The Dissertation Committee for Changyong Hu
certifies that this is the approved version of the following dissertation:

Theory and Algorithms for Matching Problems Under Preferences

Committee:

Vijay K. Garg, Supervisor

Anna Gal

Christine Julien

Sarfraz Khurshid

David Soloveichik

Theory and Algorithms for Matching Problems Under Preferences

by

Changyong Hu

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2021

Dedicated to my parents Ci'an Hu and Guilan Peng

Acknowledgments

I would like to express my deep appreciation to my advisor, Dr. Vijay Garg, for his patient guidance and support during this long and rewarding journey. He was always willing to discuss with me on my immature ideas and supported me to explore what I truly want to do.

I would like to thank Dr. Anna Gal, Dr. Sarfraz Khurshid, Dr. Christine Julien, and Dr. David Soloveichik for serving as my committee members. In particular, I would like to thank Dr. Anna Gal for helpful discussions on my research and life.

I would like to thank many friends who helped me during my difficult time. In particular, I would like to thank my lab mates and as well my friends, Chenguang Liu, Mengshi Zhang, Zhicheng Xiao, Xiong Zheng, and Jie Hua for their advice and friendship.

I am deeply thankful to my parents, Ci'an Hu and Guilan Peng, for their unconditional love and constant support.

Theory and Algorithms for Matching Problems Under Preferences

Publication No. _____

Changyong Hu, Ph.D.
The University of Texas at Austin, 2021

Supervisor: Vijay K. Garg

Matching under preferences involves matching agents to one another, subject to various optimality criteria such as stability, popularity, and Pareto-optimality, etc. Each agent expresses ordinal preferences over a subset of the others. Real-life applications include assigning graduating medical students to hospitals, high school students to colleges, public houses to applicants, and so on. We consider various matching problems with preferences. In this dissertation, we present efficient algorithms to solve them, prove hardness results, and develop linear programming theory around them.

In the first part of this dissertation, we present two characterizations for the set of super-stable matchings. Super-stability is one of the optimality criteria when the preference lists contain ties. The first algorithm computes irreducible super-stable matchings in the super-stable matching lattice. The second algorithm takes $O(mn)$ time, where m denotes the number of edges

and n denotes the number of vertices and gives an explicit rotation poset that can be used to construct all super-stable matchings.

In the second part, we present a polyhedral characterization of the set of all super-stable matchings, i.e. a linear system that is integral and describes the super-stable matching polytope. We also give alternative proof for the integrality of the strongly stable matching polytope. We also use linear programming techniques to solve an application of the stable matching problem.

In the third part, we present NC algorithms for the popular matching problem. Popularity is another optimality criterion, where each agent gives a vote and the outcome matching has majority votes.

In the last part, we consider envy-freeness, a relaxation of stability in the Hospitals/Residents setting, which allows blocking pairs involving a resident and an empty position of a hospital. Envy-free matching might not exist. We prove NP-hardness results of minimizing envy (if envy is inevitable) in terms of envy-pairs and envy-residents in the Hospitals/Residents Problem with Lower Quota.

Table of Contents

List of Figures

Chapter 1

Introduction

Matching problems have been studied historically for a long time and are still very active research topics in computer science, discrete mathematics, and economics [68, 82, 20]. Besides their importance in theory, matching problems have many real-world applications [66, 2, 1, 4]. We focus on matching problems under preferences, i.e. each participant expresses her preferences over others as an ordered list, possibly with ties. Our task is to find a matching that is optimal with respect to these preferences. The notion of optimality can differ, such as stability [20], popularity [22, 6], and Pareto-optimality [5], etc.

In 1962, Gale and Shapley [20] published their seminal paper entitled "College Admissions and the Stability of Marriage", in which they introduced the stable marriage problem, and the college admission problem, which is also referred as to the hospitals/residents problem. A matching is stable when there is no two people of opposite sex who both prefer each other to their current partner under the matching. Their fundamental result is that every instance of the stable marriage problem admits at least one stable matching and can be computed by a linear-time algorithm known as the deferred acceptance (DA) algorithm.

In this dissertation, we discuss various problems related to stable matchings. Around these problems, we present efficient algorithms, develop linear programming theory, and prove hardness results. In this chapter, we first provide some basic notions in stable matchings and then give an overview of the rest of this dissertation.

1.1 The Stable Marriage Problem

In the classical *stable marriage problem*, a complete bipartite graph $G = (A \cup B, E)$ is given, where A represents a set of n men and B represents a set of n women. Vertices in G are often referred to as agents. For each agent in G, there is a strictly ordered preference list over all agents in the opposite side. The set of these preference lists is denoted by $\mathcal{O}$. In the *stable matching problem*, which is a natural generalization of the *stable marriage problem*, the bipartite graph G is not necessarily a complete bipartite graph and the size of A may not be equal to the size of B. Man m and woman w are connected by an edge $(m, w) \in E$ if they find each other mutually acceptable. For each agent in G, its preference list only includes its neighbors in G. Unless otherwise stated, we consider the generalized *stable matching problem* in this dissertation.

We introduce one useful notation with respect to the set of men, which can be applied to the set of women analogously. A man $m \in A$ prefers woman w_1 to woman w_2 if w_1 precedes w_2 in m's preference list and we denote it by $w_1 \succ_m w_2$.

A set of edges $M \subseteq E$ is a *matching* in G if no two edges in M are

adjacent in G. The vertex that is matched to u in M is denoted by $M(u)$ or $p_M(u)$. If man m and woman w are matched in M, then m and w are called partners in M, written as $m = M(w)$ and $w = M(m)$ or $m = p_M(w)$ and $w = p_M(m)$.

A pair $(m, w) \in E \backslash M$ is called a *blocking pair* for M, if m prefers w to $M(m)$ and w prefers m to $M(w)$, i.e. $w \succ_m M(m)$ and $m \succ_w M(w)$. In the case that some agent u is unmatched in M, we write $M(u) = \emptyset$. Any agent prefers being matched instead of staying alone.

Definition 1.1.1. A matching M is *stable* if there is no blocking pair for M.

1.1.1 The Deferred Acceptance Algorithm

Gale and Shapley's [20] fundamental result is that every instance of the stable marriage problem admits at least one stable matching. They proved this result by giving a linear-time algorithm (the deferred acceptance algorithm or DA algorithm for short) that is guaranteed to find such a matching. Since then, many variants of DA algorithms have been developed to solve different variants of the stable matching problem. In later chapters, we give some variants of DA algorithms as well.

The DA algorithm of Gale and Shapley is very simple and natural and can be outlined as follows. Initially, no men and women are matched. In turn, each unmatched man proposes to his most preferred woman who he has not yet proposed in his preference list. If a woman is unmatched and receives a proposal, then she accepts the proposal. If a woman is matched and receives

a proposal better than her current fiance, she rejects her fiance and accepts the new proposal. If she prefers her fiance over the new proposal, she rejects the new proposal. A rejected man continue proposing until he runs out of his preference list. The algorithm ends once every man is matched or has proposed to all women in his preference list.

Theorem 1.1.1 ([20]). *Given an instance $I = (G = (A \cup B, E), \mathcal{O})$ of the stable matching problem, the DA algorithm outputs a stable matching M in $O(|E|)$ time and M assigns each man his best partner among all stable matchings in I.*

1.1.2 The Lattice of Stable Matchings

An instance of the stable matching problem can have exponentially many stable matchings. The set of all stable matchings admits a rich structure.

Theorem 1.1.2 (Knuth [44] attributes this result to John Conway). *The set of stable matchings in a given instance forms a distributive lattice.*

A *distributive lattice* is a partial order in which

(i) each pair of elements a, b has a greatest lower bound, or meet, denoted by $a \wedge b$, so that $a \wedge b \preceq a$, $a \wedge b \preceq b$, and there is no element c such that $c \preceq a$, $c \preceq b$ and $a \wedge b \prec c$;

(ii) each pair of elements a, b has a least upper bound, or join, denoted by $a \vee b$, so that $a \preceq a \vee b$, $b \preceq a \vee b$, and there is no element c such that $a \preceq c$, $b \preceq c$ and $c \prec a \wedge b$;

(iii) the *distributive laws* hold, $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ and $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

The join and meet operators of the lattice of stable matchings can be defined as below.

Definition 1.1.2. Given two stable matchings M_1 and M_2, the join ($\vee$) of M_1 and M_2, written as $M_1 \vee M_2$, is defined as for each men m, match him to the worst woman in $M_1(m)$ and $M_2(m)$. The meet ($\wedge$) of M_1 and M_2, written as $M_1 \wedge M_2$, is defined similarly.

Unsurprisingly, both $M_1 \vee M_2$ and $M_1 \wedge M_2$ are stable matchings. Additionally, $M_1 \vee (M_2 \wedge M_3) = (M_1 \vee M_2) \wedge (M_1 \vee M_3)$.

The two extreme points of this lattice are called *man-* and *woman-optimal stable matchings*. The man-proposing DA algorithm outputs the man-optimal stable matching. The man-optimal stable matching is also the woman-pessimal stable matching: it assigns to each woman the worst partner among all feasible stable matchings. The woman-optimal stable matching can be obtained by running the DA algorithm reversely, with women proposing instead of men.

More details of the lattice structure of stable matchings and its relation to Birkhoff's representation theorem [8] will be discussed in later chapters.

1.2 Extensions of The Stable Marriage Problem

1.2.1 The Hospitals/Residents Problem

The Hospitals/Residents (HR, for short) problem, or sometimes referred to as the College Admission problem, is a many-to-one extension of the stable matching problem. The problem involves a set of residents R and a set of hospitals H. The bipartite graph $G = (R \cup H, E)$ is defined similarly as in the stable matching problem. Besides, each hospital $h \in H$ has an upper quota u_h, indicating the maximum capacity of this hospital. Each resident and each hospital has a strict order preference list over the opposite side as well.

An assignment M is a subset of E. If $(r, h) \in M$, we say r is assigned to h in M and h is assigned to r in M. The vertices that are assigned to u is denoted by $M(u)$. If a resident $r \in R$ and $M(r) = \emptyset$, we say r is unassigned, otherwise r is assigned. Similarly, a hospital $h \in H$ is under-subscribed, full or over-subscribed according as $M(h)$ is less than, equal to or greater than u_h, respectively.

A many-to-one matching (or simply a matching) M is an assignment such that each resident is assigned to at most one hospital and no hospital is over-subscribed, i.e. $|M(r)| \leq 1$ for each resident r and $|M(h)| \leq u_h$ for each hospital h. The stability definition needs to be revised for the Hospitals/Residents problem.

Definition 1.2.1. Given a matching M of an instance I of HR. A pair $(r, h) \in E \backslash M$ is a blocking pair for M, if

(i) r is unassigned or prefers h to $M(r)$;

(ii) h is under-subscribed or prefers r to at least one member of $M(h)$.

M is stable if it admits no blocking pair.

The DA algorithm can be easily extended to solve the Hospitals/Residents problem and the resident-optimal and hospital-optimal stable matchings can be computed. There may be other stable matchings as well. A famous theorem called Rural Hospitals Theorem describes some key structural properties held regarding unassigned residents and under-subscribed hospitals with respect to all stable matchings in a given HR instance, as follows:

Theorem 1.2.1 (Rural Hospitals Theorem [21, 66, 67])**.** *For all stable matchings of a given HR instance, the following properties hold:*

(i) the same residents are assigned;

(ii) each hospital is assigned the same number of residents;

(iii) any hospital that is under-subscribed in one stable matching is assigned exactly the same set of residents.

Due to this theorem, we are not able to accommodate more residents to those "rural" hospitals while still maintaining stability.

1.2.2 Ties in Preference Lists

Another natural extension is the stable matching problem with ties. In large-scale matching schemes, participants may not able to provide a genuine strict preference list over a large number of participant from the other party and may prefer to express ties or indifference in their preference lists. When ties are present, the definition of stability needs to be revised. Irving and Manlove [33, 51] define three levels of stability, called *weak stability*, *strong stability* and *super-stability*.

Before we give the definition of each stability, we introduce some notations in term of ties. A man m is indifferent between w_1 and w_2 if w_1 and w_2 appear in the same tie of m's preference list. We denote this indifference by $w_1 =_m w_2$. We call that m *weakly prefers* w_1 to w_2 if $w_1 \succ_m w_2$ or $w_1 =_m w_2$ and denote it by $w_1 \succeq_m w_2$. The definition of blocking pairs of each stability is defined as below.

Definition 1.2.2. Given a matching M of an instance of the stable matching problem with ties, a pair $(m, w) \in E \backslash M$ is a blocking pair of M if

- *weak stability*:
  $w \succ_m M(m)$ and $m \succ_w M(w)$;
- *strong stability*:
  (i) $w \succ_m M(m)$ and $m \succeq_w M(w)$, or
  (ii) $w \succeq_m M(m)$ and $m \succ_w M(w)$;

- *super-stability*:

  $w \succeq_m M(m)$ and $m \succeq_w M(w)$.

1.3 Stable Matchings and Linear Programming

Linear programming is a powerful tool in solving many combinatorial optimization problems (see more details in books by Schrijver [71, 72]). The first linear programming (LP) formulation for stable matchings was given by Vande Vate [78], where the underlying graph is a complete bipartite graph. Rothblum [70] extended Vande Vate's work to arbitrary bipartite graph by giving a LP formulation for the general stable matching problem. Later, Roth et al [69] used duality theory to obtain new results and to derive new proofs of known results for the stable matching problem. They also defined a *fractional stable matching* to be a solution to the linear inequalities system (not necessarily an extreme point) and showed that the set of fractional stable matchings forms a distributive lattice. The following theorem summarizes Vande Vate and Rothblum's work.

Theorem 1.3.1 ([78, 70]). *Let $(G = (V, E), \mathcal{O})$ be an instance of the stable matching problem where the graph G is bipartite. Then, the stable matching polytope $SM(G)$ is described by the following linear system:*

$$\sum_{u \in N(v)} x_{u,v} \leq 1, \qquad \forall v \in V, \tag{1.1a}$$

$$\sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + x_{u,v} \geq 1, \qquad \forall (u,v) \in E, \tag{1.1b}$$

$$x_{u,v} \geq 0, \qquad \forall (u,v) \in E \tag{1.1c}$$

where $N(v)$ denotes the set of neighbors of v in G, and $w >_u v$ means u prefers w to v.

Theorem 1.3.1 implies polynomial-time algorithms to find egalitarian, minimum regret and minimum weight stable matchings [35, 15, 9]. Kiraly and Pap [42] further showed that the LP formulation by Rothblum is totally dual integral (TDI) and gave a strongly polynomial-time algorithm for finding an integer optimal dual solution.

Recently, Kunysz [46] gave a LP formulation for the strongly stable matching problem and proved that the strongly stable matching polytope given by their linear inequalities is integral. In this dissertation, we give a LP formulation for the super-stable matching problem.

1.4 The Housing Allocation Problem

The Housing Allocation (HA, for short) problem is another two-sided matching problem involving preference lists, where only one side expresses preference. The HA problem is often described as allocating a set H of indivisible goods, e.g. houses, jobs, among a set A of applicants [74, 32, 13].

Each applicant a has an ordinal preference list over a subset of H. In the literature the situation where each applicant initially owns a good is known as the Housing Market [74, 68, 65] problem. When there is no prior assignment, we obtain the Housing Allocation [32, 82, 3] problem.

