter we look at some basics of parameterized complexity. We begin by defining a parameterized problem.

2.1 Defining a Parameterized Problem

In classical complexity theory we classify problems based on their run time that depends only on the input size. In practical problems we often have more information about the input than the size, such as some structure of the input or solution size required. In parameterized complexity we generally exploit these structures of the input and try to capture the *'hardness'* of the problem with this value. By this we mean, we try and capture the non-polynomial part of the run time with this value so that the run time dependence on the instance size is still polynomial. In a parameterized problem, the input is always the instance along with a natural number (which we call our parameter). We aim to find algorithms that run in time polynomial in the input size times some function of the parameter, which are called *FPT Algorithms*. For detailed motivation on the study of parameterized complexity refer to [4]. We now formally define a parameterized problem.

Definition 8. *[3] (Parameterized problem) A parameterized problem is a language* $L \subseteq \Sigma^* \times \mathbb{N}$*, where* Σ *is a fixed, finite alphabet. For an instance* $(x, k) \in \Sigma^* \times \mathbb{N}$*,* k *is called the parameter.*

As in classical complexity we always try to look for algorithms that are polynomial time solvable in the input size, in parameterized complexity we try and look for

algorithms that are *fixed-parameter tractable.*

Definition 9. *[3](fixed-parameter tractable) A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called fixed-parameter tractable (FPT) if there exists an algorithm $\mathcal{A}$ (called a fixed-parameter algorithm), a computable function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k)|(x, k)|^c$. The complexity class containing all fixed-parameter tractable problems is called FPT.*

We now define another complexity class of problems XP.

Definition 10. *[3](XP) A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called slice-wise polynomial (XP) if there exists an algorithm $\mathcal{A}$ and two computable functions $f, g : \mathbb{N} \to \mathbb{N}$ such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k)|(x, k)|^{g(k)}$. The complexity class containing all slice-wise polynomial problems is called XP.*

2.2 W-Hierarchy

Similar to the NP-completeness theory of polynomial time computation there is a lower bound theory for parameterized problems. Such lower bounds are very important in ruling out certain kinds of algorithms. Unlike classical complexity in parameterized complexity different levels of hardness are observed. Classes W[1],W[2],.. are used to capture these problems. The details of the different levels observed can be found in Chapter 13 of [3]. The main assumption here is $FPT \neq W[1]$ which is a stronger assumption than $P \neq NP$. The W-Hierarchy can be thought of as shown in figure 2.1

To classify problems into such classes we need a notion of a *reduction*, that reduces a parameterized problem A to a parameterized problem B such that if B has an algorithm of a certain kind then so does A. For our purposes, we mainly try to rule out the existence of FPT algorithms. It has been shown that CLIQUE parameterized by solution size is W[1] complete. This means that the class of problems W[1] is the set of all problems that can be obtained through a *parameterized reduction* from CLIQUE parameterized by solution size. We now define the notion of a *parameterized reduction.* If we can find a parameterized reduction from CLIQUE or some other W[1]

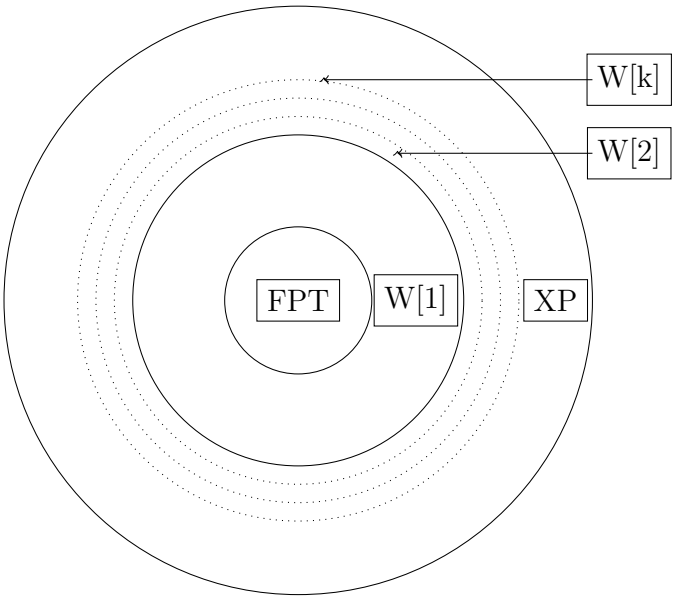

Figure 2.1: Representation of the W-Hierarchy

hard problem to a problem X then we can say that X cannot have an FPT algorithm unless FPT=W[1].

Definition 11. *[3](Parameterized reduction)Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A parameterized reduction from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that*

1. *(x, k) is a Yes-instance of A if and only if (x', k') is a Yes-instance of B,*
2. *$k \leq g(k)$ for some computable function g, and*
3. *the running time is $f(k)|x|^{\mathcal{O}(1)}$ for some computable function f.*

The following results hold for a parameterized reduction. We state them without proof which can be found in [3].

Theorem 5. *[3] If there is a parameterized reduction from A to B and B is FPT, then A is FPT as well.*

Theorem 6. *[3] If there are parameterized reductions from A to B and from B to C, then there is a parameterized reduction from A to C.*

2.3 Kernelization

In this section we define one of the most used tools for finding FPT algorithms. Let $A \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. If suppose we want to design an FPT algorithm for A, one natural way of thinking could be, given an instance (x, k) of A, find an equivalent instance (that preserves decision) that has a size bounded by a function of the parameter. Doing so we can then use some brute force algorithm in the new equivalent instance and decide the answer for (x, k). This is the main idea of kernelization. An intereting relation between a kernelization algorithm and FPT algorithm is that a paramterized problem Q is FPT if and only if it has a kernelization algorithm [3].

Definition 12. *[3](Reduction Rule) A data reduction rule, or simply, reduction rule, for a parameterized problem Q is a function $\phi : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ that maps an instance $(\mathcal{I}, k)$ of Q to an equivalent instance $(\mathcal{I}', k')$ of Q such that ϕ is computable in time polynomial in $|\mathcal{I}|$ and k.*

In the above definition, equivalence means that $(\mathcal{I}, k)$ is a Yes-instance of Q if and only if $(\mathcal{I}', k')$ is a Yes-instance of Q. This property is referred to as the *safeness* of the reduction rule. The idea of a *kernelization algorithm* is to design a series of such reduction rules that shrink the size of the instance with each application.

Definition 13. *[3] (Kernelization, kernel) A kernelization algorithm, or simply a kernel, for a parameterized problem Q is an algorithm $\mathcal{A}$ that, given an instance (I, k) of Q, works in polynomial time and returns an equivalent instance (I', k') of Q. Moreover, we require that $size_{\mathcal{A}}(k) \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.*

We abuse the term kernel to refer to the algorithm as well as the equivalent instance returned by the algorithm. An example of a kernelization algorithm is presented in chapter 3, we show a kernelization algorithm to prove that the Stable Marriage problem with incomplete lists and ties is FPT when parameterized by the solution size. In chapter 4 we give an XP algorithm for the Hospital/Residents problem parameterized by the number of blocking residents, followed by a W[1] hardness result that shows that we do not expect to find an FPT algorithm for this problem as long as our assumption $FPT \neq W[1]$ holds.

Chapter 3

Parameterized Algorithms for the Stable Matching Problem with Ties and Incomplete Lists

3.1 Introduction

This chapter is based on the work done in [11]. The problem *Stable Marriage with Ties and Incomplete Lists (SMTI)* has been introduced in section 1.2. The problem of finding a maximum or a minimum sized weakly stable matching in an instance of SMTI is known to be NP-Hard [5]. In this chapter we study the parameterized complexity of the problems when the underlying graph is bipartite (SMTI instance), non-bipartite (SRTI) and planar. The parameters considered are the solution size for SMTI instance and the size of the maximum matching of the graph for an SRTI instance.

Let us motivate our study in the context of the classical work Gale and Shapleys work [1], namely the National Resident Medical Program, where medical residents are matched to available positions in medical colleges. In particular, we would thus argue that even when the objective is to maximize the size of the stable matching, solution size is a realistic parameter. First, it is generally not practical that residents (hospitals) should have to submit a ranking of all qualifying hospitals (resp. residents). Instead, a truncated list containing their top choices is desirable. Second, it is also likely that the agents (residents or hospitals) would be tied on some of their

admitted choices, signifying that the choices are equally good. Assuming that the ordering on tied groups is transitive, we have the practical realization of the theoretical model behind SMTI. Additionally, any given hospital is likely to have more than one position to offer, and any resident can only accept at most one position, but the total number of available positions is likely to be significantly smaller than the number of residents who are applying. Therefore, the maximum size of a matching can be significantly smaller than the size of the instance. To utilize the available resources, it is imperative for the matching agency to assign as many available positions to as many qualifying residents. These considerations lead us to believe that the size of the largest stable matching is a realistic parameter for further analysis.