There are a range of optimality criteria that can be applied to instances of the Housing Allocation problem, such as Pareto optimality, popularity and profile-based optimality. Many-to-one extension and ties can also be applied to the Housing Allocation problem.

We discuss some results in the Housing Allocation problem in Chapter 4.

1.5 Overview

In this section, we give a summary of each chapter of the rest of this dissertation.

In Chapter 2, we present two algorithms that give two compact representations of size $O(m)$ that can be used to construct all super-stable matchings, where m denotes the number of edges in the graph. The first algorithm computes irreducible super-stable matchings in the super-stable matching lattice. The second algorithm takes $O(mn)$ time, where n denotes the number of vertices in the graph, and gives an explicit rotation poset similar to the rotation poset in the classical stable marriage problem.

In Chapter 3, we use linear programming approach to solve two prob-

lems related to the stable matching problem. We give a polyhedral characterisation of the set of all super-stable matchings and prove that the super-stable matching polytope is integral, thus solving an open problem stated in the book by Gusfield and Irving [25]. We also give an alternative proof for the integrality of the strongly stable matching polytope. Another problem we discuss in this chapter is to compute a stable matching closest to a given initial matching or assignment under an appropriate distance function between matchings.

In Chapter 4, we consider the popular matching problem. The popular matching problem is of matching a set of applicants to a set of posts, where each applicant has a preference list, ranking a non-empty subset of posts in the order of preference, possibly with ties. A matching M is popular if there is no other matching M' such that more applicants prefer M' to M. We give the first NC algorithm to solve the popular matching problem without ties. We also give an NC algorithm that solves the maximum-cardinality popular matching problem. No NC or RNC algorithms were known for the matching problem in preference systems prior to this work. Moreover, we give an NC algorithm for a weaker version of the stable matching problem, that is, the problem of finding the "next" stable matching given a stable matching.

In Chapter 5, we consider the Hospitals/Residents problem with Lower Quota (HRLQ for short), where each hospital has a lower quota for the number of residents it receives. In this setting, a stable matching may not exist. Envy-freeness is introduced as a relaxation of stability that allows blocking pairs involving a resident and an empty position of a hospital. Hamada et al.

[27] considered the problem of minimizing the number of blocking pairs and showed hardness of approximation results. We present NP hardness results of minimizing envy in terms of envy-pairs and envy-residents in the Hospitals/Resident problem with Lower Quota. We also give a simple exponential-time algorithm for the Minimum-Envy-Pair HRLQ problem.

Chapter 2

Characterization of Super-stable Matchings

2.1 Introduction

The problem of characterizing super-stable matchings was stated in the book by Gusfield and Irving [25] as one of the 12 open problems. The structure of the set of all stable matchings in the stable marriage problem without ties is well understood in Gusfield and Irving's book [25]. Recently, Kunysz et al. [46] gave compact representations for the set of all strongly stable matchings and showed that the construction can be done in $O(mn)$ time, where n and m denote the number of vertices and edges in the graph. Scott [73] investigated the structure of all super-stable matchings by defining an object called meta-rotation, which corresponds to one collection of rotations in some arbitrary tie-breaking instance of the original instance and the time complexity of the construction is $O(m^2)$.

We give two compact representations of the set of all super-stable matchings that can be constructed in, respectively, $O(nm^2)$ and $O(mn)$ time.

The first representation of the set of all super-stable matchings consists of $O(m)$ matchings, each of which is a man-optimal stable matching among all super-stable matchings that contains a given edge. We show that computing

such matching for each edge can be reduced to computing a man-optimal super-stable matching in a reduced graph by deleting an appropriate subset of edges in graph G. The algorithm is described in Section 2.3.

Our second representation explicitly constructs rotations, which are differences between consecutive super-stable matchings in a maximal sequence of super-stable matchings starting with a man-optimal super-stable matching and ending with a woman-optimal super-stable matching. Unlike Scott's [73] meta-rotation, our rotation is the symmetric difference of two super-stable matchings, which could be a cycle or multiple cycles.

Our construction takes $O(mn)$ time, while Scott's [73] algorithm takes $O(m^2)$ time. We also show how to efficiently construct a partial order among rotations. This poset can be used to solve other problems connected to super-stable matchings such as the enumeration of all super-stable matchings and the maximum weight super-stable matching problem. Fleiner et al. [18] solve the weight super-stable matching by reducing it to the 2-SAT problem and the time complexity is $O(mn \log(W))$, where W is the maximum weight among all edges in G. By using the rotation poset constructed in this paper, the weighted problem can also be solved in $O(mn \log(W))$ time.

The results presented in this chapter are joint work with Vijay Garg and published in [29].

2.1.1 Related Work

Irving [33] gave an $O(m)$ algorithm to find a super-stable matching if it exists. Spieker [75] showed that super-stable matchings form a distributive lattice. Further properties of super-stable matchings were proved by Manlove in [52]. Scott [73] introduced the concept called *meta-rotation poset* for super-stable matchings and showed the one-to-one correspondence between super-stable matchings and closed subsets of the poset.

Irving [33] and Manlove [52] gave an $O(m^2)$ algorithm to find a strongly stable matching if it exists. Kavitha et al. [40] gave an $O(nm)$ algorithm for the strongly stable matching problem. Manlove [52] showed that strongly stable matchings form a distributive lattice. Kunysz et al. [47] gave a characterisation of all strongly stable matchings and later Kunysz [46] gave a polyhedral description for the set of all strongly stable matchings and proved that the strongly stable matching polytope is integral.

For weakly stable matchings, it is not true that all weakly stable matchings of a given instance always have the same size. Weakly stable matching can be easily found by running the deferred-acceptance algorithm while breaking ties in an arbitrary manner. The problem of computing a maximum-size weakly stable matching is NP-hard, which has been proved by Iwama et al. [36]. Thus finding good approximations of the problem becomes very interesting. For the version when ties are allowed on both sides, the currently best approximation factor is 3/2 [55, 61, 43]. For the case when ties only occur on one side, there are a sequence of works pushing the approximation factor

lower. Iwama et al. [37] gave an 25/17 approximation algorithm. Huang and Kavitha [31] improved it to 22/15. Later Radnai [64] improved the approximation factor to 41/28, then Dean et al. [12] pushed the approximation factor to 19/13. Most recent result by Lam and Plaxton [48] gave the currently best approximation factor of $1 + 1/e$.

2.2 Preliminaries

In this section we give some definitions and theorems that are useful in the following sections.

Theorem 2.2.1. *[33, 52] There is an $O(m)$ algorithm to determine a man-optimal super-stable matching of the given instance or report that no such matching exists.*

Theorem 2.2.2. *[52] In a given instance of the super-stable matching problem, the same set of vertices are matched in all super-stable matchings.*

Lemma 2.2.3. *[52] Let M, N be two super-stable matchings in a given super-stable matching instance. Suppose that, for any agent p, $(p, q) \in M$ and $(p, q') \in N$, where p is indifferent between q and q', then $q = q'$.*

We recall some standard notations and definitions from the theory of matchings under preferences. For a given edge (m, w), any matching containing (m, w) is called an (m, w)-matching. Let us denote the set of all super-stable matchings of G by $\mathcal{M}_G$. Let $\mathcal{M}_G(m, w)$ be the set of all super-stable (m, w)-matchings in G.

For two super-stable matchings M and N, we say that M *dominates* N and write $M \succeq N$ if each man m weakly prefers $M(m)$ to $N(m)$. If M dominates N and there exists a man m who prefers $M(m)$ to $N(m)$, then we say M *strictly dominates* N, write $M \succ N$ and we call N a *successor* of M. Note that by Lemma 2.2.3, $M \succeq N$ implies $M \succ N$, assuming M is not equal to N.

2.3 Irreducible Super-stable Matchings

In this section, we give our first representation via irreducible matchings. Birkhoff's representation theorem [8] for distributive lattices states that the elements of any finite distributive lattice can be represented as finite sets in such a way that the lattice operations correspond to unions and intersections of sets. The theorem gives a one-to-one correspondence between distributive lattices and partial orders. Our goal is to find the partial order that represents the set of all super-stable matchings.

Distributive lattice is closely related to rings of sets, which is a family of sets that is closed under set unions and set intersections. If the sets in a ring of sets are ordered by set inclusion, they form a distributive lattice. Theory regarding rings of sets and its application to representations of the set of stable matchings in the classical stable marriage problem is well studied by Irving and Gusfield [25]. Below we give a brief summary of this theory that serves as a preliminary for our algorithm.

Given a finite set B, the *base* set, a family $\mathcal{F} = \{F_0, F_1, \cdots, F_k\}$ of

subsets of B is called a ring of sets over B if $\mathcal{F}$ is closed under set union and intersection. A ring of sets contains a unique minimal element and a unique maximal element.

For any element $a \in B$, we denote $\mathcal{F}(a)$ the set of all elements of $\mathcal{F}$ that contains a. It is obvious that $\mathcal{F}(a)$ is also a ring of sets over B. We define $F(a)$ to be the unique minimal element of $\mathcal{F}(a)$. An element $F \in \mathcal{F}$ that is $F(a)$ for some $a \in B$ is called *irreducible*. We denote $I(\mathcal{F})$ the set of all irreducible elements of $\mathcal{F}$. We view $(I(\mathcal{F}), \leq)$ as a partial order under the relation $\leq$ of set containment. We give the Birkhoff's representation theorem in the language of rings of sets below.

Theorem 2.3.1. *[25] i) There is a one-to-one correspondence between the closed subsets of $I(\mathcal{F})$ and the elements of $\mathcal{F}$.*
ii) If S and S' are closed subsets of $I(\mathcal{F})$ that generate $F = \bigcup S$ and $F' = \bigcup S'$ respectively, then $F \subseteq F'$ if and only if $S \subseteq S'$.

In the context of super-stable matchings, the base set B corresponds to the set of all acceptable pairs $(m, w) \in E$. We define the P-set of a super-stable matching M to be the set of all pairs (m, w), where w is either $M(m)$ or a woman whom m weakly prefers to $M(m)$, which corresponds to an element in $\mathcal{F}$. It is obvious that the unique minimal (man-optimal) super-stable matching in $\mathcal{M}_G(m, w)$, if nonempty, is *irreducible*.

We describe an $O(|E|)$ algorithm for computing a man-optimal super-stable (m, w)-matching in G. Algorithm 1 essentially constructs a reduced

graph $G' \subseteq G$ by removing some edges from G (line 4 to line 13 in Algorithm 1). After that, the algorithm computes a man-optimal super-stable matching M' in the reduced graph G'. By adding back the edge (m, w), the new matching $M \cup (m, w)$ is super-stable in G.

Figure 2.1 gives an example on how edges are removed in line 4-8. m and w are removed from G. Any edges (m', w') that satisfy $m \preceq_w m'$ and $w \succeq_{m'} w'$ are removed from G. Note that if any such edge (m', w') is matched in a super-stable matching M in the reduced graph G', then the matching $M \cup \{(m, w)\}$ is blocked by (m', w') in G.

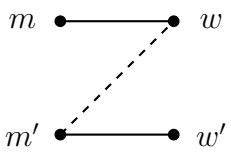


Figure 2.1: $m \preceq_w m'$ and $w \succeq_{m'} w'$

Lemma 2.3.2. *Let M be a super-stable (m, w)-matching. Then $M' = M \backslash \{(m, w)\}$ is a super-stable matching in the reduced graph G'.*

Proof. We need to prove $M' \subseteq G'$ or equivalently none of edges removed from G is matched in M'. Suppose not, an edge (m', w') was removed from G and is matched in M'. Note that $m' \neq m$ and $w' \neq w$. Hence, it follows that there is an edge (m, w') or (m', w) which caused the removal of (m', w'). W.l.o.g, let's assume it is (m, w') which caused the removal of (m', w'). Then we have $w \preceq_m w'$ and $m \succeq_{w'} m'$. Obviously, (m, w') is a blocking pair, which leads to a contradiction of M being super-stable.

Algorithm 1: Computing man-optimal super-stable (m, w)-matching

Input: the graph $G = (A \cup B, E)$ and preference lists of G and an edge $(m, w) \in E$.
Output: man-optimal super-stable (m, w)-matching or deciding that no such matching exists.
$G' \leftarrow G \backslash \{m, w\}$ // remove m and w and all edges that are incident to them
for m' s.t. $(m', w) \in E$ and $m \preceq_w m'$ **do**
    for w' s.t. $(m', w') \in E$ and $w \succeq_{m'} w'$ **do**
        $G' \leftarrow G' \backslash (m', w')$
    end for
end for
for w' s.t. $(m, w') \in E$ and $w \preceq_m w'$ **do**
    for m' s.t. $(m', w') \in E$ and $m \succeq_{w'} m'$ **do**
        $G' \leftarrow G' \backslash (m', w')$
    end for
end for
compute man-optimal super-stable matching in G'.
if exists man-optimal super-stable matching M in G' and $M \cup (m, w)$ is super-stable in G
    return $M \cup (m, w)$
else
    return no super-stable (m, w)-matching exists.
end if

To prove super-stability of M' is easy. If there were an edge e blocking M', it would also block M. □

Lemma 2.3.3. *Let M' be some super-stable matching in the reduced graph G' if exists. If $M' \cup (m, w)$ is a super-stable matching in G, then for each super-stable matching N' in G', $N' \cup (m, w)$ is a super-stable matching in G. If G' does not have any super-stable matching, then there is no super-stable (m, w)-matching.*

Proof. Let $M = M' \cup (m, w)$. Since M' is super-stable in G'. It follows that only the removed edges in $E \backslash E'$ can potentially block M. We have two cases shown as below:

Case 1: Any edge that is incident to m or w cannot block M. W.l.o.g, Suppose that for some w' that is incident to m, and (m, w') blocks M. Then we have $w' \succeq_m w$. By the construction of G', any edge (m', w') such that $m \succeq_{w'} m'$ was removed. Hence w' must be unmatched in M. From Theorem 2.2.2, w' is unmatched in any super-stable matching of G. Let us assume there exists some super-stable (m, w)-matching N. Then $N' = N \backslash (m, w)$ is super-stable in G'. Since w' is unmatched in N, (m, w') blocks N, contradiction.

Case 2: Any edge (m', w') such that $m' \neq m$ and $w' \neq w$ cannot block M. By the construction of the reduced graph G', the removal of (m', w') was caused by some edge (m, w') or (m', w). W.l.o.g, some edge (m, w') caused the removal of (m', w'). Hence, if w' is matched in M, then $M(w') \succeq_{w'} m'$. (m', w') does not block M. In the case that w' is unmatched in M, w' is unmatched in

any super-stable matching in G. Similar to Case 1, if there exists some super-stable (m, w)-matching N, then (m, w') blocks N, contradiction. By the same argument, if M is super-stable in G, for any other super-stable matching N' in G', M' and N' match the same set of vertices. No edges in $E \backslash E'$ can block $N' \cup (m, w)$. □

Theorem 2.3.4. *Let (m, w) be an edge in G. There is an $O(m)$ algorithm for computing a man-optimal super-stable (m, w)-matching or deciding that no super-stable (m, w)-matching exists.*

Proof. Lemma 2.3.3 makes sure if Algorithm 1 outputs a matching M, then M is super-stable in G. Lemma 2.3.2 guarantees that if there exists any super-stable matching in G, then Algorithm 1 would never miss it. □

Theorem 2.3.5. *$(I(\mathcal{M}_G), \leq)$ can be constructed in $O(nm^2)$ time.*

Proof. $I(\mathcal{M}_G)$ can be computed in $O(m^2)$ time by running Algorithm 1 for each edge $(m, w) \in E$. The set $I(\mathcal{M}_G)$ has at most m elements. By checking each pair of $I(\mathcal{M}_G)$, we can construct the partial order. Each check takes $O(n)$ time. Thus, the total time is $O(nm^2)$. □

2.4 A Maximal Sequence of Super-stable Matchings

Representation via irreducible matchings is intuitive, but the time complexity is high. In this section, we give another representation via rotation poset and the time complexity to construct this rotation poset is only $O(mn)$.