3.2 Formulation and Problem Definition

We formulate SMTI via bipartite graphs: given a bipartite graph $G = (A \cup B, E)$, the men (women) are represented by the vertices in A (resp. B). Moreover, we have a family of preference lists, $\mathcal{L}^A$: for each vertex $a \in A$, $\mathcal{L}^A(a) \in \mathcal{L}^A$ is a (not necessarily strict) ranking over a subset of B. Symmetrically, we have a family $\mathcal{L}^B$. Note that we assume without loss of generality that $b \in \mathcal{L}^A(a)$ if and only if $a \in \mathcal{L}^B(b)$. We now define the parameterized problems Max-SMTI and Min-SMTI.

Max-SMTI
Input: Graph $G = A \cup B, E$, preference lists $\mathcal{L}^A, \mathcal{L}^B$, integer k.
Parameter: k
Question: Does there exist a weakly stable matching of size at least k?

Min-SMTI
Input: Graph $G = A \cup B, E$, preference lists $\mathcal{L}^A, \mathcal{L}^B$, integer k.
Parameter: k
Question: Does there exist a weakly stable matching of size at most k?

When we extend this problem to the Stable Roommates setting, we have an arbitrary input graph $G = (V, E)$ where each vertex $v \in V$ has a preference ranking over its neighbors $N(v)$. This ranking need not be strict and can involve ties. Clearly, the notion of stability is well defined also in this setting. Given an instance of SRTI,

merely testing whether there exists a stable matching is NP-hard [10]. Thus, defining the problem Max-SRTI similar to Max-SMTI, we can conclude that there does not exist (unless P=NP) any algorithm for Max-SRTI which runs in time of the form $f(k)|V|^{\mathcal{O}(1)}$ (or even $|V|^{f(k)}$) where f depends only on k. Indeed, by setting $k = 1$, we could employ such an algorithm to test the existence of a stable matching in polynomial time. Hence, we introduce an alternative parameterization. Note that a stable matching, if one exists, is a maximal matching in the graph. Furthermore, the size of any maximal matching is at least half the size of the maximum matching of the graph. Let ℓ denote the size of the maximum matching of the graph. Thus, if a stable matching exists, then its size and the value of ℓ differ by a factor of at most 2. Clearly, this leads us to directly to the parameterization of Max-SRTI by ℓ.

Max-SRTI
Input: Graph $G = V, E$, preference lists $\mathcal{L}^V$, integer k.
Parameter: ℓ
Question: Does there exist a weakly stable matching of size at least k?

3.3 Our Contribution

We show that both Max-SMTI and Min-SMTI are FPT when parameterized by k, the solution size, and that Max-SRTI is FPT when parameterized by the size of a maximum matching. We employ the method of polynomial-time preprecessing, kernelization. By devising a nontrivial marking scheme, we show that all the above problems admit a polynomial kernel. For example, in the context of Max-SMTI (Min-SMTI), given an instance $(\mathcal{I}, k)$ of Max-SMTI (Min-SMTI), we output (in polynomial time) another instance $(\mathcal{I}', k)$ of Max-SMTI (Min-SMTI) where $|\mathcal{I}'| = \mathcal{O}(k^2)$ and $(\mathcal{I}, k)$ is a Yes-instance of Max-SMTI (Min-SMTI) if and only of $(\mathcal{I}', k)$ is Yes-instance of Max-SMTI (Min-SMTI). Our kernels also result in the design of FPT algorithms: First obtain an equivalent instance by applying the kernelization algorithm, where the output graph $G' = (A' \cup B', E')$ has $\mathcal{O}(k^2)$ edges. If we solve Max-SMTI, then we enumerate all subsets of edges of size q in G', where $k \leq q \leq 2k$. Since $\sum_{q=k}^{2k} \binom{|E'|}{q} \leq |E'|^{\mathcal{O}(k)}$; so $|E'| = \mathcal{O}(k^2)$ implies the running time is $2^{\mathcal{O}(k \log k)}$. If we solve Min-SMTI, then we enumerate all subsets of edges of size at most k in G'; again, the running time

is $\sum_{q=1}^{k} \binom{|E'|}{k} = 2^{\mathcal{O}(k \log k)}$. In both cases, for each subset of edges, we test whether it is a stable matching in polynomial time. Overall, we show that

Theorem 7. *Max-SMTI admits a kernel of size $\mathcal{O}(k^2)$, and an algorithm with running time $2^{\mathcal{O}(k \log k)} + n^{\mathcal{O}(1)}$.*

Theorem 8. *Min-SMTI admits a kernel of size $\mathcal{O}(k^2)$, and an algorithm with running time $2^{\mathcal{O}(k \log k)} + n^{\mathcal{O}(1)}$.*

For Max-SRTI, we derive the following theorem.

Theorem 9. *Max-SRTI admits a kernel of size $\mathcal{O}(\ell^2)$, and an algorithm with running time $2^{\mathcal{O}(\ell \log \ell)} + n^{\mathcal{O}(1)}$.*

Finally, we restrict the input to *planar graphs*, which are extensively studied in real-life applications. Intuitively speaking, a representation of 3D objects on a 2D surface, such as construction plans and traffic maps, are planar. Many combinatorial problems that are computationally hard in general, are known to be easier on planar graphs. This begs us to question whether or not, planar graphs admit kernels smaller than those shown in Theorems 7–9. Specifically, we obtain the following theorems. (Max-SMTI, Min-SMTI and Max-SRTI are NP-Hard on planar graphs as well. In [15], it is shown that Minimum Maximal Matching is NP-Hard on Planar graphs. As a result the reduction to Max/Min-SMTI in [16] goes through.)

Theorem 10. *Max-SMTI (Min-SMTI) on planar graphs admits a kernel of size $\mathcal{O}(k)$ and an algorithm running in time $2^{\mathcal{O}(\sqrt{k} \log k)} + n^{\mathcal{O}(1)}$.*

Theorem 11. *Max-SRTI on planar graphs admits a kernel of size $\mathcal{O}(\ell)$ and an algorithm running in time $2^{\mathcal{O}(\sqrt{\ell} \log \ell)} + n^{\mathcal{O}(1)}$.*

3.4 Preliminaries

Graph Theory: For two disjoint subsets $S, S' \subseteq V$, we let $E(S, S')$ denote the set of edges with one endpoint in S and the other endpoint in S'. Given a set $S \subseteq V$, we let $E(S)$ denote the set of edges with both endpoints in S. Given a vertex $v \in V$, we let $N_G(v)$ denote the neighbor-set of v. In other words, $N_G(v) = \{u \in V : (u, v) \in E\}$. When G is clear from context, we will omit the subscript G. Given a set $S \subseteq V$, we denote $N_S(v) = N(v) \cap S$. Furthermore, we let $\deg_S(v)$ denote the number of

neighbors of v in S. In other words, $\deg_S(v) = |N_S(v)|$. Given a matching μ in G, we let $\mu(v)$ denote the vertex which is matched to v in μ.

Preferences: Let $G = (V, E)$ be a graph where vertices are associated with preference lists. Given vertices $v, v', x \in V$, we use the notation $v >_x v'$ to indicate that x prefers v over v'. Moreover, we use the notation $v \geq_x v'$ to indicate that either x prefers v over v' or v and v' are tied in x's preference list. Now, we explain how we view preference lists with ties. Given a vertex $v \in V$, the preference list of v can be seen as a strict ordering of ties. For example, a strict preference list of a vertex $v \in V$ corresponds to an ordering of ties such that each tie is of length 1. In other words, the ordering on ties is transitive. This ordering also defines the order in which we process the preference list of any vertex such that the internal order in which we process the vertices in each tie is arbitrary.

Operations that remove edges/vertices. Given a graph $G = (V, E)$ where vertices are associated with preference lists, the operations that delete edges and vertices from G are defined as follows. To delete an edge $(u, v) = e \in E$, remove the edge e from G, remove the vertex u from v's preference list, and remove the vertex v from u's preference list. The order in which u and v rank their remaining neighbors is not altered. Now, to delete a vertex $v \in V$, we first delete each of the edges incident to it (in an arbitrary order), and then we remove the vertex v from G. For a graph G and an edge $uv \in G$, we define the operation of contracting edge uv as follows: we delete vertices u and v from G, and add a new vertex adjacent to all vertices that u or v was adjacent to in G.

Definition 14. *(Graph Minors) A graph H is a minor of graph G if H can be obtained from G by a series of vertex deletions, edge deletions and edge contractions.*

Theorem 12. *(Wagner's Theorem) A graph is planar if and only if it does not have K_5 and $K_{3,3}$ as a minor. (Here K_5 is the complete graph on 5 vertices and $K_{3,3}$ is a complete bipratite graph with 3 vertices in each bipartition.)*

3.5 Algorithms for SMTI

In this section we design kernels for Max-SMTI and Min-SMTI.

3.5.1 Kernel for Max-SMTI

We start by giving a kernel for Max-SMTI with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges, where k is the solution size.

Decomposition Lemma: First, given an instance $\mathcal{I}$ of SMTI, we arbitrarily break all ties. Then, since there are no ties, we can compute a stable matching μ in the new instance. Note that μ is a (weakly) stable matching in $\mathcal{I}$. Indeed, if a pair (x, y) is a block edge for μ in $\mathcal{I}$, then (x, y) is also a blocking edge for μ in $\mathcal{I}'$. Overall, we conclude that if $|\mu| \geq k$, then it is safe to output a trivial Yes-instance. From now onwards, we assume that $|\mu| < k$. This leads us to the following observation.

Observation 1. *μ is a maximal matching.*

Proof. Suppose, by way of contradiction, that the claim is false. Then, there exists an edge outside μ whose endpoints are unmatched. This edge is a blocking edge for μ, a contradiction to the fact that μ is stable. □

For any maximal matching, the set of vertices saturated by the matching edges forms a *vertex cover* of the graph. We use X to denote the set of vertices saturated by the stable matching μ. Thus, X is a vertex cover for the input graph, and it holds that $|X| < 2k$. Furthermore, we know that the maximum matching in a graph is at most twice the size of any maximal matching, and therefore $k \leq |X|$. Hence, if $|X| < k$, then we can output a trivial No-instance, and conclude our argument. From now on, we assume that $k \leq |X| < 2k$.

Overall, we partition $V = A \cup B$ into two sets: the vertex cover X and the independent set $I = V \setminus X$. The maximum number of edges that can lie within X, obtained when the graph induced on X is a complete bipartite graph, is $\mathcal{O}(k^2)$. However, to obtain a kernel for Max-SMTI, we also need to bound the number of vertices in I and the number of edges in $E(X, I)$. Let us first summarize the arguments given so far in the following "decomposition lemma".

Lemma 3. *Given an instance of Max-SMTI, in polynomial time we can either conclude that the instance is a No-instance or a Yes-instance, or decompose the vertex-set V into a vertex cover X and an independent set $I = V \setminus X$ such that $k \leq |X| < 2k$.*

Isolated Vertices: To bound $|I|$ and $|E(X, I)|$, we first present the following simple reduction rule.

Reduction Rule 1. *If G contains an isolated vertex v, then delete v. The new instance is $\mathcal{I}' = (V(\mathcal{I}) \setminus \{v\}, k)$.*

Recall that the operation that deletes a vertex was defined in Section 3.4. To see that this reduction rule is safe, note that isolated vertices do not participate in *any* matching or a blocking edge. An exhaustive application of Reduction Rule 1 takes $\mathcal{O}(n)$ time, where n denotes the number of vertices in the instance. We are now ready to present our base case.

Base Case: For any $x \in X$, consider $E(\{x\}, I)$, the subset of edges whose one endpoint is x and the other endpoint is some vertex in I. Our base case assumes that for every $x \in X$, it holds that $|E(\{x\}, I)| \leq 2k + 1$. Then, since there are no isolated vertices (due to Reduction Rule 1), it holds that $|I| \leq 2k(2k + 1) = \mathcal{O}(k^2)$. Furthermore, $|E(X, I)| < 2k(2k + 1) = \mathcal{O}(k^2)$. Thus, the total number of vertices is $|X| + |I| \leq 2k + |X|(2k + 1) = 4k(k + 1) = \mathcal{O}(k^2)$, and the total number of edges is $|E(X)| + |E(X, I)| \leq |X|^2 + |X|(2k + 1) = 8k^2 + 2k = \mathcal{O}(k^2)$. Overall, the size of the instance is bounded by $\mathcal{O}(k^2)$. Hence, in our base case, it is safe to output $(\mathcal{I}, k)$. From now on, we assume that there exists at least one vertex $x \in X$ such that $|E(\{x\}, I)| > 2k + 1$.

Marking Edges: We proceed by *marking* some of the edges incident to the vertices in X. For each $x \in X$, consider the edges in $E(\{x\}, I)$ that are currently unmarked. If there are less than $2k + 1$ such edges, then mark all of them. Otherwise, mark the edges incident to the $2k + 1$ most preferred neighbors of x in I (where we break ties arbitrarily). The edges with both endpoints in X are considered marked. We say that a vertex in V that is incident to at least one marked edge is a *marked vertex*. Note that if a vertex in X has an unmarked neighbor, then it must have $2k + 1$ marked neighbors in I. Moreover, if a vertex in I is unmarked, then all the edges incident to it are unmarked. Refer to figure 3.2.

We are now ready to present a reduction rule that utilizes our marking scheme.

Reduction Rule 2. *If there exists $x \in X$ with more than $2k+1$ neighbors in I, then delete all the unmarked edges in $E(\{x\}, I)$. (Note that no edge in $E(\{x\}, X \setminus \{x\})$ is deleted, figure 3.3.)*

Lemma 4. *Reduction Rule 2 is safe.*

Proof. Given an instance $(\mathcal{I}, k)$ of Max-SMTI, let $\mathcal{I}'$ denote the instance obtained by applying Reduction Rule 2 on $\mathcal{I}$. We prove that $(\mathcal{I}, k)$ is a Yes-instance of Max-SMTI if and only if $(\mathcal{I}', k)$ is a Yes-instance of Max-SMTI.

($\Leftarrow$) Let μ' denote a stable matching in $\mathcal{I}'$ such that $|\mu'| \geq k$. We will prove that μ' is a stable matching in $\mathcal{I}$.

For the sake of contradiction, let us assume that there is a blocking edge for μ' in $\mathcal{I}$. Clearly this must be an unmarked edge incident to x, since all other edges are present in $\mathcal{I}'$. We denote this edge by (x, y) where $y \in I$. Since x has an unmarked neighbor, it must also have $2k + 1$ marked neighbors. Recall from our earlier discussion that the size of a maximum matching in $\mathcal{I}$, and therefore also in $\mathcal{I}'$, is at most $|X|$. Hence, $|\mu'| \leq |X|$. Therefore, there exists at least one marked neighbor of x who is unmatched in μ', denoted by y'. By our marking scheme, we know that $y' \geq_x y >_x \mu'(x)$, so the marked edge (x, y') is a blocking edge for μ'. Therefore, μ' cannot be stable in $\mathcal{I}'$, a contradiction. Hence, we conclude that μ' is stable in $\mathcal{I}$; thus, ($\Leftarrow$) is proved.

($\Rightarrow$) Let μ denote a stable matching in $\mathcal{I}$ such that $|\mu| \geq k$. If every edge in μ is also present in $\mathcal{I}'$, then μ is a matching in $\mathcal{I}'$. Since the graph G' in $\mathcal{I}'$ is a subgraph of the graph G in $\mathcal{I}$, we have that μ (if present in $\mathcal{I}'$) must also be stable in $\mathcal{I}'$. For the sake of contradiction, we assume that at least one of the edges in μ does not exist in $\mathcal{I}'$. All missing edges from $\mathcal{I}'$ are unmarked edges incident to x. Therefore, μ contains an unmarked edge incident to x, denoted by (x, y), for some $y \in I$. Since $|\mu| \leq |X| \leq 2k$, there exists at least one marked neighbor x, denoted by y', which is unmatched in μ. As before, we know that $y' \geq_x y$, where $y = \mu(x)$. Without loss of generality, we may assume that y' is the most preferred marked neighbor of x that is unmatched in μ, i.e. a more preferred neighbor is matched to some other vertex in μ. Since y' is a marked neighbor of x, we have that (y', x) is present in $\mathcal{I}'$.

We claim that $\mu' = \mu \setminus \{(x, y)\} \cup \{(x, y')\}$ is a stable matching in $\mathcal{I}'$. We prove this by showing that μ' is stable in $\mathcal{I}$, and it is a matching in $\mathcal{I}'$. Together these claims imply that μ' must be stable in $\mathcal{I}'$. Recall that $(y', x) \in \mu'$ is present in $\mathcal{I}'$. All other edges in μ' exist in $\mathcal{I}'$ (as they are not incident to x), and so μ' is a matching in $\mathcal{I}'$. Recall that μ is stable in $\mathcal{I}$, and μ and μ' differ only in the edge incident on x. Therefore, if μ' is not stable in $\mathcal{I}$, then it must be blocked by an edge incident to x. That is, there exists a marked edge (x, y'') such that $y'' >_x \mu'(x) \geq_x \mu(x)$, and $x >_{y''} \mu'(y'')$. Since $\mu(y'') = \mu'(y'')$, we have that (x, y'') must also block μ, a contradiction. Therefore, μ' is stable in $\mathcal{I}$. Since $|\mu'| = |\mu| \geq k$, the direction ($\Rightarrow$) is

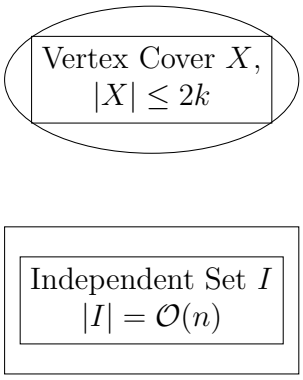

Figure 3.1: Step 1-Decompose graph G into Vertex Cover and Independent Set

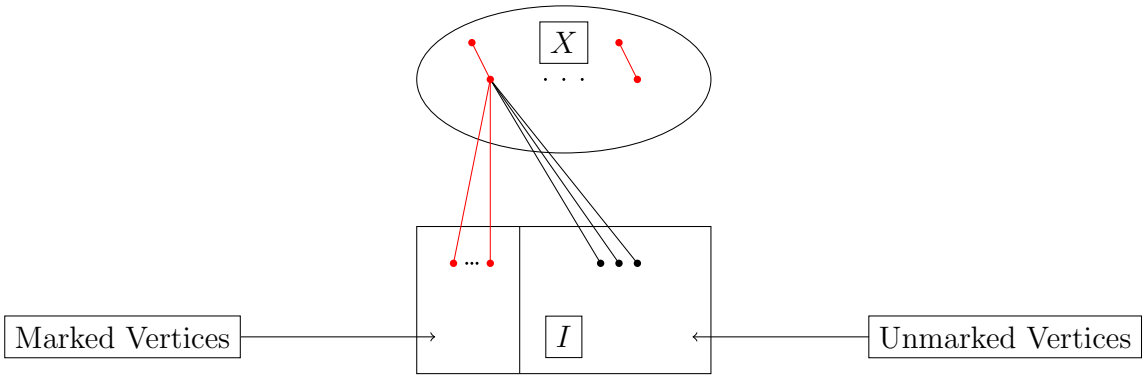

Figure 3.2: Step 2-After deleting Isolated vertices, mark edges and vertices. Red denotes marked vertices and edges. The left partition of the independent set refers to the top $2k+1$ preferred vertices that are marked for every vertex in X. The right partition contains the remaining vertices.

proved.