Rotation poset derives from the concept of *minimal differences* of a ring of sets. A chain $C = \{C_1, \cdots, C_q\}$ in $\mathcal{F}$ is an ordered set of elements of $\mathcal{F}$ such that C_i is an immediate predecessor of C_{i+1} for each $i \in [q]$. The maximal chain is a chain that begins at the minimal element of $\mathcal{F}$, F_0 and ends at the maximal element of $\mathcal{F}$, F_z. Let F_i and F_{i+1} be two elements of $\mathcal{F}$ such that F_i is an immediate predecessor of F_{i+1}. The difference $D = F_{i+1} \backslash F_i$ is called a *minimal difference* of $\mathcal{F}$. Note that for each two consecutive elements of a chain C, there is a minimal difference D. We say that C contains D. The following two theorems give another version of Birkhoff's representation theorem in the language of minimal differences. The reader can find more details in Irving and Gusfield's book [25].

Theorem 2.4.1. *[25] If F and F' are two elements in $\mathcal{F}$ such that $F \subset F'$, then every chain from F to F' in $\mathcal{F}$ contains exactly the same set of minimal differences (in a different order).*

Theorem 2.4.2. *[25] Let $D(\mathcal{F})$ denote the set of all minimal differences in $\mathcal{F}$. For two minimal differences D and D', $D \prec D'$ if and only if D appears before D' on every maximal chain in $\mathcal{F}$. There is a one-to-one correspondence between the elements of $\mathcal{F}$ and the closed subsets of $D(\mathcal{F})$.*

In the context of super-stable matchings, we want to compute a maximal sequence of super-stable matchings in $\mathcal{M}(G)$, i.e. a sequence $M_0 \succ M_1 \succ \cdots \succ M_z$ where M_0 is the man-optimal super-stable matching and M_z is the woman-optimal super-stable matching and for each $1 \leq i \leq z$, there is no

super-stable matching M' such that $M_{i-1} \succ M' \succ M_i$. We call a matching M' a strict successor of a matching M if M' is a successor of M, i.e. $M \succ M'$ and there exists no super-stable matching M'' such that $M \succ M'' \succ M'$. We can solve this problem by computing a strict successor of any super-stable matching M.

Let M be a super-stable matching in G and m a vertex in A. Suppose that there exists a super-stable matching M' such that m gets a worse partner in M' than in M, i.e. $M(m) \succ_m M'(m)$. Let $w = M'(m)$, by Lemma 2.2.3, w must be matched in M and $m \succ_w M(w)$. Hence we are essentially searching for some vertex w such that $M(m) \succ_m w$ and $m \succ_w M(w)$. In Algorithm 2, the set E_c contains, for each man m, highest ranked edges incident to him that satisfy the condition above. For each man m, the candidate edge (m, w) is not unique; there might be other edge (m, w') that forms a tie with (m, w). While in the case of strict preference list, the candidate edge is unique.

A strongly connected component S of a directed graph G is a subgraph S that is strongly connected, i.e. there is a path in S in each direction between each pair of vertices of S, and is maximal with this property: no additional edges or vertices from G can be included in the subgraph without breaking its property of being strongly connected. We say that $e = (m, w)$ is an outgoing edge of S if $m \in S$ and $w \notin S$.

In Algorithm 2 given below we maintain a directed graph $G_d = (V, E_d)$, whose every edge $(m, w) \in E_d \cap M$ is directed from w to m and every other edge (m, w) is directed from m to w. G_d is a subgraph of G that contains

the edges the algorithm traverses so far. The basic idea of this algorithm is that for each man m such that $M(m) \neq M_z(m)$, we traverse the preference list of m until we find some candidate edges defined above. We add the edges traversed into G_d and the candidate edges into G_c. For each strongly connected component S of G_d without outgoing edges, we try to find a perfect matching on S in $G_c = (V, E_c)$. If we are successful, we find a strict successor of M. Otherwise, we modify G_c and G_d by allowing edges of lower ranks.

Figure 2.2 gives an example to show an execution of Algorithm 2. $M_0 = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4)\}$ is the man-optimal super-stable matching. Dashed edges are those candidate edges in E_c. We can see that $outdeg(S(a_3)) = 0$ and $outdeg(S(a_1)) = 1$. The matching $M_1 = \{(a_1, b_1), (a_2, b_2), (a_3, b_4), (a_4, b_3)\}$ is extended because $outdeg(S(a_3)) = 0$. While updating G_c and G_d, edge (a_2, b_3) is deleted and now $outdeg(S(a_1)) = 0$. Thus the matching $M_2 = \{(a_1, b_2), (a_2, b_1), (a_3, b_4), (a_4, b_3)\}$ is extended.

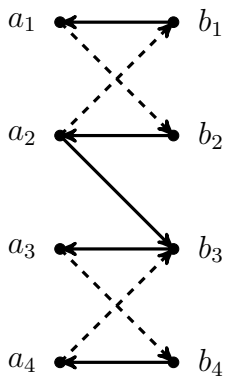


Figure 2.2: Each solid edge (a, b) denotes a rank 1 edge for a and rank 2 edge for b and each dashed edge (a, b) denotes a rank 2 edge for a and rank 1 edge for b

Algorithm 2: Computing a maximal sequence of super-stable matchings

let M_0 be the (unique) man-optimal super-stable matching of G.
let M_z be the (unique) woman-optimal super-stable matching of G.
$M \leftarrow M_0$
let M' contain edge $(m, M(m))$ for each man m s.t. $M(m) =_m M_z(m)$
let E_d contain all edges of M
let G_d be the directed graph (V, E_d) s.t. each edge $(m, w) \in E_d \cap M$ is directed from w to m and every other edge (m, w) is directed from m to w
$E' \leftarrow E \backslash E_d$
let $E_c = M'$ and $G_c = (V, E_c)$
for each $(m, w) \in M$ remove from E' each edge (m', w) s.t. $m' \prec_w m$ and each edge (m, w') s.t. $w' \succeq_m w$
repeat
  while $(\exists m \in A)$ $deg_{G_c}(m) = 0$ and $outdeg(S(m)) = 0$ **do**
    add the set E_m of top choices of m from E' to E_d
    if $outdeg(S(m)) = 0$ **then**
      add every edge $(m, w) \in E_m$ s.t. $m \succ_w M(w)$ and $M(m) \succ_m w$ to E_c
      for each edge (m, w) of E_c that becomes strictly dominated by some added edge (m', w), remove it from G_c
      remove E_m from E'
    end if
  end while
  for each $m \in A$ s.t. $outdeg(S(m)) = 0$ **do**
    delete all lowest ranked edge in $E_c \cup E'$ incident to any $w \in S$ s.t. w is multiple engaged
  end for
  while $(\exists S)$ $outdeg(S) = 0$ and E_c is a perfect matching on S **do**
    $M \leftarrow (E_c \cap S) \cup (M \backslash S)$
    $M_i \leftarrow M$
    output M_i

26 $i \leftarrow i + 1$

27 update G_c and G_d: $E_c \cap S$ contains only edges $(m, M(m))$ s.t. $M(m) =_m M_z(m)$; an edge $(m, w) \in S$ stays in G_d only if $w = M(m)$ and $rank_w(m) \leq rank_M(w)$

28 **end while**

29 **until** $(\forall m \in A)\ rank_M(m) = rank_{M_z}(m)$

2.4.1 Correctness of Algorithm 2

Lemma 2.4.3 proves that any edge removed from G_d (line 9 and line 27) never block any super-stable matching that the algorithm will output.

Lemma 2.4.3. *Let M be a super-stable matching in G. For any successor N of M such that N is also a super-stable matching in G and each $(m, w) \in M$, any edge (m, w') such that $w' \succeq_m w$ or (m', w) such that $m \succ_w m'$ cannot block N.*

Proof. For any edge (m, w') such that $w' \succeq_m w$, this edge (m, w') cannot block M since M is super-stable. Thus we must have $m \prec_{w'} M(w')$. The matching N is a successor of M from the man's point of view, hence from the woman's point of view, M is a successor of N. Then we have $m \prec_{w'} N(w')$ since $M(w') \preceq_{w'} N(w')$, which implies that the edge (m, w') would not block N. Similarly, for any edge (m', w) such that $m \succ_w m'$, we have $N(w) \succ_w m'$ since $N(w) \succeq_w M(w)$, which implies that the edge (m', w) would not block N. □

Lemma 2.4.4. *No edge deleted in line 15 can belong to any super-stable matching N dominated by M.*

Proof. Suppose that the algorithm deletes an edge (m, w) from E_c because it is dominated by some edge (m', w), i.e. w strictly prefer m' to m. We want to show that (m, w) cannot belong to any super-stable matching dominated by M. Suppose, for a contradiction, that the edge (m, w) belongs to a super-stable matching N dominated by M. m' must match to another woman w' in N and m' strictly prefers w' to w, otherwise the edge (m', w) would block N. Thus we have that $rank_M(m') \leq rank_{m'}(w') < rank_{m'}(w)$, where the first inequality comes from the fact that N is dominated by M. Hence, we have two cases shown as below:

Case 1: $rank_M(m') < rank_{m'}(w') < rank_{m'}(w)$. In this case, w' must match to a different man $M(w')$ other than m' in M. Again because M dominates N, $m' \succeq_{w'} M(w')$. If $m' \succ_{w'} M(w')$, by our algorithm, in order to let (m', w') belong to E_d, (m', w') must also belong to E_c and this requires that $rank_{m'}(w) = rank_{m'}(w')$, which contradicts with the fact that is $rank_{m'}(w') < rank_{m'}(w)$.

Case 2: $rank_M(m') = rank_{m'}(w') < rank_{m'}(w)$. First, we rule out the case that m' is indifferent between $M(m')$ and $w' = N(m')$ by Lemma 2.2.3. Thus we must have $w' = M(m') = N(m')$. In order to prove a contradiction, we need the property of strongly connected component. m' and w are in the same strongly connected component, hence there must be a directed path P from w to m'. Arc (w', m') is the unique arc that points to m' since $(m', w') \in M$. Hence, there must be an arc (m'', w') in P. If $rank_M(m'') < rank_N(m'')$, then $rank_{m''}(w') \leq rank_N(m'')$, thus (m'', w') would block N. So $M(m'') = N(m'')$.

Let $w'' = M(m'')$. (m'', w'') is also in path P, Let us continue this process until it reaches to w and we will have $M(w) = N(w)$, which is a contradiction. □

Lemma 2.4.5. *No edge deleted in line 20 can belong to any super-stable matching N dominated by M.*

Proof. Suppose that an edge (m, w) is deleted in line 20, there must be an edge $(m', w) \in E_c$ such that $m =_w m'$. Let N be a super-stable matching dominated by M that includes the edge (m, w). m' must match to a woman w' in N and m' strictly prefer w' to w, otherwise the edge (m', w) would block N. m' and w are in the same strongly connected component. By the same argument as in Lemma 2.4.4, we will have a contradiction. We omit the proof here. □

Lemma 2.4.6. *The output matching M_i is super-stable and a strict successor of M_{i-1}.*

Proof. Note that the algorithm outputs M_i when the edge set E_c is a perfect matching in a strongly connected component S with no outgoing edges and $M_i = (M_{i-1} \backslash S) \cup (E_c \cap S)$. Suppose, for a contradiction, that M_i is blocked by some edge $(m, w) \in E_d$. There are four cases.

Case 1: $m \notin S$ and $w \notin S$, it is obvious that (m, w) cannot block M_i, since it would block M_{i-1} as well.

Case 2: $m \in S$ and $w \notin S$, this is not possible, because this will imply S has an outgoing edge in E_d.

Case 3: $m \notin S$ and $w \in S$, then $M_i(m)(= M_{i-1}(m)) \succ_m w$, hence (m, w) would not block M_i.

Case 4: $m \in S$ and $w \in S$, if (m, w) never belong to E_c, then $M_i(w) \succ_w M_{i-1}(w) =_w m$, (m, w) can not block M_i; if (m, w) once belongs to E_c and got deleted later, then w always get a strictly better partner than m. We prove that no edge from E_d can block M_i. There might be some other edges $e \notin E_d$ that can potentially block M_i. These edges are deleted during the updating of E_d. Lemma 2.4.3 gives a proof that these set of edges cannot block any matching N that is dominated by M_{i-1}. Hence M_i is super-stable.

Next we prove that M_i is a strict successor of M_{i-1}. Suppose not and let m be any man in S and N a successor of M_{i-1} such that $M_{i-1}(m) \succ N(m) \succ M_i(m)$. If $(m, N(m)) \in E_c$ and is not deleted during the algorithm, then $(m, M_i(m))$ would not be in E_c, which is not true. Since N is a successor of M and is super-stable, by Lemma 2.4.4 and Lemma 2.4.5, the edge $(m, N(m))$ can never once belong to E_c. Let $w = N(m)$, by our updating rule of E_d, we have $N(w) \succeq_w M(w)$. While if $N(w) \succ_w M(w)$, then the edge (m, w) must once belong to E_c. Thus we have $N(w) =_w M(w)$, which violates Lemma 2.2.3. □

Lemma 2.4.7. *If $M_i \neq M_z$, the algorithm always outputs a matching.*

Proof. The algorithm will end without outputting any matching if and only if in line 22 the while loop, it cannot find any strongly connected component with no outgoing edges. Note that every directed graph can be expressed as

a directed acyclic graph of its strongly connected components. Hence, we can always find a strongly connected component without outgoing edges. □

Theorem 2.4.8. *Algorithm 2 computes a maximal sequence of super-stable matchings.*

Proof. By Lemma 2.4.6 and Lemma 2.4.7, it is obvious that Algorithm 2 outputs a maximal sequence of super-stable matchings. □

2.4.2 Running Time of Algorithm 2

Theorem 2.4.9. *The running time of Algorithm 2 is $O(mn)$.*

Proof. Each time we add new edges into E_d, we need to compute strongly connected components of G_d. Computing strongly connected component of any directed graph $G' = (V', E')$ can be done in $O(E)$ time. Each edge e of G is added to G_d at most once, and G_d is always a subgraph of G. Hence, a naive implementation takes $O(m^2)$ on computing strongly connected components of G_d. As mentioned in [47], Pearce [63] and Pearce and Kelly [62] sketch how to extend their algorithm and that of Marchetti-Spaccamela et al. [53] to compute strongly connected component dynamically. Their algorithm runs in $O(mn)$ if edges can only be added to the graph and not deleted. The edges in G_d can be deleted during the algorithm, but they are deleted only when E_c is perfect on a strongly connected component without outgoing edges. Thus, other strongly connected components are unchanged. Also as mentioned in [47], the edges remaining in the selected strongly connected component can

be treated as they were added anew to the graph. Since the ranks of men increase as we output subsequent super-stable matchings, each edge can be added anew to G_d constant number of times. Thus, the amortized cost of edge insertion remains unchanged. The reader can easily check the other part of the algorithm takes at most $O(m)$ time. Hence, the total time is $O(mn)$. $\square$

2.4.3 Rotation Poset

We have shown all rotations $D(\mathcal{M}_G)$ can be found in time $O(mn)$ by Algorithm 2. It remains to show how to efficiently construct the precedence relation $\prec$ on $D(\mathcal{M}_G)$. Our construction is essentially the same as the construction given in [25] for the classical stable marriage problem. The only difference here is that one rotation for super-stable matchings can be one or multiple cycles, while one rotation for stable matchings in the classical stable marriage problem is always a cycle. For the completeness, we briefly sketch it. The reader can find more details in [25].

Let $\rho = \{(m_1, w_1), \cdots, (m_{k-1}, w_{k-1})\}$ be a rotation. Each rotation corresponds to the symmetric difference of two consecutive super-stable matchings in a maximal chain of $\mathcal{M}_G$. We say ρ is exposed to a super-stable matching M if $\rho \in M$ and we can eliminate it to obtain another super-stable matching $M' = M \oplus \rho$.

We say that ρ moves m_i down to w_{i+1} and moves w_i up to m_{i-1} for each $i \in [k]$. We also say ρ moves m_i below w if $w_{i-1} \prec_{m_i} w \preceq_{m_i} w_i$ and moves w_i above m if $m_i \preceq_{w_i} m \prec_{w_i} m_{i-1}$.

Let us consider a directed graph $(D(\mathcal{M}_G), E)$, E contains two types of edges:

- Type 1: For each pair $(m, w) \in \rho$, if ρ' is the unique rotation that moves m down to w, then $(\rho', \rho) \in E$.
- Type 2: If ρ moves m below w and $\rho' \neq \rho$ is the unique rotation that moves w above m, then $(\rho', \rho) \in E$.

The algorithm to construct the set E is simple. For each rotation ρ and each pair $(m_i, w_i) \in \rho$, label w_i in m_i's preference list with a type 1 label of ρ, and for each m strictly between m_i and m_{i-1} on w_i's list, label w_i in m's preference list with a type 2 label of ρ. Then, traverse each woman w on m's preference list, set $\rho^* = \emptyset$. If w has a type 1 label of ρ, add (ρ^*, ρ) into E and set $\rho^* = \rho$. If w has a type 2 label of ρ, add (ρ, ρ^*) into E. The algorithm takes $O(m)$ time since it only traverse the preference lists once.

It turns out that the transitive closure of $(D(\mathcal{M}_G), E)$ is exactly $(D(\mathcal{M}_G), \prec)$. The reader can find the proof in Irving and Gusfield's book [25].

We summarize Section 2.4 with the following theorem.

Theorem 2.4.10. *The partial order $(D(\mathcal{M}_G), \prec)$ can be constructed in $O(mn)$.*

Proof. The construction of $D(\mathcal{M}_G)$ takes $O(mn)$ time by running Algorithm 2. The precedence relation can be constructed in $O(m)$ time. Hence, the time complexity is $O(mn)$. □

2.5 Conclusion and Open Problems

In this chapter, we give two algorithms to characterize the set of all super-stable matchings. The first algorithm computes irreducible super-stable matchings in the super-stable matching lattice. The second algorithm constructs a rotation poset in $O(mn)$ time. One interesting open question is how to extend the results in this chapter to the roommate setting. In the roommate setting, the underlying graph can be arbitrary and each vertex in the graph ranks all its neighbors. Blocking pair, thus, is defined for every pair of vertices in this graph. For the stable matching problem in the roommate setting, semi-rotation was introduced in [25]. One might use the techniques in this chapter to find semi-rotation structures for the super-stable matching problem in the roommate setting.

Chapter 3

Linear Programming and its Applications

In this chapter we give a polyhedral characterisation for the set of all super-stable matchings and prove that the super-stable matching polytope is integral. This result implies that the maximum weight super-stable matching problem can be solved in polynomial time. Though the complexity of solving LP is usually higher than combinatorial methods, like in [18], this gives an alternative direction to solve the weighted super-stable matching problem. Previously, it has been shown that the stable matching polytope and the strongly stable matching polytope are integral [78, 70, 46], we complete all three cases by proving that the super-stable matching polytope is integral as well.

We prove a property called self-duality for the super-stable matching polytope, which also holds for the classical stable matching polytope [77] and the strongly stable matching polytope [46].

We also show how to use LP technique to solve an application of the stable marriage problem, called the path to stability problem.

See in Chapter 1 Section 1.3 for a brief introduction of the connection of stable matchings and linear programming.

The results presented in Section 3.1 and Section 3.2 are joint work with

Vijay Garg and published in [29]. Section 3.3 is joint work with Vijay Garg and published in [23].

3.1 The Super-stable Matching Polytope

In this section, we give a polyhedral characterisation of the set of all super-stable matchings and prove that the super-stable matching polytope is integral. The main result is the following theorem.

Theorem 3.1.1. *Let $G = (V, E)$ be a stable matching problem with ties where the graph G is bipartite, then the super-stable matching polytope $SUSM(G)$ is described by the following linear system:*

$$\sum_{u \in N(v)} x_{u,v} \leq 1, \qquad \forall v \in V, \tag{3.1a}$$

$$\sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + x_{u,v} \geq 1, \qquad \forall (u,v) \in E, \tag{3.1b}$$

$$x_{u,v} \geq 0, \qquad \forall (u,v) \in E \tag{3.1c}$$

where $N(v)$ denotes the set of neighbors of v in G, and $w >_u v$ means u strictly prefers w to v.

Inequalities (3.1a) and (3.1c) are standard matching constraints. If $x \in SUSM(G)$ is an integral solution, then constraint (3.1b) for an edge (u, v) implies that (u, v) does not block the matching associated with x. Thus integral solutions of $SUSM(G)$ are exactly super-stable matchings of G.

Proof. Let x be a feasible solution. Define E^+ to be the set of edges (u, v) with $x_{u,v} > 0$, and V^+ the set of vertices covered by E^+. For each $u \in V^+$, let $N^*(u)$ be the maximal elements in $\{i : x_{u,i} > 0\}$. Note that there might be multiple maximal elements that form a tie.

We first show the following lemma.

Lemma 3.1.2. *For each vertex u and each vertex $v \in N^*(u)$, u is the unique minimal element in $\{j : x_{j,v} > 0\}$ and that $\sum_{j \in N(v)} x_{j,v} = 1$.*

Proof. Indeed, (3.1b) implies

$$1 \leq \sum_{j >_v u} x_{j,v} + x_{u,v} = \sum_{j \in N(v)} x_{j,v} - \sum_{j <_v u} x_{j,v} - \sum_{\substack{j =_v u; \\ j \neq u}} x_{j,v} \leq 1 - \sum_{j <_v u} x_{j,v} - \sum_{\substack{j =_v u; \\ j \neq u}} x_{j,v} \leq 1 \tag{3.2}$$

Hence we have equality throughout in (3.2). This implies that $x_{j,v} = 0$ for each $\{j : j <_v u\}$ and each $\{j : j =_v u; j \neq u\}$ and that $\sum_{j \in N(v)} x_{j,v} = 1$. Since $x_{j,v} = 0$ for each $\{j : j =_v u; j \neq u\}$, v strictly prefers any other vertices in $\{j : x_{j,v} > 0\}$ over u, making u the unique minimal element in $\{j : x_{j,v} > 0\}$. □

We then prove that for any v such that $v \in N^*(u)$ for some u, then u is unique. Suppose not, there is a vertex $u' \neq u$ and $v \in N^*(u')$. By Lemma 3.1.2, u is the unique minimal element in $\{j : x_{j,v} > 0\}$, and u' is the unique minimal element in $\{j : x_{j,v} > 0\}$, contradiction.

Now let vertex sets U and W be the two parts of bipartite graph G. For any $u \in U \cap V^+$, there is at least one unique vertex $w \in N^*(u)$, such

that $\sum_{j\in N(w)} x_{j,w} = 1$. Let $F_W(x)$ be the set of these vertices. Formally, $F_W(x) = \{w : w \in N^*(u), u \in U \cap V^+\}$. Then we have $|F_W(x)| \geq |U \cap V^+|$. We also have that

$$|F_W(x)| = \sum_{w\in F_W(x)} \sum_{j\in N(w)} x_{j,w} = \sum_{j\in U\cap V^+} \sum_{w\in F_W(x)} x_{j,w} \leq \sum_{j\in U\cap V^+} 1 = |U \cap V^+| \tag{3.3}$$

implying that $|F_W(x)| = |U \cap V^+|$. Hence, we conclude that for each $u \in U \cap V^+$, $|N^*(u)| = 1$, which implies that u has an unique maximal element in $\{i : x_{u,i} > 0\}$. Since $|N^*(u)| = 1$, we denote this unique vertex as $x^*(u)$. We then have the following corollary.

Corollary 3.1.3. *There is a bijection between $U \cap V^+$ and $F_W(x)$, and for each $u \in U \cap V^+$, $\sum_{i\in N(u)} x_{u,i} = 1$.*

Similarly, we may define $F_U(x) = \{u : u \in N^*(w), w \in W \cap V^+\}$ and we have

Corollary 3.1.4. *There is a bijection between $W \cap V^+$ and $F_U(x)$, and for each $w \in W \cap V^+$, $\sum_{j\in N(w)} x_{j,w} = 1$.*

Then we have $|U \cap V^+| = |F_W(x)| \leq |W \cap V^+|$ and $|W \cap V^+| = |F_U(x)| \leq |U \cap V^+|$, implying $|U \cap V^+| = |W \cap V^+| = |F_W(x)| = |F_U(x)|$. Then any $u \in U \cap V^+$ is also in $F_U(x)$, hence, u has an unique minimal element, denoted by $x_*(u)$.

The bijection between $U \cap V^+$ and $F_W(x)$ forms a perfect matching M in (V^+, E^+), i.e. the set of edges $\{(u, x^*(u)) : u \in U \cap V^+\}$. Similarly, the bijection between $W \cap V^+$ and $F_U(x)$ forms another perfect matching N, i.e. the set of edges $\{(x^*(w), w) : w \in W \cap V^+\}$.

Consider the vector $x' = x + \varepsilon\chi^M - \varepsilon\chi^N$, with ε close enough to 0 (positive or negative). we will show that x' is also feasible solution of (3.1a)-(3.1c). It is easy to see that x' satisfies (3.1a) and (3.1c). For each vertex $u \in U \cap V^+$, there is an unique maximal element $x^*(u)$ and $(u, x^*(u)) \in M$ and an unique minimal element $x_*(u)$ and $(u, x_*(u)) \in N$, implying $\sum_{i \in N(u)} x'_{u,i} = \sum_{i \in N(u)} x_{u,i} \leq 1$. To see that x' satisfies (3.1b), let (u, v) be an edge in E^+ attaining equality in (3.1b). The case that $(u, v) \in M$ or $(u, v) \in N$ is trivial. So assume that $(u, v) \notin M$ and $(u, v) \notin N$. The edge $(u, x^*(u)) \in M$ and $x^*(u) >_u v$. There is no other edge in $\{(u, i) : i \in N(u)\}$ belongs to M. We prove that there is no edge (j, v) in M and $j >_v u$ since if $(j, v) \in M$, j is the minimal element of v. Similarly, we can prove that there is exact one edge $(j, v) \in N$ and $j >_v u$. Concluding, $\sum_{i >_u v} x'_{u,i} + \sum_{j >_v u} x'_{j,v} + x'_{u,v} = \sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + x_{u,v} = 1$. Let x be an extreme point. The feasibility of x' implies that $\chi^M = \chi^N$, that is, $M = N$. So $E^+ = M$ since the maximal element is the same as the minimal element for each vertex, hence, $x = \chi^M$. $\square$

3.1.1 Self-Duality

Let's consider the linear program:

$$(\mathbf{LP})\ \textit{maximize} \quad \sum_{(u,v)\in E} x_{u,v} \tag{3.4a}$$

$$\text{subject to} \quad x \in FSUSM(G,P). \tag{3.4b}$$

The dual problem with variables $(\alpha, \beta) \in R^V \times R^E$, is given by

$$(\mathbf{DLP})\ \textit{minimize} \quad \sum_{v\in V} \alpha_v - \sum_{(u,v)\in E} \beta_{u,v} \tag{3.5a}$$

$$\text{subject to} \quad \alpha_u + \alpha_v - \sum_{i<_u v} \beta_{u,i} - \sum_{j<_v u} \beta_{v,j} - \beta_{u,v} \geq 1, \quad \forall (u,v) \in E, \tag{3.5b}$$

$$\alpha_v \geq 0, \quad \forall v \in V, \tag{3.5c}$$

$$\beta_{u,v} \geq 0, \quad \forall (u,v) \in E. \tag{3.5d}$$

Lemma 3.1.5 (Self-Duality). *Each $x \in FSUSM(G,P)$ is an optimal solution of* (**LP**) *and (α, x) is an optimal solution of* (**DLP**)*, where*

$$\alpha_v = \sum_{i\in N(v)} x_{v,i} \quad \forall v \in V. \tag{3.6}$$

Proof. Let $x \in FSUSM(G,P)$ and let α be defined by (3.6). Let $(u,v) \in E$, we have that

$$
\begin{aligned}
&\alpha_u + \alpha_v - \sum_{i <_u v} x_{u,i} - \sum_{j <_v u} x_{v,j} - x_{u,v} \\
=& \sum_{i \in N(u)} x_{u,i} + \sum_{j \in N(v)} x_{v,j} - \sum_{i <_u v} x_{u,i} - \sum_{j <_v u} x_{v,j} - x_{u,v} \\
=& \sum_{i >_u v} x_{u,i} + \sum_{\substack{i =_u v; \\ i \neq v}} x_{u,i} + \sum_{j >_v u} x_{v,j} + \sum_{\substack{j =_v u; \\ j \neq u}} x_{v,j} + x_{u,v} \\
\geq& \sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{v,j} + x_{u,v} \\
\geq& 1,
\end{aligned}
$$

where the last two inequalities hold since x satisfies (3.1c) and (3.1b). Hence (α, x) is feasible for (DLP). To see that x and (α, x) are optimal solutions of (LP) and (DLP), respectively. Note that

$$
\sum_{v \in V} \alpha_v - \sum_{(u,v) \in E} x_{u,v} = 2 \sum_{(u,v) \in E} x_{u,v} - \sum_{(u,v) \in E} x_{u,v} = \sum_{(u,v) \in E} x_{u,v},
$$

so the objective function in (LP) and that in (DLP) are equal. Thus, the weak duality theorem of linear programming implies that x is optimal for (LP) and (α, x) is optimal for (DLP). □

3.1.2 Partial Order Preference Lists

Partial order preference lists are generalisation of preference lists with ties in such a way that the preference list of each man or woman is an arbitrary partial order. It turns out that the linear system (3.1a)-(3.1c) can also describe the set of all super-stable matchings with partial order preference list.