This completes the proof of Lemma 4. □

Each application of Reduction Rule 2 takes $\mathcal{O}(n)$ time (since we have to scan the neighborhood of a vertex in X). The rule can be applied at most $|X|$ times, therefore the total time taken is $\mathcal{O}(kn)$.

After applying Reduction Rule 2 exhaustively,(refer to figure 3.4) we obtain an instance $(\mathcal{I}, k)$, in which every vertex in X has at most $2k+1$ neighbors in I, all of which are marked. This instance is of the form handled by our Base Case. By the discussion in the introduction, we conclude the correctness of Theorem 8.

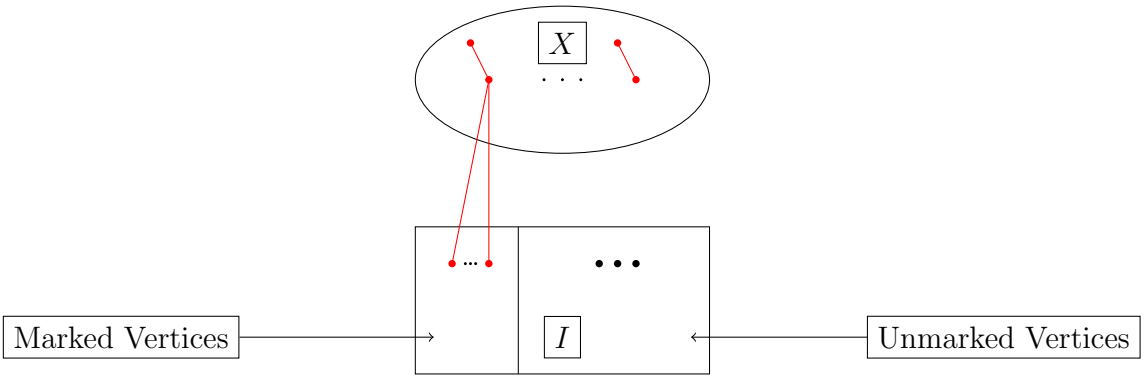

Figure 3.3: Step3-Delete all marked edges

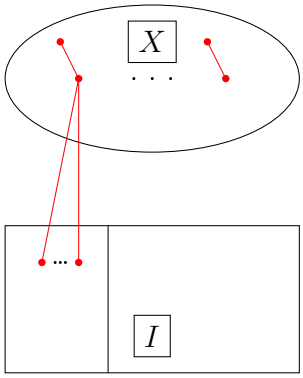

Figure 3.4: Step 4-Apply Reduction Rule 1 again and now delete all the isolated vertices. This remaining graph is the *kernel*.

3.5.2 Kernel for Min-SMTI

Here, we give a kernel for Min-SMTI with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges, where k is the solution size.

As in Section 3.5.1, we first obtain a (weakly) stable matching μ in $\mathcal{I}$. However, now we output a trivial Yes-instance if $|\mu| \leq k$ rather than $|\mu| \geq k$.

Suppose that $|\mu| > k$. By Observation 1, μ is a maximal matching. Recall that the size of a maximum matching in G is at most twice the size of any maximal matching. Thus, if $\mathcal{I}$ is a Yes-instance-instance—that is, there exists a stable matching (which is maximal) of size at most k—then the size of any maximal matching in $\mathcal{I}$ cannot exceed $2k$. Hence, if $|\mu| > 2k$, then we will output a trivial No-instance and conclude

our argument. From now onwards, we assume that $k < |\mu| \leq 2k$. As in Section 3.5.1, we obtain a decomposition lemma.

Lemma 5. *Given an instance of Min-SMTI, in polynomial time we can either conclude that the instance is a No-instance or a Yes-instance, or decompose the vertex-set V into a vertex cover X and an independent set $I = V \setminus X$ such that $2k \leq |X| < 4k$.*

Recall that in the analysis of Max-SMTI (in Section 3.5.1), we have used the assumption that $k < |X| \leq 2k$. However, the construction of the kernel for Min-SMTI is very similar to the one presented for Max-SMTI; we will discuss the main differences, but skip the details.

First, note that Reduction Rule 1 is applicable here, and it is evidently safe. Then, in our Base Case, we assume that for every $x \in X$, it holds that $|E(\{x\}, I)| \leq k+1$. In this case, we conclude that the total number of vertices is bounded by $4k(k+2)$, and the total number of edges is bounded by $20k^2 + 4k$. Next, for every $x \in X$, we mark the more preferred $k+1$ neighbors of x in the independent set I. The proof of the safeness of the reduction rule below follows from arguments almost identical to those in the proof of Lemma 4—essentially, we replace occurrences of $2k$ by occurrences of k.

Reduction Rule 3. *If there exists $x \in X$ with more than $k+1$ neighbors in I, then delete all the unmarked edges in $E(\{x\}, I)$. (Note that no edge in $E(\{x\}, X \setminus \{x\})$ is deleted.)*

After applying Reduction Rule 3 exhaustively, we obtain an instance of the form handled by our Base Case. By the discussion in the introduction, we conclude the correctness of Theorem 8.

3.6 Algorithms for SRTI

The analysis of Max-SRTI differs from that of Max-SMTI since a stable matching may not exist in an instance of Max-SRTI. In this context, recall that we have argued (in the introduction) that unless P=NP, we cannot obtain an FPT algorithm for Max-SRTI when the parameter is the solution size. Here, we describe a polynomial sized kernel for Max-SRTI, where the parameter ℓ is the size of a maximum matching in the graph G. Our kernel will consist of $\mathcal{O}(\ell^2)$ vertices and $\mathcal{O}(\ell^2)$ edges.

If $k > \ell$, then we can output a trivial No-instance: since ℓ is the size of the maximum matching of the graph, we cannot obtain a stable matching, or any matching, of size at least k. Thus, we next focus on the case where $k \leq \ell$. Let μ denote any maximal matching in the graph $G = (V, E)$ in $\mathcal{I}$. Let X denote the set of vertices matched by μ, and let I denote the set of vertices outside X. As discussed in Section 3.5.1, X is a vertex cover for G such that $|X| \leq 2|\mu| \leq 2\ell$, and I is an independent set.

We begin our construction by removing isolated vertices. The safeness of this rule is self-evident.

Reduction Rule 4. *If G contains an isolated vertex v, then delete v.*

We proceed by marking certain edges incident to vertices in X, after which we discuss our base case. For each $x \in X$, consider the set of edges in $E(\{x\}, I)$. If there are less than $\ell + 1$ such edges, then mark all of them. Otherwise, mark the edges incident on the $\ell + 1$ most preferred neighbors of x in I (where we break ties arbitrarily). The vertices in I that are incident to at least one marked edge are called *marked vertices*, and the others are called unmarked.

Reduction Rule 5. *If there exists $x \in X$ with more than $\ell + 1$ neighbors in I, then delete all the unmarked edges in $E(\{x\}, I)$.*

Lemma 6. *Reduction Rule 5 is safe.*

Proof. Note that $k \leq \ell \leq |X|$ and any matching in $\mathcal{I}$ or $\mathcal{I}'$, the instance obtained by applying Reduction Rule 5, is of size at most ℓ. Thus, the proof of this lemma is very similar to the one presented for Lemma 4. We skip the details. □

Each application of Reduction Rules 4 and 5 takes $\mathcal{O}(n)$ time (recall that $n = |V|$). Each of these rules can be applied at most n times. Therefore, the total running time is bounded by $\mathcal{O}(n^2)$.

After applying Reduction Rule 5 exhaustively, we obtain an instance $(\mathcal{I}, k)$ where every vertex in X has at most $\ell + 1$ neighbors in I, all of which are marked. A vertex in I that is not incident to any marked edge is in fact an isolated vertex. Therefore, Reduction Rule 1 is applied to remove them from the instance. This final step yields an instance $(\mathcal{I}', k)$ in which there are $|X| + |I| \leq 2ell + (\ell + 1)2l = \mathcal{O}(\ell^2)$ vertices, and $|E(X)| + |E(X, I)| \leq |X|^2 + |X|(\ell + 1) = \mathcal{O}(\ell^2)$ edges. Therefore, at this point, we output the instance $(\mathcal{I}', k)$ and conclude our argument. With this we have proven Theorem 9.