Theorem 3.1.6. *Let $G = (V, E)$ be a stable matching problem with partial order preference lists where the graph G is bipartite, then the super-stable matching polytope $SUSMP(G)$ is described by the following linear system:*

$$\sum_{u \in N(v)} x_{u,v} \leq 1, \qquad \forall v \in V, \tag{3.7a}$$

$$\sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + x_{u,v} \geq 1, \qquad \forall (u,v) \in E, \tag{3.7b}$$

$$x_{u,v} \geq 0, \qquad \forall (u,v) \in E \tag{3.7c}$$

where $N(v)$ denotes the set of neighbors of v in G, and $w >_u v$ means u strictly prefers w to v.

Proof. Let x be a feasible solution. Define E^+ to be the set of edges (u, v) with $x_{u,v} > 0$, and V^+ the set of vertices covered by E^+. For each $u \in V^+$, let $N^*(u)$ be the maximal elements in $\{i : x_{u,i} > 0\}$. Note that there might be multiple maximal elements that are incomparable to each other.

The following lemma is an analogue to Lemma 3.1.2.

Lemma 3.1.7. *For each vertex u and each vertex $v \in N^*(u)$, then u is the unique minimal element in $\{j : x_{j,v} > 0\}$ and that $\sum_{j \in N(v)} x_{j,v} = 1$.*

Proof. Indeed, (3.1b) implies

$$1 \leq \sum_{j >_v u} x_{j,v} + x_{u,v} = \sum_{j \in N(v)} x_{j,v} - \sum_{j <_v u} x_{j,v} - \sum_{j \parallel_v u} x_{j,v} \leq 1 - \sum_{j <_v u} x_{j,v} - \sum_{j \parallel_v u} x_{j,v} \leq 1 \tag{3.8}$$

, where $w \parallel_u v$ means w is incomparable with v in u's preference list. Hence we have equality throughout in (3.2). This implies that $x_{j,v} = 0$ for each $\{j : j <_v u\}$ and each $\{j : j \parallel_v u\}$ and that $\sum_{j \in N(v)} x_{j,v} = 1$. Since $x_{j,v} = 0$ for each $\{j : j \parallel_v u\}$, v strictly prefers any other vertices in $\{j : x_{j,v} > 0\}$ over u, making u the unique minimal element in $\{j : x_{j,v} > 0\}$. □

The rest of the proof is essentially the same as in Theorem 3.1.1. □

3.2 The Strongly Stable Matching Polytope

Kunysz [46] gives a linear system that characterizes the set of all strongly stable matchings and proves this linear system is integral using the duality theory of linear programming. Here, we give an alternate and simpler proof that does not rely on the duality theory and uses only Hall's theorem.

Theorem 3.2.1 (Kunysz, [46]). *Let $G = (V, E)$ be a stable matching problem with ties where the graph G is bipartite, then the strongly stable matching polytope $SSM(G)$ is described by the following linear system:*

$$\sum_{u \in N(v)} x_{u,v} \leq 1, \qquad \forall v \in V, \tag{3.9a}$$

$$\sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + \sum_{k =_u v} x_{u,k} \geq 1, \qquad \forall (u,v) \in E, \tag{3.9b}$$

$$\sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + \sum_{k =_v u} x_{k,v} \geq 1, \qquad \forall (u,v) \in E, \tag{3.9c}$$

$$x_{u,v} \geq 0, \qquad \forall (u,v) \in E \tag{3.9d}$$

where $N(v)$ denotes the set of neighbors of v in G, and $w >_u v$ means u strictly prefers w to v.

We give an alternative proof that does not rely on the duality theory of linear programming.

Proof. It is easy to verify that the incidence vector of any strongly stable matching satisfies constraints (3.9a)-(3.9d). We need to prove each extreme point of the polytope defined by (3.9a)-(3.9d) is integral.

Let x be a feasible solution. Define E^+ to be the set of edges (u, v) with $x_{u,v} > 0$, and V^+ the set of vertices covered by E^+. For each $u \in V^+$, let $N^*(u)$ be the set of maximal elements in $\{i : x_{u,i} > 0\}$. Note that there might be multiple maximal elements that form a tie. Similarly, for each $u \in V^+$, let $N_*(u)$ be the set of minimal elements in $\{i : x_{u,i} > 0\}$.

We first show the following lemma.

Lemma 3.2.2. *For each vertex u and each vertex $v \in N^*(u)$, then $u \in N_*(v)$ and that $\sum_{j \in N(v)} x_{j,v} = 1$, and $\sum_{k=_u v} x_{u,k} \geq \sum_{k=_v u} x_{k,v}$.*

Proof. Indeed, (3.9c) implies

$$1 \leq \sum_{j >_v u} x_{j,v} + \sum_{k=_v u} x_{k,v} \leq \sum_{j \in N(v)} x_{j,v} - \sum_{j <_v u} x_{j,v} \leq 1 - \sum_{j <_v u} x_{j,v} \leq 1 \qquad (3.10)$$

Hence we have equality throughout in (3.10). This implies that $x_{j,v} = 0$ for each $\{j : j <_v u\}$ and that $\sum_{j \in N(v)} x_{j,v} = 1$. Hence, $u \in N_*(v)$. (3.9b) implies

$$1 \leq \sum_{j >_v u} x_{j,v} + \sum_{k=_u v} x_{u,k} \leq 1 - \sum_{k=_v u} x_{k,v} + \sum_{k=_u v} x_{u,k}$$

Hence, $\sum_{k=_u v} x_{u,k} \geq \sum_{k=_v u} x_{k,v}$. □

Now let vertex sets U and W be the two parts of bipartite graph G. Let $E^*(x)$ be the set of edges $\{(u,v) : u \in U \cap V^+, v \in N^*(u)\}$ and $F_W(x)$ be the set of vertices in W and covered by $E^*(x)$. We will show that the subgraph induced by $E^*(x)$ contains a perfect matching M.

Suppose not, by Hall's theorem, let S be the unique critical subset of $U \cap V^+$. A subset of X is critical if it is maximally deficient and contains no maximally deficient proper subset. Hence, we have $|N(S)| < |S|$ and there is a matching M' saturating for $N(S)$ in the subgraph induced by $S \cup N(S)$. Fixing the matching M', let S' be the set of vertices in S that is matched in M', so $S' \subsetneq S$. By Lemma 3.2.2, we have

$$\sum_{u \in S'} \sum_{k \in N^*(u)} x_{u,k} \geq \sum_{v \in N(S)} \sum_{k =_v M'(v)} x_{k,v} \tag{3.11}$$

$$\geq \sum_{v \in N(S)} \sum_{k \in S} x_{k,v} \tag{3.12}$$

$$= \sum_{u \in S} \sum_{k \in N^*(u)} x_{u,k} \tag{3.13}$$

The first equality follows from Lemma 3.2.2. The second inequality follows from that each vertex $v \in N(S)$ is indifferent with all neighbors. The third equality follows by double counting. Hence, the vertices in $S \backslash S'$ are isolated, contradiction. So we also have $|U \cap V^+| = |F_W(x)|$. Again by Lemma 3.2.2, we have

$$\sum_{u \in U \cap V^+} \sum_{k =_u M(u)} x_{u,k} \geq \sum_{v \in F_W(x)} \sum_{k =_v M(v)} x_{k,v} \geq \sum_{u \in U \cap V^+} \sum_{k =_u M(u)} x_{u,k} \tag{3.14}$$

Hence, we have equality throughout (3.14). So for each $v \in F_W(x)$ and each $k \in N_*(v)$, $v = N^*(k)$.

Similarly, let E_* be the set of edges $\{(u,v) : v \in W \cap V^+, u \in N^*(v)\}$ and $F_U(x)$ be the set of vertices in U and covered by E^+. The subgraph induced by E_* contains a perfect matching N. Hence, $|W \cap V^+| = |F_U(x)|$. So $|U \cap V^+| = |W \cap V^+| = |F_U(x)| = |F_W(x)|$. It follows that E_* is exactly the set of edges $\{(u,v) : u \in U \cap V^+, v \in N_*(u)\}$.

Consider the vector $x' = x + \varepsilon\chi^M - \varepsilon\chi^N$, with ε close enough to 0 (positive or negative). we will show that x' is also feasible solution of (3.9a)-(3.9d). It is easy to see that x' satisfies (3.9a) and (3.9d). For each vertex $u \in U \cap V^+$, there is one edge in M and one edge in N incident to it, implying $\sum_{i \in N(u)} x'_{u,i} = \sum_{i \in N(u)} x_{u,i} \leq 1$. To see that x' satisfies (3.9b), let (u,v) be an edge in E^+ attaining equality in (3.9b). First we know that there is one edge in M incident to u such that $M(u) \geq_u v$. Also there is no M edge incident to v and $M(v) >_v u$ since $M(v) \in N_*(v)$. Similarly, there is one edge in N incident to v such that $M(v) \geq_v u$. Then we have two cases. If there is one edge in N incident to v such that $N(v) >_v u$, then there is no edge in N incident to u such that $M(u) \geq_u v$, because otherwise, $u \in N^*(v)$, which contradicts with $N(v) >_v u$. The other case is that there is one edge in N incident to v such that $N(v) =_v u$, then there must be an edge in N incident to u such that $N(u) =_u v$. Hence, $\sum_{i >_u v} x'_{u,i} + \sum_{j >_v u} x'_{j,v} + \sum_{k =_u v} x'_{u,k} = \sum_{i >_u v} x_{u,i} + \sum_{j >_v u} x_{j,v} + \sum_{k =_u v} x_{u,k} = 1$. Similarly, we can prove x' satisfies (3.9c). Let x be an extreme point. The feasibility of x' implies that $\chi^M = \chi^N$, that is,

$M = N$. So for each $v \in V^+$, we have $N^*(v) = N_*(v)$ and $\sum_{k \in N^*(v)} x_{k,v} = 1$. Then x is an extreme point of the perfect matching polytope of (V^+, E^+). x must be a perfect matching in (V^+, E^+). □

3.3 Paths to Stability

In many applications, it is useful to consider the initial state of the system as an arbitrary assignment of men to women and then to find a path to a stable matching. For example, suppose that we consider a system in which there are more women than men and suppose that every man is matched to a unique woman such that there is no blocking pair. Now, if a new man or a woman joins the system, it is more natural to start with the initial state as the existing assignment rather than the empty matching. In particular, if there is some cost associated with breaking up an existing couple, then we may be interested in the paths to stability that are of short lengths.

As another example, suppose that we have a stable matching. In a dynamic preference mechanism, a woman may change her list of preferences. The existing matching may not be stable under new preferences of the woman. Again, it is more natural to start with the existing matching and then to find a path to a stable matching under new preferences. Thus, the generalization allows one to consider a *dynamic* stable matching algorithm in which preferences of a man or a woman may change and the goal is to find a stable matching under new preferences.

Our another goal is to find a matching that is not too far from the

original matching (or the initial proposal vector). Given any proposal vector I, the *regret* of a man is defined as the rank the woman he is assigned in I, i.e., if a man is assigned his k^{th} top choice in I then his regret is k. Given two proposal vectors I and M, we define the distance between I and M, $dist(I, M)$ as the sum of differences of regrets for all men in I and M, i.e. the L_1 distance between two vectors, $dist(I, M) = \|I - M\|_1$.

The following algorithm is based on a linear programming formulation of the stable marriage problem by Rothblum [70]. By appropriately defining the objective function to minimize the distance from the initial proposal vector, we get a polynomial time algorithm to find the closest stable marriage.

Given an arbitrary proposal vector (not necessarily a matching) I, we want to find a stable matching M such that the distance between the proposal vector I and the stable matching M is minimized over all stable matchings, $\mathcal{M}$. The distance we consider here is L_1 distance, a.k.a Manhattan distance between two vectors. We denote the distance as $dist(I, M) = \|I - M\|_1$. The problem hence can be rephrased as: find the marriage $M \in \mathcal{M}$ that minimizes $dist(I, M)$.

It is well-known that the convex hull of stable matchings of an arbitrary bipartite preference system can be described by a linear system [70] as follows:

$$\sum_{j\in[w]} x_{i,j} \leq 1 \quad \forall i \in [m] \tag{3.15a}$$

$$\sum_{i\in[m]} x_{i,j} \leq 1 \quad \forall j \in [w] \tag{3.15b}$$

$$\sum_{i'\in[m];i'>_j i} x_{i',j} + \sum_{j'\in[w];j'>_i j} x_{i,j'} + x_{i,j} \geq 1 \quad \forall (i,j) \in [m]\times[w] \tag{3.15c}$$

Here, we define that for each man or woman i, $p >_i q$ denotes that i prefers p over q in his/her preference list. Rothblum [70] proved that the linear system above is integral, i.e. every basic feasible solution of Equation 3.15 is integral. Suppose that every possible marriage (i, j) has a cost $c(i, j)$, we can find a minimum-cost stable matching in polynomial time by solving the LP above.

Now we show that our problem of minimizing the distance between an initial proposal vector and any stable matching can be translated into a minimum-cost stable matching problem with a carefully designed cost function.

For each pair (i, j), we assign the cost $c(i, j) = |I[i] - mrank[i][j]|$. Hence, for each stable matching M, we have:

$$\begin{aligned} dist(I, M) &= \sum_{i\in[m]} |I[i] - M[i]| \\ &= \sum_{(i,j)\in[m]\times[w]} c(i, j) \cdot \mathbb{1}_{\rho(M,i)=j} \end{aligned}$$

Hence, we can rewrite our problem as:

$$\textit{minimize} \quad \sum_{(i,j)\in[m]\times[w]} c_{i,j} \cdot x_{i,j} \tag{3.16a}$$

$$\text{subject to} \quad \sum_{j\in[w]} x_{i,j} \leq 1 \quad \forall i \in [m] \tag{3.16b}$$

$$\sum_{i\in[m]} x_{i,j} \leq 1 \quad \forall j \in [w] \tag{3.16c}$$

$$\sum_{i'\in[m];i'>_j i} x_{i',j} + \sum_{j'\in[w];j'>_i j} x_{i,j'} + x_{i,j} \geq 1 \quad \forall (i,j) \in [m] \times [w] \tag{3.16d}$$

Solving the above LP gives us a stable matching that is nearest to the initial proposal vector. This LP has $O(n^2)$ variables and constraints where n is the total number of men and women. However, we note that the minimum-cost stable matching problem can be reduced to the minimum-cost closed subset of a poset due to the rotation poset structure of stable matching problem. See [25] for more details of rotation poset. Feder [16] has shown that the minimum-cost stable matching problem in a bipartite preference system can be solved in $O(n^3)$ time if $max(c_{i,j}) = O(n)$.

We summarize the preceding discussion as the following theorem.

Theorem 3.3.1. *Given an arbitrary proposal vector I, we can find a stable matching M that minimizes the distance $dist(I, M)$ over all stable matchings in $O(n^3)$ time where n is the total number of men and women.*

3.4 Conclusion and Open Problems

In this chapter, we give proofs for the integrality of the super-stable matching polytope and the strongly stable matching polytope. As mentioned in Chapter 1 Section 1.3, Kiraly and Pap [42] proved the LP formulation by Rothblum is totally dual integral (TDI). It would be interesting to investigate whether the linear systems for the super-stable matching polytope and the strongly stable matching polytope given in this chapter is TDI.

Chapter 4

Parallel Algorithms for Popular Matchings in the Housing Allocation Model

The results presented in this chapter are joint work with Vijay Garg and published in [28].

4.1 Introduction

The notion of a popular matching was first introduced in [22] in the context of the stable marriage problem. We say that a matching M is *more popular than* M' if the number of nodes that prefer M to M' exceeds the number of nodes that prefer M' to M. A matching M is popular if M is optimal under the *more popular than* relation. Gupta et al. [24] and Faenza et al. [14] recently showed that the popular matching problem is NP-complete in the general roommate setting. The popular matching problem we consider is from [6] such that the preference system is only one-sided. Abraham et al. [6] gave a linear-time algorithm for the problem in the case of strictly-ordered preference lists and a polynomial-time algorithm for the case of preference lists with ties. There are other problems with other definitions of optimality such as *Pareto optimal* matching [5], *rank-maximal* matching [34] etc. We do not

discuss them here.

The matching problem in the normal case, that is the problem of checking if a given graph has a perfect matching, and the corresponding search problem of finding a perfect matching have received considerable attention in the field of parallel computation. Tutte and Lovasz [50] observed that there is an RNC algorithm for the decision problem. The search version was shown to be in RNC by Karp, Upfal and Wigderson [39] and subsequently by Mulmuley, Vazirani and Vazirani using the celebrated Isolation Lemma [58]. We note that no NC or RNC algorithms were known for the matching problem in preference systems prior to this work. The problem of finding an NC algorithm for the stable marriage problem has been open for a long time. Mayr and Subramanian [54] showed that the stable marriage problem is CC-complete. Subramanian [76] defined the complexity class CC as the set of problems log-space reducible to the comparator circuit value problem (CCV). Cook et al. [11] conjectured that CC is incomparable with the parallel class NC, which implies none of the CC-complete problems has an efficient polylog time parallel algorithm. Recently, Zheng and Garg [81] showed that computing a Pareto optimal matching in the housing allocation model is in CC and computing the core of a housing market is CC-hard.

4.1.1 Our Contributions

1. We give NC algorithms for both the popular matching problem and the maximum-cardinality popular matching problem in the setting of

strictly-ordered preference lists.

2. In the case that preference lists contain ties, we show that maximum-cardinality bipartite matching is NC-reducible to popular matching.

3. We also give an NC algorithm to find the "next" stable matching if one stable matching is given. We will define "next" in Section 4.6.

4.2 Preliminaries

4.2.1 The Popular Matching Problem

Let $\mathcal{A}$ be a set of applicants and $\mathcal{P}$ be a set of posts, associated with each member of $\mathcal{A}$ is a preference list (possibly involving ties) comprising a non-empty subset of the elements of $\mathcal{P}$. An instance of the *popular matching problem* is a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, E)$ and a partition $E = E_1 \dot{\cup} E_2 \dot{\cup} \cdots \dot{\cup} E_r$ of the edge set. The partition E consists of all pairs (a, p) such that post p appears in the preference list of applicant a and we say that each edge $(a, p) \in E_i$ has a rank i if post p is on the i-th position of the preference list of applicant a. If $(a, p) \in E_i$ and $(a, p') \in E_j$ with $i < j$, we say that a prefers p to p'. If $i = j$, we say that a is indifferent between p and p'. We say that preference lists are strictly ordered if no applicant is indifferent between any two posts on his/her preference list. Otherwise, we say that preference lists contain ties.

A *matching* M of G is a set of edges no two of which share an endpoint. A node $u \in \mathcal{A} \cup \mathcal{P}$ is either unmatched or matched to some node, denoted by $M(u)$. We say that an applicant a prefers matching M' to M if (i) a is

matched in M' and unmatched in M, or (ii) a is matched in both M' and M, and a prefers $M'(a)$ to $M(a)$. Let $\mathcal{M}$ be the set of matchings in G and let $M, M' \in \mathcal{M}$. Let $P(M, M')$ denote the set of applicants who prefer M to M'. Define a "*more popular than*" relation $\succ$ on M as follows: if $M, M' \in \mathcal{M}$, then M' is *more popular than* M, denoted by $M' \succ M$, if $|P(M', M)| > |P(M, M')|$.

Definition 4.2.1. A matching $M \in \mathcal{M}$ is popular if there is no matching M' such that $M' \succ M$.

The popular matching problem is to determine if a given instance admits a popular matching, and to find such a matching, if one exists. Note that popular matchings may have different sizes, a largest popular matching may be smaller than a maximum-cardinality matching since no maximum-cardinality matching needs to be popular. The maximum-cardinality popular matching problem then is to determine if a given instance admits a popular matching, and to find a largest such matching, if one exists. Figure 4.1 shows an example of a popular matching instance. The reader can check that $\{(a_1, p_1), (a_2, p_2), (a_3, p_4), (a_4, p_3), (a_5, p_5), (a_6, p_7),$
$(a_7, p_8), (a_8, p_9)\}$ is a popular matching.

As in [6], we add a unique *last resort post* $l(a)$ for each applicant a and assign the edge $(a, l(a))$ higher rank than any edge incident on a. In this way, we can assume that every applicant is matched, since any unmatched applicant can be matched to his/her unique last resort post. From now on, we only focus on matchings that are *applicant-complete*, and the size of a matching is the number of applicants not matched to their last resort posts.

a_1 : p_1 p_4 p_5 p_2 p_6
a_2 : p_4 p_5 p_7 p_2 p_8
a_3 : p_4 p_1 p_3 p_8
a_4 : p_1 p_7 p_4 p_3 p_9
a_5 : p_5 p_1 p_7 p_2 p_6
a_6 : p_7 p_6
a_7 : p_7 p_4 p_8 p_2
a_8 : p_7 p_4 p_1 p_5 p_9 p_3

Figure 4.1: A popular matching instance I

Definition 4.2.2. A matching $M \in \mathcal{M}$ is applicant-complete if each applicant $a \in \mathcal{A}$ is matched to some post $p \in \mathcal{P}$.

4.3 Finding Popular Matching in NC

4.3.1 Characterizing Popular Matchings

We restrict our attention to strictly-ordered preference lists. For each applicant a, let $f(a)$ denote the first-ranked post on a's preference list. We call any such post p an f-post, and denote by $f^{-1}(p)$ the set of applicants a for which $f(a) = p$. For each applicant a, let $s(a)$ denote the first non-f-post on a's preference list (note that $s(a)$ always exists, due to the introduction of $l(a)$). We call any such post p an s-post, and remark that f-posts are disjoint from s-posts. We also call any last resort post p an l-post.

The following theorem, proved in [6], completely characterizes popular matchings.

Theorem 4.3.1 ([6])**.** *A matching M is popular if and only if*

(i) every f-post is matched in M, and

(ii) for each applicant a, $M(a) \in \{f(a), s(a)\}$.

Let G' be the reduced graph of G that only includes f-posts and s-posts. For a reduced graph G', let M be a popular matching, and let a be an applicant. Denote by $O_M(a)$ the post on a's reduced preference list to which a is not assigned in M. Note that since G' is a reduced graph of G, $O_M(a)$ is well-defined. If a is matched to $f(a)$ in M, then $O_M(a) = s(a)$, whereas if a is matched to $s(a)$ in M, then $O_M(a) = f(a)$.

4.3.2 Algorithmic Results

Now we show Algorithm 3 is an NC algorithm for the popular matching problem with strictly-ordered preference lists. First we construct the reduced graph G' from G. Then we find an applicant-complete matching M in G'. Hence for each applicant a, $M(a) \in \{f(a), s(a)\}$. Then for any f-post p that is unmatched in M, we match p with any applicant in $f^{-1}(p)$.

The most non-trivial part is line 4 that determines an applicant-complete matching M in the reduced graph G'. Perfect matching in bipartite graph is in Quasi-NC [17], but we do not know whether it is in NC. Recent results in [7] show that perfect matching in planar graph is in NC. But the reduced graph for popular matching problem is not necessarily planar. It is easy to check that the reduced graph G' may contain a subgraph that is a subdivision of the complete bipartite graph $K_{3,3}$.

Algorithm 3: Popular Matching

```
1  Input: Graph G = (A ∪ P, E).
2  Output: A popular matching M or determine that no such
     matching.
3  G' := reduced graph of G;
4  if G' admits an applicant-complete matching M then
5      for each f-post p unmatched in M in parallel do
6          let a be any applicant in f^{-1}(p);
7          promote a to p in M;
8      return M;
9  else
10     return "no popular matching";
```

We first show how to construct the reduced graph G' from G in parallel (line 3). For each post p, we check if there is any incident edge $(a, p) \in E_1$. Let $\mathcal{F}$ be the set of such posts, which corresponds to all f-posts. Then for each post $p \in \mathcal{F}$, we remove all incident edges $(a, p) \notin E_1$. After that, for each applicant a, we find the highest ranked incident edge $(a, p) \notin E_1$, which corresponds to $s(a)$, and remove all other incident edges. The remaining graph must be G'. It is clear that each step can be done in logarithmic time with a polynomial number of operations.

It remains to show how to find an applicant-complete matching in G' (line 4), or determine that no such matching exists in NC. Now we explain Algorithm 4 that finds an applicant-complete matching.

The *while* loop (line 4) gradually matches applicants to posts of degree 1 or 2 until there is no post of degree 1. Then, either the remaining graph admits a perfect matching or we can conclude that there is no applicant-

Algorithm 4: Applicant-Complete Matching

Input: Graph $G' = (\mathcal{A} \cup \mathcal{P}, E')$.
Output: An applicant-complete matching M or determine that no such matching exists.
$M := \emptyset$;
while some post p has degree 1
    Find all maximal paths that end at p;
    for each edge (p', a') at an even distance from some p **in parallel do**
        $M := M \cup \{(p', a')\}$;
        $G' := G' - \{p', a'\}$;
for each post p has degree 0 **in parallel do**
    $G' := G' - p$
// Every post now has degree at least 2;
// Every applicant still has degree 2;
if $|\mathcal{P}| < |\mathcal{A}|$ **then**
    return "no applicant-complete matching";
else
    // G' decomposes into a family of disjoint even cycles
    $M' :=$ any perfect matching of G';
    return $M \cup M'$;

complete matching. We show the details below.

First, we identify all vertices of degree 2 in G'. Note that all applicants have degree 2, but posts may have any degree. We only need to identify posts of degree 2. Some of these vertices might be connected to each other, in which case we get paths formed by these vertices. We can extend these paths, by the doubling trick in polylog time to find maximal paths consisting of degree 2 vertices. Let the vertices of the path be $(v_1, v_2, \cdots, v_k)$. Further, let v_0 be the vertex we would get if we extended this path from v_1 side and v_{k+1} be the one we would get from v_k side. Note that $\deg(v_i) = 2$ for $i = 1, \cdots, k$ but not for $i = 0, k + 1$.

Then, in parallel, we consider each maximal path with at least one of v_0 and v_{k+1} of degree 1. W.l.o.g, let v_0 be the vertex of degree 1. For each such path, we add each edge at an even distance from v_0 to M (e.g. the edge (v_0, v_1) is at zero distance from v_0 and must be added to M) and delete $v_0, \cdots, v_k$ and their incident edges. Note that v_0 and v_{k+1} can only be posts since all applicants have degree exactly 2. Hence, any maximal path must have even length and v_{k+1} is not matched. In the case both end points have degree 1, we only consider this path once and choose v_0 or v_{k+1} to be matched arbitrarily. After one round, there would be some new vertices of degree 1 because the degree of v_{k+1} decreases by 1 for each maximal path that ends at v_{k+1}. Run the same process until there is no post that has degree 1. After removing any isolated posts, we can conclude that either there is no applicant-complete matching, or the remaining graph is a family of disjoint

even cycles.

4.3.2.1 Correctness

Algorithm 4 begins by repeatedly matching maximal paths $(v_0, v_1, \cdots, v_{k+1})$ with $\deg(v_0) = 1$. After first round, no subsequent augmenting path can include any vertices v_i for $i = 0, 1, \cdots, k$ since they are matched and any alternating path that includes them must end at v_0, which is matched and has degree 1. So we can remove all matched vertices from consideration. The same argument holds for subsequent rounds. Also note that the while loop always terminates because whenever we find a post of degree 1, we match at least one edge (v_0, v_1) and remove at least two vertices that are $\{v_0, v_1\}$.

Now we have a matching and we only need to match remaining posts and applicants. All remaining posts have degree at least 2, while all remaining applicants still have degree exactly 2. Now, if $|\mathcal{P}| < |\mathcal{A}|$, G' cannot admit an applicant-complete matching by Hall's Marriage Theorem [26]. Otherwise, we have that $|\mathcal{P}| \geq |\mathcal{A}|$, and by a double counting argument, we have $2|\mathcal{P}| \leq \sum_{p \in \mathcal{P}} \deg(p) = 2|\mathcal{A}|$. Hence, it must be that $|\mathcal{P}| = |\mathcal{A}|$ and every post has degree exactly 2. G' becomes 2-regular bipartite graph and consists of disjoint union of even cycles. By choosing any edge e in an even-length cycle C, even distance (resp. odd distance) from e is well-defined. Choosing all edges of even distance yields a perfect matching in G'. Now we have an applicant-complete matching in G'. Hence for each applicant a, $M(a) \in \{f(a), s(a)\}$. Then for any f-post p that is unmatched in M, we match p with any applicant in $f^{-1}(p)$.

By Theorem 4.3.1, the resulting matching is a popular matching.

4.3.2.2 Complexity

Lemma 4.3.2 proves that the *while* loop in Algorithm 4 runs $O(\log(n))$ number of times.

Lemma 4.3.2. *The* while loop (line 4) runs $O(\log(n))$ number of times.

Proof. For any vertex v of $\deg(v) \geq 3$ that is reduced to degree of 1, it must be the end point of $\deg(v) - 1$ maximal paths. If in round r, s.t. $r > 1$, there are t vertices of degree 1 (for some constant t), then we must have deleted at least $2t$ vertices in round $r - 1$. After round r, we have deleted at least $(2^r - 1)t$ vertices. Hence, it is clear that the while loop can be run at most $\lceil \log(n) \rceil + 1$ times since the total number of vertices is bounded by n. □

Finding all maximal paths of degree 2 vertices and calculating the distance from v_0 in the path can be done in polylog time. Furthermore, the *while* loop runs at most a logarithmic number of times. Finding a perfect matching in a 2-regular bipartite graph i.e. graph consisting of even-length cycles is in NC. More generally, searching for a perfect matching in regular bipartite graphs can be done in NC [49]. So, Algorithm 4 is in NC. The *for* loop in Algorithm 3 can be done in constant time since for every f-post p, $f^{-1}(p)$ is disjoint from each other.

We summarize the preceding discussion in the following theorem.