3.7 SMTI and SRTI on planar graphs

In Section 3.5, we designed polynomial kernels for Max-SMTI and Min-SMTI of size $\mathcal{O}(k^2)$, where k is the solution size. Furthermore, in Section 3.6 we designed a polynomial kernel for Max-SRTI of size $\mathcal{O}(\ell^2)$, where ℓ is the size of a maximum matching of the input graph G. In this section we improve all these kernels, as well as design faster FPT algorithms, when the input is restricted to planar graphs. The kernel and the faster FPT algorithm we obtain for Max-SMTI, Min-SMTI and Max-SRTI in this section are very similar to each other. Thus, we focus only on Max-SMTI.

Given an input instance $(\mathcal{I}, k) = (A, B, \mathcal{L}^A, \mathcal{L}^B, k)$ of Max-SMTI, we first apply Lemma 3, and in polynomial time we either conclude that the instance is a No-instanceor a Yes-instance, or decompose the vertex-set V into a vertex cover X and an independent set $I = V \setminus X$ such that $k \leq |X| < 2k$. Next, we design a kernel such that $|I|$ is bounded by $\mathcal{O}(k)$. In particular, this would overall give us a kernel where $|V| = \mathcal{O}(k)$ and $|E| = \mathcal{O}(k)$

Towards our goal, we partition the independent set I into three sets: $I_{\deg=1}$, $I_{\deg=2}$ and $I_{\deg\geq 3}$. These sets are defined as follows:

$$\begin{aligned} I_{\deg=1} &= \{v \in I \mid \deg_X(v) = 1\} \\ I_{\deg=2} &= \{v \in I \mid \deg_X(v) = 2\} \\ I_{\deg\geq 3} &= \{v \in I \mid \deg_X(v) \geq 3\} \end{aligned}$$

Furthermore, we define

$$\begin{aligned} Y_{\leq 1} &= \{N(v) \mid v \in I \text{ and } \deg_X(v) \leq 1\} \\ Y_{=2} &= \{N(v) \mid v \in I \text{ and } \deg_X(v) = 2\} \\ Y_{\geq 3} &= \{N(v) \mid v \in I \text{ and } \deg_X(v) \geq 3\} \end{aligned}$$

Observe that since I is an independent set, all the neighbors of the vertices in I are in X, and thus for any $v \in I$, we have that $\deg_V(v) = \deg_X(v)$. To upper bound $|I_{\deg=1}|$, $|I_{\deg=2}|$ and $|I_{\deg\geq 3}|$, we will use the following known result from [13]. Since the proof is not provided in [13], we give a proof here.

Lemma 7 ([13, Lemma 1])**.** *Let G be a planar graph and let X be a vertex cover in G. Then, $|\{N(v) \mid v \notin X\}| \leq 6|X| + 1$. In particular, $|Y_{\leq 1}| \leq |X| + 1$, $|Y_{=2}| \leq 3|X|$ and*

$|Y_{\geq 3}| \leq 2|X|$.

Proof. We assume that the conditions stated in the lemma hold, and denote $I = V \setminus X$ (which is an independent set). We count the number of "open neighborhoods" in three steps. We first bound $Y_{\leq 1}$, then $Y_{=2}$ and finally $Y_{\geq 3}$. Overall, the quantity we need to bound is $|Y_{\leq 1}| + |Y_{=2}| + |Y_{\geq 3}|$. Obviously, there are at most $|X| + 1$ distinct open neighborhoods of one vertex or less, so $|Y_{\leq 1}| \leq |X| + 1$.

To bound $|Y_{=2}|$, consider the graph $G_2 = (V_2, E_2)$ on the vertex-set $V_2 = X$, with an edge between $u, v \in X$ if there is a vertex $y \in Y_{=2}$ such that $N_G(y) = \{u, v\}$. The resulting graph is a minor of G (therefore, it is planar), and every neighborhood in $Y_{=2}$ corresponds to a unique edge in G_2. By Euler's formula, the number of edges in a simple planar graph is at most three times the number of vertices, and we find that $|Y_{=2}| = |E_2| \leq 3|X|$.

Finally, to bound $|Y_{\geq 3}|$, consider the bipartite graph $G_3 = (V_3, E_3)$ that is the subgraph of G obtained by removing the edges in $G[X]$ (the subgraph of G induced by X), removing the vertices in I that have degree less than 3, and keeping exactly one vertex in each twin class in the remainder of I (that is, if there are at least two vertices with the same neighbourhood, then we delete all but one). The resulting bipartite graph is planar, with one partite set corresponding to X, while the other partite set has a vertex for every open neighborhood in $Y_{\geq 3}$. The number of vertices in G_3 is therefore $|X| + |Y_{\geq 3}|$. Another application of Euler's formula shows that the number of edges in a simple bipartite planar graph on n vertices is at most $2n$. Since every vertex in $Y_{\geq 3}$ has degree at least 3 in G_3 we find $|E_3| \geq 3|Y_{\geq 3}|$. Combining this with the upper bound on the edge count we find $3|Y_{\geq 3}| \leq |E_3| \leq 2(|X| + |Y_{\geq 3}|)$, which implies that $|Y_{\geq 3}| \leq 2|X|$.

Since $|\{N(v) \mid v \notin X\}| = |Y_{\leq 1}| + |Y_{=2}| + |Y_{\geq 3}|$ the bound of $6|X| + 1$ follows. □

To bound the sizes of $I_{\deg=1}$ and $I_{\deg=2}$, we need to apply new reduction rules. We first give a rule that will enable us to bound the size of $I_{\deg=1}$. Towards this end, we will design a reduction rule that allows us to remove all but a small subset of vertices in $I_{\deg=1}$. For each $x \in X$, consider the set $N_I(x) \cap I_{\deg=1}$, containing the degree-1 neighbors of x in I. For each $x \in X$, we remove all but the most preferred neighbor from this set. The reduction rule is stated as follows.

Reduction Rule 6. *Let $x \in X$, and let u denote x's most preferred neighbor (breaking ties arbitrarily) in $N_I(x) \cap I_{\deg=1}$. Then, delete all vertices in $N_I(x) \cap I_{\deg=1}$ except*

u.

Lemma 8. *Reduction Rule 6 is safe.*

Proof. We will prove that if $(\mathcal{I}, k)$ is a Yes-instance if and only if $(\mathcal{I}', k)$ is also a Yes-instance.

Let μ denote a stable matching in $\mathcal{I}$ such that $|\mu| \geq k$. If $\mu(x) \notin I_{\deg=1}$, then μ is a matching in $\mathcal{I}'$. Also μ is stable in $\mathcal{I}'$, since $\mathcal{I}'$ is a subgraph of $\mathcal{I}$. Now, if $\mu(x) = u$, then also μ is a matching in $\mathcal{I}'$, and so is stable. We are left with the case that $\mu(x) \neq u$ and $\mu(x) \in I_{\deg=1}$. In this case, u is unmatched in μ. If $u >_x \mu(x)$, then (u, x) blocks μ in $\mathcal{I}$, a contradiction. Therefore, u and $\mu(x)$ must be in a tie in x's preference list. But then, the matching $\mu' = \mu \setminus \{(x, \mu(x))\} \cup \{(x, u)\}$ is also stable in $\mathcal{I}$ and μ' is a matching in I'. Hence μ' is also a stable matching in $\mathcal{I}'$. This proves the ($\Rightarrow$) direction.

Let μ' denote a stable matching in $\mathcal{I}'$ such that $|\mu'| \geq k$. We claim that μ' is also stable in $\mathcal{I}$. Suppose not, then there exists a blocking edge for μ' in $\mathcal{I}$. Since $\mathcal{I}'$ defines a subgraph in $\mathcal{I}$ and μ' is a stable matching in the former, it follows that the blocking edge for μ' in $\mathcal{I}$ must be one of the edges that were removed during the application of Reduction Rule 6. Let (x, w) denote the blocking edge in $\mathcal{I}$, where $w \in N_I(x) \cap I_{\deg=1} \setminus \{u\}$ such that $w >_x \mu'(x)$. By the definition of u, we know $u \geq_x w >_x \mu'(x)$. Since the edge (x, u) exists in $\mathcal{I}'$, this implies that (x, u) blocks μ' in $\mathcal{I}'$. Thus, we have arrived at a contradiction. We can conclude that μ' is a stable matching in $\mathcal{I}$, and ($\Leftarrow$) is proved. □

Reduction Rule 6 is used to bound the size of $I_{\deg=1}$. Now we give our final rule that will enable us to bound the size of $I_{\deg=2}$.

We define an equivalence relation $\sim$ on the set $I_{\deg=2}$ as follows: for a pair of vertices $u, v \in I_{\deg=2}$, $u \sim v$ if and only if $N_G(u) = N_G(v)$, that is they have the same neighborhood. Let $\mathcal{E}_i$ denote the i^{th} equivalence class defined by the relation $\sim$. Let the total number of equivalence classes be $\mathcal{N}$.

Reduction Rule 7. *Let $x \in X$. For each equivalence class $\mathcal{E}_i$ ($1 \leq i \leq \mathcal{N}$) that contains exactly one neighbor of x, mark the singleton vertex in the class. For all other classes that contain neighbors of x, mark the two most preferred neighbors, breaking ties arbitrarily. Delete all edges between x and its unmarked neighbors.*

Lemma 9. *Reduction Rule 7 is safe.*

Proof. Let μ denote a stable matching in $\mathcal{I}'$ of size at least k. Clearly μ is a matching in $\mathcal{I}$, since the graph in $\mathcal{I}'$ is a subgraph of $\mathcal{I}$. Suppose that μ is not stable in $\mathcal{I}$, then the blocking edge(s) must be (one of) the deleted edge(s). Let (x, u) denote a blocking edge for μ in $\mathcal{I}'$, where u is an unmarked neighbor of x such that $u >_x \mu(x)$. The equivalence class containing u (say $\mathcal{E}_i$) must have two marked neighbors (denoted by u_{i_1} and u_{i_2}) such that $u_{i_1} \geq_x u > \mu(x)$, and $u_{i_2} \geq_x u > \mu(x)$, i.e., both are preferred by x at least as much as u. Thus, implies that $\mu(x) \notin \{u_{i_1}, u_{i_2}\}$. Recall that these vertices are in $I_{\deg=2}$, so they have exactly two neighbors, one of which is x. Therefore, at least one of the marked neighbors (say u_{i_1}) must be unmatched in μ. But then, (x, u_{i_1}) is a blocking edge for μ in $\mathcal{I}'$, a contradiction. Hence, ($\Leftarrow$) direction is proved.

For the other direction, let μ denote a stable matching in $\mathcal{I}$ of size at least k. If μ is also a matching in $\mathcal{I}'$ then it is also stable in $\mathcal{I}'$. Therefore, we assume that μ contains an edge incident on x that was deleted during the application of Reduction Rule 7. Let (x, u) denote the edge in μ that does not exist in $\mathcal{I}'$. Let $\mathcal{E}_i$ denote the equivalence class containing u. As argued in the earlier case, since u was deleted (it is an unmarked neighbor of x in $\mathcal{E}_i$), x has two marked neighbors in $\mathcal{E}_i$ whom x prefers at least as much as u. Furthermore, at least one of the marked neighbors must be unmatched in μ. Let u_{i_1} denote the marked neighbor in $\mathcal{E}_i$ who is unmatched in μ. Since μ is stable in $\mathcal{I}$, so u and u_{i_1} and must be in a tie in x's preference list. This allows us to claim that the matching $\mu' = \mu \setminus \{(x, u)\} \cup \{(x, u_{i_1})\}$ is also stable in $\mathcal{I}$. Since μ' exists in $\mathcal{I}'$, hence it is also stable in $\mathcal{I}'$. This completes the proof of ($\Rightarrow$) direction. Thus, Lemma 9 is proved. □

For our kernelization algorithm, we apply Reduction Rules 1, 6, and 7 exhaustively. Since each of the reduction rule either deletes an edge or a vertex, the number of times we can apply these rules is bounded by $\mathcal{O}(n)$. Furthermore, since each of the reduction rule can be applied in polynomial time, the kernelization algorithm runs in polynomial time.

Bounding the size. Let $(\mathcal{I}, k) = (A, B, \mathcal{L}^A, \mathcal{L}^B, k)$ denote the reduced instance of Max-SMTI. That is, an instance on which Reduction Rules 1, 6, and 7 are no longer applicable. Since Reduction Rule 1 is not applicable there are no isolated vertices. Thus, Reduction Rule 6, in conjunction with Lemma 7 yields the following relation.

Lemma 10. $|I_{\deg=1}| \leq |Y_{\leq 1}| \leq |X| + 1$.

Recall, that each vertex in each equivalence class sees *exactly* two neighbours in

X. This implies that after application of Reduction Rule 7, each equivalence class has at most four vertices. Since the number of equivalence classes is upper bounded by $|Y_2| \leq 3|X|$, we get the following result.

Lemma 11. $|I_{\deg=2}| \leq 4|Y_2| \leq 12|X|$.

Now we bound the size of $I_{\deg\geq 3}$. By Lemma 7 we know that $|Y_{\geq 3}| \leq 2|X|$. Note that for any $W \in Y_{\geq 3}$, there can be at most two vertices in I whose neighborhood is W. Otherwise, $K_{3,3}$ is a minor of G, a contradiction to its planarity. As a consequence, we obtain the following result.

Lemma 12. $|I_{\deg\geq 3}| \leq 2|Y_{\geq 3}| \leq 4|X|$.

Therefore, applications of Reduction Rules 1, 6 and 7 yields an instance in which the independent set I has size $|I| = |I_{\deg=1}| + |I_{\deg=2}| + |I_{\deg\geq 3}| \leq 17|X| + 1$, thereby giving the following result.

Lemma 13. *Max-SMTI parameterized by the solution size k admits $\mathcal{O}(k)$ sized kernel on planar graphs.*

Similar to the proof of Lemma 13, we can design an $\mathcal{O}(k)$ and $\mathcal{O}(\ell)$ sized kernels for Min-SMTI and Max-SRTI, respectively, on planar graphs.

Chapter 4

Parameterized Algorithms for the Hospital/Residents Problem with Lower Quota

This chapter is based on the work done in [12].

4.1 Introduction

In section 1.3 we had defined the Hospital/Residents Problem. The Hospital/Residents Problem with lower quota(HR-LQ, in short) involves a set H of hospitals and R . Every hospital has a preference list over residents i.e., a total ordering of a subset $R' \subseteq R$. Similarly every resident has a preference list over hospitals which is again a total ordering of a subset $H' \subseteq H$. Each hospital $h \in H$ has a **lower quota** and **upper quota** associated to it such that the former does not exceed the latter. The upper quota represents the maximum number of residents it can accommodate. A **matching** is an assignment of residents to hospitals such that every resident is assigned to exactly one hospital and no hospital is assigned more than its upper quota number of residents. A **feasible matching** is a matching that satisfies the additional constraint that every hospital is assigned at least its lower quota number of residents. An assignment/matching that is not feasible, is said to be an **infeasible matching**.

Let μ denote a matching in an instance of HR-LQ. We will abuse notations when representing the matching partners of hospitals and resident in μ. For any resident

$r \in R$, we will use $\mu(r)$ to denote the hospital to which r is assigned in μ. But for a hospital $h \in H$, we will use $\mu(h)$ to denote the subset of residents assigned to h in μ. This relationship is symmetric in the sense that $\mu(r) = h$ if and only if $r \in \mu(h)$. A **blocking pair** for a matching μ is a pair of resident and hospital (r, h) not matched to each other in μ that satisfy one of the following conditions:

1. r is unmatched and h prefers r over one of its assigned residents, or r prefers h over $\mu(r)$ and h prefers r over one of its assigned residents.

2. h has not satisfied its upper quota and r prefers h over $\mu(r)$ (we say $\mu(r) = r$ if r is unmatched and r prefers being matched to any hospital than beng matched to itself).

A matching/assignment that has no blocking pair is called a stable matching. For any instance of HR where the lower quota of every hospital is zero, we can use the Gale-Shapley algorithm to find a stable matching. Given an instance $\mathcal{I}$ create an instance $\mathcal{I}'$ as follows. For every $h \in H$ with upper quota u_h, create u_h copies of h each having the same preference list as h. In the residents lists, expand each h into u_h copies and arbitrarily order them. Gale Shapley applied to this new instance $\mathcal{I}'$ gives a matching that is stable in $\mathcal{I}$ [2]. Unless the lower quota for every hospital is zero, the stable matching found need not be feasible. By the Rural Hospital Theorem we know that every stable matching matches the same set of agents. So if one stable matching is infeasible, then all stable matchings for the instance are infeasible.

If a stable matching is infeasible, then we want a matching that is "as close to a stable matching as possible". We quantify this closeness in two ways : (i) minimizing the number of blocking pairs, (ii) minimizing the number of **blocking residents** (the subset of residents who are part of a blocking pair). Each quantification leads to an optimization problem, stated below. Both these optimization problems are known to be NP-hard [8].

In each of the following problems, the input is an instance of HR-LQ, containing a set of residents R, their preference lists $\mathcal{L}^R$, a set of hospitals H, their preference lists $\mathcal{L}^H$, and their upper and lower quotas $\{(l_h, u_h)\}_{h \in H}$.

HR with min blocking pairs (min-BP-HRLQ) **Input:** An instance of HR-LQ, integer k **Question:** Does there exist a feasible matching with at most k blocking edges?

HR with min blocking residents (min-BR-HRLQ)
Input: An instance of HR-LQ, integer k
Question: Does there exist a feasible matching with at most k blocking residents.

In this chapter we look at the parameterized complexity of the above problems for the following parameters. We will formally define the parameterized problems in the respective sections containing the algorithm or W[1] hardness.

1. r, the number of blocking residents
2. b, the number of blocking edges
3. $\sum_i l_i$, the sum of lower quotas over all hospitals
4. $|H^*|$, the number of hospitals with positive lower quota

4.2 An XP-Algorithm

In this section we give an XP-algorithm for the problem min-BR-HRLQ parameterized by the number of blocking residents.