Theorem 4.3.3. *We can find a popular matching, or determine that no such matching exists in NC.*

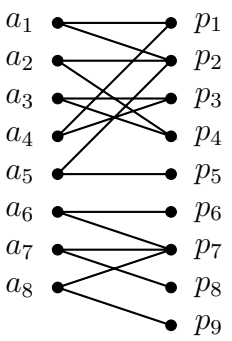
4.3.3 Example of Popular Matchings

To illustrate Algorithm 4, we provide a detailed example. Figure 4.1 shows the preference lists for a popular matching instance I. The set of f-posts is $\{p_1, p_4, p_5, p_7\}$ and the set of s-posts is $\{p_2, p_3, p_6, p_8, p_9\}$.

Figure 4.2 shows the reduced preference lists of I and reduced graph G'.

$a_1 :$ $\underline{p_1}$ p_2
$a_2 :$ p_4 $\underline{p_2}$
$a_3 :$ $\underline{p_4}$ p_3
$a_4 :$ p_1 $\underline{p_3}$
$a_5 :$ $\underline{p_5}$ p_2
$a_6 :$ $\underline{p_7}$ p_6
$a_7 :$ p_7 $\underline{p_8}$
$a_8 :$ p_7 $\underline{p_9}$

(a) The reduced preference lists of I with popular matching M denoted by underlining



(b) The reduced graph G' of G

Figure 4.2: The reduced instance of I

In the while loop of Algorithm 4, pairs $(a_8, p_9), (a_6, p_6)$, $(a_7, p_8), (a_5, p_5)$ are matched. Figure 4.3 shows the reduced graph after the while loop of Algorithm 4. The graph consists of only even-length cycles. Choose one perfect matching in the reduced graph such as pairs $(a_1, p_1), (a_2, p_2), (a_3, p_4), (a_4, p_3)$,

we obtain an applicant-complete matching. Note that one f-post p_7 is not matched in this applicant-complete matching. So we can promote any applicant from $\{a_6, a_7, a_8\}$ to match with p_7, e.g. a_6 is matched to p_7. The resulting popular matching M is $\{(a_1, p_1), (a_2, p_2), (a_3, p_4), (a_4, p_3), (a_5, p_5), (a_6, p_7), (a_7, p_8), (a_8, p_9)\}$.

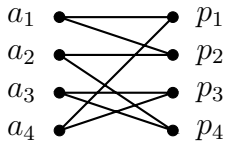


Figure 4.3: The reduced graph after the while loop of Algorithm 4

4.4 Finding Maximum-Cardinality Popular Matching in NC

We now consider the maximum-cardinality popular matching problem. Let $\mathcal{A}_1$ be the set of all applicants a with $s(a) = l(a)$, and let $\mathcal{A}_2 = \mathcal{A} - \mathcal{A}_1$. Our target matching must satisfy conditions (i) and (ii) of Theorem 4.3.1, and among all such matchings, allocate the fewest $\mathcal{A}_1$-applicants to their last resort. To be able to find maximum-cardinality matching in NC, we need another characterization of popular matching problem called *switching graph* [56], a directed graph which captures all the possible ways in which applicants may form different popular matchings by switching between the two posts on their reduced preference lists.

Given a popular matching M for an instance $G = (\mathcal{A} \cup \mathcal{P}, E)$, the switching graph G_M of M is a directed graph with a vertex for each post

p, and a directed edge (p_i, p_j) for each applicant a, where $p_i = M(a)$ and $p_j = O_M(a)$. Then each edge is labelled with the applicant that it represents. A *component* of G_M is any maximal weakly connected subgraph of G_M. An applicant (resp. post) is said to be in a component, or path, or cycle of G_M if the edge (resp. vertex) representing it is in that component, path or cycle. The following lemma in [56] gives some simple properties of switching graphs.

Lemma 4.4.1 ([56], Lemma 1)**.** *Let M be a popular matching for an instance of $G = (\mathcal{A} \cup \mathcal{P}, E)$, G_M be the switching graph of M. Then*

(i) Each vertex in G_M has outdegree at most 1*.*

(ii) The sink vertices of G_M are those vertices corresponding to posts that are unmatched in M, and are all s-post vertices.

(iii) Each component of G_M contains either a single sink vertex or a single cycle.

A component of a switching graph G_M is called a *cycle component* if it contains a cycle, and a *tree component* if it contains a sink vertex. Each cycle in G_M is called a *switching cycle*. If T is a tree component of G_M with sink vertex p, and if q is another s-post vertex in T, the unique path from q to p is called a *switching path*. Note that each cycle component of G_M has a unique switching cycle, but each tree component may have zero or multiple switching paths; to be precise it has one switching path for each s-post vertex it contains, other than the sink vertex.

Figure 4.4 shows the switching graph G_M for popular matching M. There are one switching cycle and two switching paths starting from p_8 and p_9 respectfully.

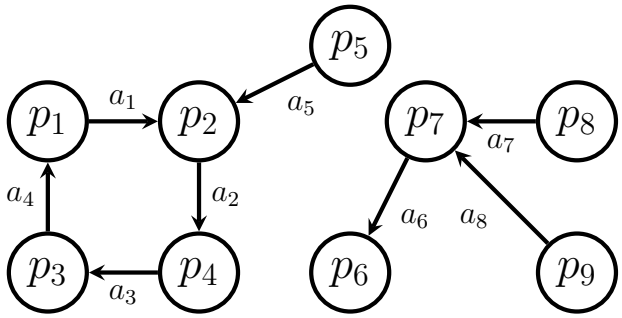


Figure 4.4: The switching graph G_M for popular matching M

Lemma 4.4.1 shows that the switching graph G_M is indeed a directed pseudoforest. Next we give several NC algorithms for finding all switching cycles and switching paths in G_M.

4.4.1 Finding Cycles in Pseudoforest in NC

Definition 4.4.1. A **pseudoforest** is an undirected graph in which every connected component has at most one cycle. A **pseudotree** is a connected pseudoforest. A **directed pseudoforest** is a directed graph in which each vertex has at most one outgoing edge, i.e., it has outdegree at most one. A **directed 1-forest** (most commonly called a functional graph, sometimes maximal directed pseudoforest) is a directed graph in which each vertex has outdegree exactly one.

It is easy to see that every weakly connected component in a directed

pseudoforest contains either a single sink vertex or a single cycle.

We consider the problem of finding switching cycles in G_M, later we will show that finding switching paths is as easy as finding switching cycles.

Given a directed pseudoforest G_P, we want to find each unique cycle C in each component of G_P. There could not be any cycle in a component of G_M if it is a tree component. The first approach is based on transitive closure G_P^* of G_P since computing the transitive closure is in NC by Theorem 4.4.2. We compute the transitive closure G_P^* and for any two vertices i and j s.t. $i \neq j$ in G_P, if $G_P^*(i,j) = 1$ and $G_P^*(j,i) = 1$, then both i and j are in the unique cycle C. Hence we can identify the cycle C by checking each pair of vertices in parallel.

Theorem 4.4.2 ([38]). *The transitive closure of a directed graph with n vertices can be computed in $O(\log^2 n)$ time, using $O(M(n)\log n)$ operations on a CREW PRAM, where $M(n)$ is the best known sequential bound for multiplying two $n \times n$ matrices over a ring.*

We also give NC algorithms in the setting of undirected graph in which transitive closure does not help. Given an undirected pseudoforest G_P, denote the incidence matrix of G_P as I_{G_P}. Let $cc(G)$ be the number of connected components in G. The basic idea is that we remove any one edge e from G_P, if $e \in C$ s.t. C is the unique cycle in G_P, then $cc(G_P - \{e\}) = cc(G_P)$; otherwise, $cc(G_P - \{e\}) = cc(G_P) + 1$. There is a direct connection between the rank of incidence matrix I of G and the number of connected component $cc(G)$ in G.

Lemma 4.4.3. *If G is an undirected graph with k connected components, then the rank of its incidence matrix I_G is $n - k$.*

So we can compute the rank of I_{G_P} and for each e in G_P, compute the rank of $I_{G_P - \{e\}}$ in parallel. There are at most $|V|$ edges in G_P.

Theorem 4.4.4 ([57])**.** *The rank of a $n \times n$ matrix over an arbitrary field can be computed in $O(\log^2 n)$ time, using a polynomial number of processors.*

We can also compute the number of connected component of G_P directly by finding all connected components in G_P.

Theorem 4.4.5 ([10])**.** *The connected components of a graph with n vertices and m edges can be computed in $O(\log n)$ time, using $O((m+n)\alpha(m,n)/\log n)$ operations on an ARBITRARY CRCW PRAM, where $\alpha(m,n)$ is the inverse Ackermann function.*

For any tree component T, there might be zero or multiple switching paths. For each s-post p, we make a copy of T and add one directed edge from the sink vertex to p and then find the unique cycle in the new graph, which yields one switching path in T.

4.4.2 Algorithmic Results

Now we are ready to give an NC algorithm to find a maximum-cardinality popular matching.

Algorithm 5: Maximum-Cardinality Popular Matching

Input: Reduced graph $G' = (\mathcal{A} \cup \mathcal{P}, E')$ and a popular matching M.
Output: A maximum-cardinality popular matching M'.
$G_M :=$ switching graph of M and G'.
Find all weakly connected components of G_M;
for each cycle component (resp. tree component) **in parallel do**
    Find the unique switching cycle (resp. each switching path);
for each switching cycle (resp. switching path) **in parallel do**
    Compute the margin of applying this switching cycle(resp. switching path);
for each cycle component (resp. tree component) **in parallel do**
    if the margin Δ of switching cycle (resp. the largest margin of switching paths) is positive
        Apply this switching cycle (resp. switching path) to M;
// The resulting matching M' after applying such switching cycles and switching paths is the maximum-cardinality matching.
return M';

Given the reduced graph G' and a popular matching M, we construct the switching graph G_M. After that, we identify the unique switching cycle or each switching path in G_M. Then we increase the size of popular matching locally according to the margin Δ of each component.

For each switching cycle C (resp. switching path P), we define the margin Δ in Definition 4.4.2 as the difference of the number of *last resort posts* after applying C (resp. P) to M. For each applicant a, the margin Δ increases by 1 if a promotes from $l(a)$ to $f(a)$ or decreases by 1 if a demotes from $f(a)$ to $l(a)$, otherwise no change is made. The following theorem gives a one-to-one correspondence between a popular matching and a unique subset of the cycle components and the tree components of G_M, which is crucial to our algorithm for maximum-cardinality popular matching.

Definition 4.4.2. Let Δ be the margin of applying a switching cycle C (resp. switching path P) to M, i.e.

$$\Delta = \sum_{a \in C(\text{resp.}P)} \mathbb{1}_{M \cdot C(a)} - \mathbb{1}_{M(a)}$$

where $\mathbb{1}_p$ is an indicator function of posts

$$\text{s.t. } \mathbb{1}_p := \begin{cases} 1 & \text{if } p \text{ is not } l\text{-post} \\ 0 & \text{if } p \text{ is } l\text{-post} \end{cases}$$

The following theorem is crucial for the correctness of Algorithm 5.

Theorem 4.4.6 ([56], Corollary 1)**.** *Let $G = (\mathcal{A} \cup \mathcal{P}, E)$ be an instance, and let M be an arbitrary popular matching for G with switching graph G_M. Let the tree components of G_M be $T_1, T_2, \cdots, T_k$, and the cycle components of G_M*

be $C_1, C_2, \cdots, C_l$. Then the set of popular matchings for G consists of exactly those matchings obtained by applying at most one switching path in T_i for each $i(1 \leq i \leq k)$ and by either applying or not applying the switching cycle in C_i for each $i(1 \leq i \leq l)$.

4.4.3 Correctness

Any popular matching can be obtained from M by applying at most one switching cycle or switching path per component of the switching graph G_M. For any tree component T, we apply the switching path in T with the largest positive margin. Similarly, for any cycle component C, we apply the switching cycle in C with positive margin. Then, we get the largest possible total margin, which in turn implies the largest possible number of l-posts we removed from M. Hence, we obtain the maximum-cardinality popular matching. For any other popular matching obtained by applying difference subset of switching paths or switching cycles, it will have strictly less total margin than the maximum-cardinality popular matching.

4.4.4 Complexity

It is clear that the switching graph G_M can be constructed from G' and M in constant time in parallel. All weakly connected components of G_M can also be found in polylog time by Theorem 4.4.5. Moreover, in Section 4.4.1, we showed that all switching cycles and switching paths can be found in polylog time. Each switching cycle and switching path can be applied to

matching M easily in parallel since they are vertex-disjoint in G_M. So, overall the complexity of Algorithm 5 is $O(\log^2 n)$.

We summarize the preceding discussion in the following theorem.

Theorem 4.4.7. *We can find a maximum-cardinality popular matching, or determine that no such matching exists in NC.*

4.4.5 Optimal Popular Matchings

It is natural to extend the popular matching problem to a weighted version of the popular matching problem. If a weight $w(a_i, p_j)$ is defined for each applicant-post pair with p_j acceptable to a_i, then the weight $w(M)$ of a popular matching M is $\sum_{(a_i,p_j)\in M} w(a_i, p_j)$. A popular matching is optimal if it is a maximum or minimum weight popular matching. It turns out that maximum-cardinality popular matching is a special case of maximum weight popular matching if we assign a weight of 0 to each pair involving a last resort post and a weight of 1 to all other pairs.

Kavitha et al. [41] considered other optimality criteria, in terms of the so called profile of the matching. For a popular matching instance with n_1 applicants and n_2 posts, we define the profile $\rho(M)$ of M to be the $(n_2 + 1)$ tuple $(x_1, x_2, \cdots, x_{n_2+1})$ such that for each $i, 1 \leq i \leq n_2 + 1$, x_i is the number of applicants who are matched with their ith ranked post. An applicant who is matched to his/her last resort post is considered to be matched to his/her $(n_2 + 1)$th ranked post, regardless of the length of his/her preference list.

Suppose that $\rho = (x_1, x_2, \cdots, x_{n_2+1})$ and $\rho' = (y_1, y_2, \cdots, y_{n_2+1})$. We use $\succ_R$ denote the lexicographic order on profiles: $\rho \succ_R \rho'$ if $x_i = y_i$ for $1 \leq i < k$ and $x_k > y_k$, for some k. Similarly, we use $\prec_F$ to denote the lexicographic order on profiles: $\rho \prec_F \rho'$ if $x_i = y_i$ for $k < i \leq n_2 + 1$ and $x_k < y_k$, for some k.

A rank-maximal popular matching is a popular matching whose profile is maximal with respect to $\succ_R$. A fair popular matching is a popular matching whose profile is minimal with respect to $\prec_F$. Note that a fair popular matching is always a maximum-cardinality popular matching since the number of last resort posts is minimized. It is easy to check these two problems are equivalent to the optimal popular matching problem with suitable weight assignments as follows.

- Rank-maximal popular matching: assign a weight of 0 to each pair involving a last resort post and a weight of $n_1^{n_2-k+1}$ to each pair (a_i, p_j) where p_j is kth ranked post of a_i, and find a maximum weight popular matching.

- Fair popular matching: assign a weight of n_1^k to each pair (a_i, p_j) where p_j is the kth ranked post of a_i, and find a minimum weight popular matching.