HR-LQ parameterized by blocking residents (p-HR-BR)
Input: An instance of HR-LQ, and a positive integer k.
Parameter: k
Task: Find (if exists) a feasible matching with at most k blocking residents.

We start with an instance $(\mathcal{I}, k)$ of the problem and an integer k . The outline of the algorithm is as follows:

We view the instance $\mathcal{I}$ of HR-LQ as a graph $G = (H \cup R, E)$, where H and R denote the two disjoint set of vertices. For each $r \in R$, the "neighborhood" of r contains all the hospitals that are in the preference list of r. So for every $h \in \delta(r)$, the graph has the edge (r, h).

1. We guess a subset R' of residents R of size k, representing the blocking residents.
2. For every $r_i \in R'$, we guess h_i its matching partner.

3. Delete all edges between r_i and $\delta(r)$ except the edge (r_i, h_i), modify the preference lists accordingly. This gives the new instance $\mathcal{I}'$.

4. Find a stable matching μ in $\mathcal{I}'$.

5. If μ is a feasible matching in $\mathcal{I}$, then we have found a output that $\mathcal{I}$ is a YES instance of p-HR-BR, else output that $\mathcal{I}$ is a NO instance of p-HR-BR.

Algorithm 5 gives the formal description.

Algorithm 5 XP-algorithm for p-HR-BR

```
1: procedure OPT(Graph G = (R ∪ H, E), preference list for every vertex, quotas
   {(l_h, u_h) | h ∈ H}, and positive integer k.)
2:     if k ≥ Σ_i l_i then
3:         if I has a feasible matching then
4:             return Yes-instance
5:         else
6:             return No-instance
7:         end if
8:     end if
9:     for subset R' = {r_1, ..., r_k} ⊆ R do
10:        for A_R' = {(r_i, h_i) ∈ E | 1 ≤ i ≤ k} do
11:            Let E_A_R' = E \ ∪_{i=1}^{k} {(r_i, h) ∈ E | h ≠ h_i}
12:            Let I' be the restriction of I to (R ∪ H, E_A_R').
13:            Let μ be a stable matching in I'.
14:            if μ is a feasible matching in I' then
15:                return Yes-instance
16:            end if
17:            return No-instance
18:        end for
19:    end for
20: end procedure
```

We will now prove the correctness of the algorithm. We begin by proving the following two lemmas. For any matching μ, we use $BR(\mu)$ to denote the blocking residents in μ.

Lemma 14. *If $(\mathcal{I}, k)$ is a Yes-instance of p-*HR-BR *then there exists a feasible matching in $\mathcal{I}$ in which the number of blocking residents is at most $\sum_i l_i$.*

Proof. Let $\mathcal{I}$ denote a Yes-instance of p-HR-BR. We show that there exists a feasible matching μ such that $BR(\mu) \leq \sum_i l_i$. Let μ_0 denote a matching that matches every hospital h_i to l_i residents. Such a μ_0 exists since $(\mathcal{I}, k)$ is a Yes-instance. Let us call this subset of assigned residents $\hat{R}$. For every $r \in \hat{R}$ delete edges from r to all its neighbours except the hospital $\mu_0(h)$ it is assigned to by μ_0. Let us call this new instance $\mathcal{I}'$. Now find a stable matching μ in $\mathcal{I}'$.

Claim 1. *The matching μ in $\mathcal{I}'$ is a feasible assignment.*

Proof. We know that if μ matches every resident in $\hat{R}$ then it is feasible (by construction). Every resident in $\hat{R}$ has only one hospital as its neighbour in $\mathcal{I}'$. Suppose a hospital $h \in H$ has not attained its lower quota l by the matching μ. Then at least one of its neighbours $r \in L$ must be left unmatched (r has only one neighbour). This means that the matching μ is not maximal which is a contradiction since every stable matching must be maximal. □

Claim 2. *The matching μ has at most $\sum_i l_i$ blocking residents in $\mathcal{I}$.*

Proof. The matching μ is stable in $\mathcal{I}'$ and hence has no blocking edges in $\mathcal{I}'$. When perceived as a matching in $\mathcal{I}$, the only blocking edges can be the edges that are in $\mathcal{I}$ but not in $\mathcal{I}'$, i.e. the edges deleted from $\mathcal{I}$. But we have only deleted the edges incident on residents in $\hat{R}$ and $|\hat{R}| = \sum_i l_i$. So at most $\sum_i l_i$ residents can be involved in blocking edges. □

With this we have proven Lemma 14. □

The algorithm involves guessing the matched partner h_i of every $r_i \in R'$. Let H' denote the set of the guessed matching partners of $r_i \in R'$. For every R' and H', define

$$\mathcal{I}'_{R',H'} = \mathcal{I} \setminus \{ \bigcup_{r_i \in R'} \{ \bigcup_{\hat{h} \in \delta(r_i), \hat{h} \neq h_i} (r_i, \hat{h}) \} \}$$

Lemma 15. *$\mathcal{I}$ is a Yes-instance if and only if there exist sets R' and H' such that a stable matching in $\mathcal{I}'_{R',H'}$ is feasible .*

Proof. We have a set R' and the set of all matching partners of $r \in R'$ i.e. set H'. Now the instance $\mathcal{I}'_{R',H'}$ contains only one edge incident on every resident in R'.
($\Rightarrow$) $\mathcal{I}$ is a Yes-instance. So we have a subset R'' of residents of size k such that there exists a feasible matching μ and all blocking edges of μ are incident on R'' only. Let $R' = R''$ and $H' = \{\cup_{r_i \in R''} \mu(r_i)\}$. let $\mathcal{I}'_{R',H'}$ be defined as above.

Claim 3. *the matching μ is stable in $\mathcal{I}'_{R',H'}$.*

Proof. All blocking edges for μ in $\mathcal{I}$ are incident to residents in R' (by construction). In $\mathcal{I}'_{R',H'}$ we have deleted all neighbours of $r' \in R'$ except its matched partner. For every other resident $r \in R \setminus R'$ we know that there is no blocking edge incident on r. Since we have deleted all blocking edges for μ while defining $\mathcal{I}'_{R',H'}$, the matching μ is stable in $\mathcal{I}'$. □

We know that μ is a feasible matching in $\mathcal{I}$ and hence also a feasible matching in $\mathcal{I}'_{R',H'}$. Claim 3 shows that μ is stable in $\mathcal{I}'_{R',H'}$. With this we have proven the forward direction of lemma 15. For the reverse direction we show that if for some subsets R', H', we can find a feasible solution for $\mathcal{I}'_{R',H'}$, then $\mathcal{I}$ is a Yes-instance.

($\Leftarrow$) For some R' and H' ($|R'| = k$), Let μ denote a feasible stable matching in $\mathcal{I}'_{R',H'}$. The matching μ has no blocking edges in $\mathcal{I}'_{R',H'}$. Let us look at the matching μ in $\mathcal{I}$. μ is clearly a feasible matching in $\mathcal{I}$. Also, all blocking edges for μ are incident on R' only since these are the only edges that are in $\mathcal{I}$ and not in $\mathcal{I}'_{R',H'}$. We know that $|R'| = k$. So R' is the set of blocking students of size k and μ is the required solution for $\mathcal{I}$. This proves the reverse direction of lemma 15. □

By lemma 14 we know that if there exists a feasible matching in $\mathcal{I}$ then there always exists a feasible matching with fewer than $\sum_i l_i$ blocking residents. So if our input integer $k > \sum_i l_i$ then we just check if there exists some feasible matching in the instance. Lemma 15 implies that for a correct guess the instances $\mathcal{I}$ and $\mathcal{I}'$ (created by the algorithm) are equivalent. Since we are guessing all possible subsets and matching partners, if $\mathcal{I}$ is a Yes-instance, one of the guesses will correspond to the correct guess. With this we can say the Algorithm 5 gives the correct output.

Let us now look at the time complexity of this algorithm. Let the input instance have size n. Guessing k sized subsets R' of residents from n residents will take $\mathcal{O}(n^k)$ time. Next we guess the matching partner of each of these residents requires another $\mathcal{O}(n^k)$ time. After this we create the instance $\mathcal{I}'$ and apply Gale-Shapley on $\mathcal{I}'$ which is polynomial in n. We next check for feasibility of the obtained matching which is

again polynomial in n. So our algorithm has a running time of $\mathcal{O}(n^k)$, which implies algorithm 5 is in the class XP.

4.3 W[1]-Hardness Results

In this section we show that min-BP-HRLQ and min-BR-HRLQ are W[1] hard for different parameters. Let us first define a few problems.

Stable Marriage with Capacity constraint (SMC)
Input: A stable marriage instance, a subset W^* of women and M^* of men, an integer k.
Question: Does there exist a matching that matches every vertex in $W^* \cup M^*$ and has at most k blocking edges?

The above problem is known to be NP-Hard and its parameterized complexity has been studied in [14]. The following parameterized problem has been shown to be W[1] hard in Theorem 3 of [14].

p-BE-SMC
Input: A stable marriage instance, a subset W^* of women an integer k.
Parameter: $k + |W^*|$
Question: Does there exist a matching that matches every vertex in W^* and has at most k blocking edges?

From the reduction from Multi-colored clique to p-BE-SMC given in [14], we observe that the blocking edges are vertex disjoint in the SMC instance created in the reduction. The construction of the gadget ensures that the number of blocking edges and blocking men remains the same. We can show a similar reduction using the same gadget and same properties of the construction and show the following problem is also W[1] hard.

p-BM-SMC
Input: A stable marriage instance, a subset W^* of women an integer k.
Parameter: $k + |W^*|$
Question: Does there exist a matching that matches every vertex in W^* and has exactly k men involved in a blocking edge?

We now define the following parameterized versions of the HR-LQ problem.

p-BP-HRLQ
Input: An instance of HR-LQ, integer k.
Parameter: $k + \sum_i l_i + |H^*|$
Question: Does there exist a feasible assignment with at most k blocking pairs?

p-BR-HRLQ
Input: An instance of HR-LQ, integer k.
Parameter: $k + \sum_i l_i + |H^*|$
Question: Does there exist a feasible assignment with exactly k residents involved in blocking pairs?

Theorem 13. *The problem p-BP-HRLQ is W[1] hard.*

Proof. We give a parameterized reduction from p-BE-SMC to p-BP-HRLQ. Let $\mathcal{I} = M \cup W$ be an instance of p-SMC. We create an instance $\mathcal{I}'$ of p-BP-HRLQ. The instance $\mathcal{I}'$ has the same underlying graph as $\mathcal{I}$ where the set W of women correspond to the set H of hospitals and the set M corresponds to the set R of residents. Let H^* denote the hospitals in $\mathcal{I}'$ that correspond to the women W^* in $\mathcal{I}$. For every hospital $h \in H$ set the upper quota $u_h = 1$. The lower quota for hospitals $h \in H \setminus H^*$ is 0 and for $h \in H^*$ is 1. Observe that a feasible assignment μ in $\mathcal{I}$ when viewed in $\mathcal{I}'$ matches the set H^* and has the same number of blocking edges. Also a feasible assignment μ' in $\mathcal{I}'$ when viewed in $\mathcal{I}$ matches the set W^* and has the same number of blocking edges. This implies that $(\mathcal{I}, k)$ is a Yes-instance of p-BE-SMC if and only if $(\mathcal{I}', k')$ is a Yes-instance of p-BP-HRLQ. Now the parameter $k' = k + \sum_i l_i + |H^*| \leq k + |W^*| + |W^*|$

(since the lower capacity is one only for the hospitals in H^*. This concludes the proof of the theorem. □

Theorem 14. *The problem p-BR-HRLQ is W[1] hard.*

Proof. We give a parameterized reduction from the problem p-BM-SMC. Let $(\mathcal{I}, k)$ be an instance of p-BM-SMC. We create an instance $(\mathcal{I}', k')$ of p-BR-HRLQ as follows. The instance $\mathcal{I}'$ has the same underlying graph as $\mathcal{I}$ where the set W of women correspond to the set H of hospitals and the set M corresponds to the set R of residents. Let H^* denote the hospitals in $\mathcal{I}'$ that correspond to the women W^* in $\mathcal{I}$. For every hospital $h \in H$ set the upper quota $u_h = 1$. The lower quota for hospitals $h \in H \setminus H^*$ is 0 and for $h \in H^*$ is 1. Observe that a feasible assignment μ in $\mathcal{I}$ when viewed in $\mathcal{I}'$ matches the set H^* and has the same number of blocking residents. Also a feasible assignment μ' in $\mathcal{I}'$ when viewed in $\mathcal{I}$ matches the set W^* and has the same number of blocking residents. This implies that $(\mathcal{I}, k)$ is a Yes-instance of p-BM-SMC if and only if $(\mathcal{I}', k')$ is a Yes-instance of p-BR-HRLQ. The parameter $k' = +\sum_i l_i + |H^*| \leq k + |W^*| + |W^*|$. With this we conclude the proof of the theorem. □

In this chapter we have given an XP-Algorithm for HR-LQ parameterized by the number of blocking residents. We have proven in theorem 14 that HR-LQ parameterized by the number of blocking residents is W[1] hard. We thus do not expect an FPT algorithm for the problem (unless FPT=W[1]). Also we have shown some hardness results in theorems 13 and 14.

Chapter 5

Results and Conclusions

This dissertation looks into two different NP-Hard variants of the classical Stable Marriage Problem in the realm of parameterized complexity. In chapter 3 we designed polynomial kernels and parameterized algorithms for Max-SMTI and Min-SMTI parameterized by the solution size; and for Max-SRTI parameterized by the size of a maximum matching. We also studied these problems when the input is restricted to planar graphs. All our algorithms on general graphs run in time $2^{\mathcal{O}(klogk)} + n^{\mathcal{O}(1)}$. For these problems, it would be interesting to either design an algorithm with running time $2^{o(k)}n^{\mathcal{O}(1)}$, or show an appropriate lower bound under complexity theoretic assumptions.

In chapter 4, we gave an XP Algorithm for the Hospital/Residents Problem with Lower quotas parameterized by the number of blocking residents. We also showed that the parameterized problem of finding a feasible assignment with exactly k blocking residents is W[1] hard and we thus do not expect an FPT algorithm for the same (unless FPT=W[1]). It would be interesting to show that the problem of minimizing blocking residents (not exact blocking residents) is W[1] hard. We also showed that the Hospital/Residents Problem with lower quotas, when trying to minimize the blocking edockingges is W[1] hard parameterized by $b + \sum_i l_i + |H^*|$, where b is the number of blocking edges, $\sum_i l_i$ is the sum of lower quotas over all hospitals and $|H^*|$ represents the number of hospitals with non zero lower quota. Also we have a similar hardness result when we look at the second optimization version of the HR-LQ problem where we minimize the number of blocking residents. It will be interesting to see what happens when the parameter is the total number of Hospitals or try to

design an XP-Algorithm with parameter $|H^*|$. Some other parameters of interest could be $|H|\ell$ where $\ell = max_i l_i$.

Bibliography

[1] D. Gale and L. S. Shapley. College admissions and the stability of marriage. The American Mathematical Monthly, 69(1):915, 1962.

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

[3] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dniel Marx, Marcin Pilipczuk, Micha Pilipczuk, Saket Saurabh. Parameterized Algorithms. Springer, ISBN 978-3-319-21274-6

[4] Rod Downey. A Basic Parameterized Complexity Primer.

[5] Kazuo Iwama, David Manlove, Shuichi Miyazaki and Yasufumi Morita. Stable Marriage with Incomplete Lists and Ties. 36th ICALP, pp. 443452 (1999)

[6] Irving, R.W., Manlove, D.F. and Scott, S. (2003). Strong stability in the Hospitals/Residents problem, in Proceedings of STACS 03: the 20th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, Vol. 2607 (Springer), pp. 439450.

[7] Irving, R.W., Manlove, D.F. and Scott, S. (2000). The Hospitals/Residents problem with Ties, in Proceedings of SWAT 00: the 7th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science, Vol.1851 (Springer), pp. 259271.

[8] Koki Hamada,Kazuo Iwama,Shuichi Miyazaki. The Hospitals/Residents Problem with Lower Quotas. Algorithmica (2016) 74:440465

[9] Robert W. Irving. An Efficient Algorithm for the Stable Roommates Problem. JOURNAL OF ALGORITHMS 6, 577-595 (1985)

[10] Eytan Ronn. NP-Complete Stable Matching Problems. JOURNAL OF ALGORITHMS 11,285-304 (1990)

[11] Deeksha Adil, Sushmita Gupta, Sanjukta Roy, Saket Saurabh, Meirav Zehavi. Parameterized Algorithms for the Stable Matching Problem with Ties and Incomplete Lists.

[12] Deeksha Adil, Sushmita Gupta, Saket Saurabh, Meirav Zehavi. Parameterized Algorithms for the Hospitals/Residents Problem with Lower Quotas.

[13] Bart M. P. Jansen. Polynomial Kernels for Hard Problems on Disk Graphs. Algorithm Theory - SWAT 2010, 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings. Pages 310–321, Lecture Notes in Computer Science, 6139, Springer 2010

[14] Matthias Mnich, Ildiko Schlotter. Stable Marriage with Covering Constraints: A Complete Computational Trichotomy. arXiv 1602.08230v1

[15] J. D. Horton and K. Kilakos. Minimum edge dominating sets. SIAM J. of Discrete Mathematics, 6(3):375387, 1993.

[16] Robert W. Irving, David Manlove and Gregg OMalley. Stable marriage with ties and bounded length preference lists. J. Discrete Algorithms, 7(2):213219, 2009.

[17] R. Irving, K. Iwama, D. F. Manlove, S. Miyazaki and Y. Morita. Hard variants of stable marriage. Theoretical Computer Science, 276(1-2):261279, 2002.

[18] R. W. Irving. Stable marriage and indifference. Discrete Applied Mathematics, 48(3):261 272, 1994.

[19] R. Diestel. Graph Theory, 4th Edition, volume 173 of Graduate texts in mathematics. Springer, 2012.