Now we are ready to give an NC algorithm for the optimal popular matching problem. Given a popular matching instance and a particular weight assignment, let M be a popular matching, and M_{opt} be an optimal popular

matching (maximum or minimum weight, depends on the context). By Theorem 4.4.6, M_{opt} can be obtained from M by applying a choice of at most one switching cycle or switching path per component of the switching graph G_M. Similar to Algorithm 5, the algorithm for computing M_{opt} will compute an arbitrary popular matching M, and make an appropriate choice of switching cycles and switching paths to apply in order to obtain an optimal popular matching. The only difference is the margin calculation. In order to decide to apply a switching cycle C or not, we need to compare $\sum_{a \in C} w(a, M(a))$ with $\sum_{a \in C} w(a, M \cdot C(a))$. In the case of maximum-cardinality popular matching, the weight assignment is either 0 or 1. While in rank-maximal popular matching and fair popular matching, w is bounded by $n_1^{n_2+1}$, which has $\tilde{O}(n)$ bits. So $\sum_{a \in C} w(a, M(a))$ can be computed in NC.

4.5 Preference Lists with Ties

In this section, we consider the popular matching problem such that preference lists are not strictly ordered, but contain ties. Without the assumption of strictly ordered preference lists, we show that the popular matching problem is at least as hard as the maximum-cardinality bipartite matching problem by showing that maximum-cardinality bipartite matching is NC-reducible to popular matching. Note that whether bipartite perfect matching is in NC is still open [17].

Now we show the following NC reduction.

Theorem 4.5.1. *Maximum-cardinality Bipartite Matching* $\leq_{NC}$ *Popular Matching.*

Proof. Suppose we have access to a black box that can solve Popular Matching in NC. Consider an arbitrary instance of Maximum-cardinality Bipartite Matching, specified by a graph $G = (\mathcal{A} \cup \mathcal{B}, E)$. We construct our Popular Matching instance by giving all edges rank 1, i.e. each applicant has the same preference over all acceptable posts. For convenience, we also use $G = (\mathcal{A} \cup \mathcal{B}, E)$ as our instance of Popular Matching. We do not add last resort posts at all. Lemma 4.5.2 and Lemma 4.5.3 show that popular matching always exists in G and any popular matching M is also a maximum-cardinality matching in G. □

Lemma 4.5.2. *Let* M *be a popular matching in* G*. Then* M *is also a maximum-cardinality matching in* G*.*

Proof. Suppose for a contradiction that M is not a maximum matching of G. Then M admits an augmenting path Q with respect to G. Since each edge in G has rank 1, after applying augmenting path Q to M, we obtain a matching M' that is *more popular than* M because M' has exactly one more edge matched than M and all rest of applicants do not have preference over M and M'. □

We know from Section 4.3 that popular matching may not exist in an arbitrary popular matching instance. We show that given the construction that each edge in G has rank 1, popular matching always exists.

Lemma 4.5.3. *Let M be a maximum-cardinality matching in G. Then M is also a popular matching in G.*

Proof. Consider any other matching M' in G. We only care about the symmetric difference $M\Delta M'$ since the rest of edges do not have preference over M and M'. Since all edges have rank 1, then $|P(M', M)|-|P(M, M')|= |M'|-|M|\leq 0$. Hence, no matching is *more popular than* M. □

We conjecture that the following reduction is also true.

Conjecture 4.5.4. *Popular Matching $\leq_{NC}$ Maximum-cardinality Bipartite Matching.*

4.6 Finding "next" Stable Matching in NC

In this section, we consider the problem of finding the "next" stable matching. Irving and Gusfield's book [25] mentioned that even if it is not possible to find the first stable matching fast in parallel, perhaps, after sufficient preprocessing, the stable matchings could be enumerated in parallel, with small parallel time per matching. Our results partially answer this question, given a stable matching, we can enumerate the "next" stable matching in the stable matching lattice in polylog time. This result can be regarded as an application of the techniques used in 4.4, that is to find cycles in pseudoforest in NC. The main result is given by Theorem 4.6.2.

We give some useful definitions in the next section.

4.6.1 The Stable Marriage Problem

We first give some basic results of the stable marriage problem. Definitions of the stable marriage problem is given in Chapter 1.

Definition 4.6.1 (Partial Order $\mathcal{M}$)**.** For a given stable marriage instance, stable matching M is said to dominate stable matching M', written $M \preceq M'$, if every man either prefers M to M' or is indifferent between them. We use the term strictly dominate, written $M \prec M'$, if $M \preceq M'$ and $M \neq M'$. We use the symbol $\mathcal{M}$ to represent the set of all stable matchings for a stable marriage instance. Then the set $\mathcal{M}$ is a partial order under the dominance relation, denoted by $(\mathcal{M}, \preceq)$.

It is well-known that the partial order $(\mathcal{M}, \preceq)$ forms a distributive lattice. Hence, the unique minimal element in $\mathcal{M}$ with respect to $\preceq$, i.e. man-optimal stable matching (denoted by M_0), as well the unique maximal element, i.e. woman-optimal stable matching (denoted by M_z) is well-defined.

Definition 4.6.2 (Rotation)**.** Let $k \geq 2$. A rotation ρ is an ordered list of pairs

$$\rho = ((m_0, w_0), (m_1, w_1), \cdots, (m_{k-1}, w_{k-1}))$$

that are matched in some stable matching M with the property that for every i such that $0 \leq i \leq k-1$, woman w_{i+1} (where $i+1$ is taken modulo k) is the highest ranked woman on m_i's preference list satisfying:

(i) man m_i prefers w_i to w_{i+1}, and

(ii) woman w_{i+1} prefers m_i to m_{i+1}.

In this case, we say ρ is exposed in M.

Definition 4.6.3 (Elimination of a Rotation)**.** Let $\rho = ((m_0, w_0)$ $, (m_1, w_1), \cdots, (m_{k-1}, w_{k-1}))$ be a rotation exposed in a stable matching M. The rotation ρ is eliminated from M by matching m_i to $w_{(i+1) \bmod k}$ for all $0 \leq i \leq k-1$, leaving all other pairs in M unchanged, i.e. matching M is replaced by matching M', where

$$M' := M \backslash \rho \ \cup \ \{(m_0, w_1), (m_1, w_2), \cdots, (m_{k-1}, w_0)\}.$$

Note that the resulting matching M' is also stable.

Lemma 4.6.1 ([25], Theorem 2.5.1)**.** *If ρ is exposed in M, then M immediately dominates $M\backslash\rho$, i.e. there is no stable matching M' such that $M \prec M'$ and $M' \prec M\backslash\rho$.*

Theorem 4.6.2. *Given a stable matching M, there is an NC algorithm that outputs stable matching $M\backslash\rho$ for each rotation ρ exposed in M or determines M is the woman-optimal matching.*

4.6.2 Algorithmic Results

We describe an NC algorithm to find the "next" stable matching in this section.

Let M be a stable matching. For any man m, let $s_M(m)$ denote the highest ranked woman on m's preference list such that w prefers m to $p_M(w)$.

Algorithm 6: "next" Stable Matching

```
1  Input: Stable matching M and preference lists mp and wp.
2  Output: M\ρ or determine M is the woman-optimal matching.
3  Compute ranking matrices mr and wr; // constant steps
4  Compute reduced preference lists mp' and wp'; // logarithmic
     number of steps
5  Construct H_M from mp';
6  if H_M is not empty then
7      Find all simple cycles(rotations) in H_M;
8      for each rotation ρ in H_M in parallel do
9          return M\ρ;
10 else
11     return M is the woman-optimal matching;
```

Let $next_M(m)$ denote woman $s_M(m)$'s partner in M. Note that since M is stable, m prefers $p_M(m)$ to $s_M(m)$.

Now let m be any man who has different partners in M and M_z and let w be m's partner in M_z. Since M_z is woman-optimal, m prefers $p_M(m)$ to w and w prefers m to $p_M(w)$. Hence, $s_M(m)$ exists. If $s_M(m)$ exists and $m' = next_M(m)$, then $s_M(m')$ exists as well. Otherwise, m' and $s_M(m)$ are partners in M_z, so m prefers $s_M(m)$ to his partner w in M_z and $s_M(m)$ prefers m to her partner m' in M_z, contradicting the stability of M_z. Denote D the set of man who has different partners in M and M_z, then for any man $m \in D$, $next_M(m) \in D$. Later we will show that the algorithm does need to know M_z.

Similar to the *switching graph* of popular matching, we define the *switching graph* of stable matching M as a directed graph H_M with a vertex for each man in D and a directed edge from the vertex for m to the vertex

for $next_M(m)$, which is also in H_M. Some simple properties of *switching graph* H_M are shown in the following lemma.

Lemma 4.6.3. *Let M be a stable matching other than the woman-optimal matching M_z, let H_M be the switching graph of M, then*

(i) Each vertex in H_M has outdegree exactly one.

(ii) Each component of H_M contains a single simple cycle.

Proof. (i) is direct from the definition of a switching graph. (ii) No vertex points to itself, so there is no self loop in H_M. If there is no cycle in one component, then there exists at least one sink vertex (consider the topological sort of H_M), contradicting (i). If there are two cycles in one component, consider any path that connects these two cycles. There must be a vertex with at least two outgoing edges again contradicting (i). □

From Definition 4.6.2, it is easy to see that any such simple cycle defines the men in a rotation exposed in M, in the order that they appear in the rotation. On the other hand, based on the uniqueness of $next_M(m)$ for each $m \in D$, if m belongs to some rotation ρ, e.g. $m = m_i$, then m_{i+1} is uniquely determined, that is $next_M(m)$. Hence the men in ρ must be a simple cycle in H_M.

We know from Section 4.4.1 that every cycle in H_M can be found in NC. It is obvious that the elimination of a rotation can be done in one parallel step. Thus, we are left to show that H_M can be constructed in NC.

Let us assume that a stable marriage instance is described by the sets of preference lists, represented as matrices mp and wp defined by

- $mp[m, i] = w$ if woman w is ranked of i in m's preference list
- $wp[w, i] = m$ if man m is ranked of i in w's preference list

We also define the ranking matrices mr and wr as below

- $mr[m, w] = i$ if woman w is ranked of i in m's preference list
- $wr[w, m] = i$ if man m is ranked of i in w's preference list

We need to identify $s_M(m)$ and $next_M(m)$ for each man m. Suppose for each woman w we delete all pairs (m', w) such that w prefers $p_M(w)$ to m'. In the resulting preference lists, which we call reduced lists, $p_M(w)$ is the last entry in w's list, and $p_M(m)$ is the first entry in m's list for if any woman w' remains above $p_M(m)$, then (m, w') blocks M. Moreover, $s_M(m)$ is the second entry in m's list if exists, for by definition, it is the highest ranked woman w on m's list such that w prefers m to $p_M(w)$. $next_M(m)$ is simply the partner in M of woman $s_M(m)$.

From the algorithmic aspect, for each entry (m, w) to be deleted in parallel, we call the ranking matrix mr to obtain woman w's rank on m's list. Then call the preference matrix mp and use soft-deletion, i.e. mark the entry $mp[m, mr[m, w]]$ zero. After each entry is soft-deleted, we can compress the preference list using parallel prefix sum technique. The resulting preference

lists are reduced lists. Now we obtain all pairs $(m, next_M(m))$ and it is easy to construct H_M.

4.6.3 Example of Stable Matchings

Figure 4.5 is an example of a stable marriage instance. The reader can verify that the matching M denoted by underlining is stable.

m_1 :	w_5	w_7	w_1	w_2	w_6	$\underline{w_8}$	w_4	$w_3\,w_1$:	m_5	m_3	m_7	$\underline{m_6}$	m_1	m_2	m_8	m_4
m_2 :	w_2	$\underline{w_3}$	w_7	w_5	w_4	w_1	w_8	$w_6\,w_2$:	m_8	m_6	m_3	m_5	$\underline{m_7}$	m_2	m_1	m_4
m_3 :	w_8	$\underline{w_5}$	w_1	w_4	w_6	w_2	w_3	$w_7\,w_3$:	m_1	m_5	m_6	$\underline{m_2}$	m_4	m_8	m_7	m_3
m_4 :	w_3	w_2	w_7	w_4	w_1	$\underline{w_6}$	w_8	$w_5\,w_4$:	$\underline{m_8}$	m_7	m_3	m_2	m_4	m_1	m_5	m_6
m_5 :	$\underline{w_7}$	w_2	w_5	w_1	w_3	w_6	w_8	$w_4\,w_5$:	m_6	m_4	m_7	$\underline{m_3}$	m_8	m_1	m_2	m_5
m_6 :	$\underline{w_1}$	w_6	w_7	w_5	w_8	w_4	w_2	$w_3\,w_6$:	m_2	m_8	m_5	m_3	$\underline{m_4}$	m_6	m_7	m_1
m_7 :	$\underline{w_2}$	w_5	w_7	w_6	w_3	w_4	w_8	$w_1\,w_7$:	m_7	$\underline{m_5}$	m_2	m_1	m_8	m_6	m_4	m_3
m_8 :	w_3	w_8	$\underline{w_4}$	w_5	w_7	w_2	w_6	$w_1\,w_8$:	m_7	m_4	$\underline{m_1}$	m_5	m_2	m_3	m_6	m_8

(a) Men's preferences (b) Women's preferences

Figure 4.5: The stable marriage instance of size 8 and the stable matching M denoted by underlining

Figure 4.6 shows the reduced lists of the men for the stable matching M. The second column corresponds to $s_M(m)$ for each m.

Finally we give the switching graph H_M for the stable matching M in Figure 4.7.

4.7 Conclusion and Open Problems

In this chapter, we established that the popular matching problem without ties is in NC. The notion of pseudoforest may have other applications for

m_1 : $\underline{w_8}$ w_3

m_2 : $\underline{w_3}$ w_6

m_3 : $\underline{w_5}$ w_1 w_6 w_2

m_4 : $\underline{w_6}$ w_8 w_5

m_5 : $\underline{w_7}$ w_2 w_1 w_3 w_6

m_6 : $\underline{w_1}$ w_5 w_2 w_3

m_7 : $\underline{w_2}$ w_5 w_7 w_8 w_1

m_8 : $\underline{w_4}$ w_2 w_6

Figure 4.6: The reduced lists of the men for the stable matching M

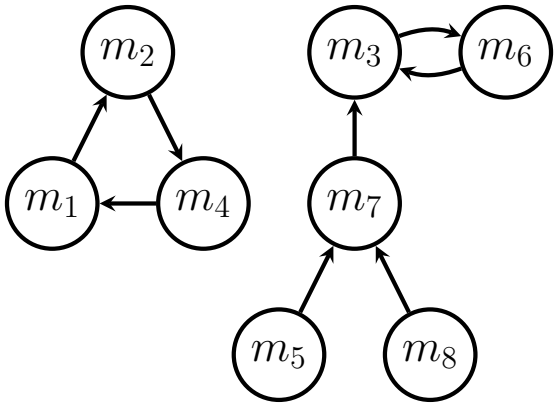


Figure 4.7: The switching graph H_M

designing parallel algorithms.

We showed that maximum-cardinality bipartite matching is NC-reducible to popular matching. One open problem is Conjecture 4.5.4. If it is true, then it means these two problems are NC-equivalent. The other open problem is establishing the NC reduction among several other matching problems in preference systems such as Pareto-optimal matching and rank-maximal matching.

To find an NC algorithm for the stable matching problem seems hard

given that the stable matching problem is CC-complete. It may be easier to give an RNC algorithm for the stable matching problem. It is also interesting to study some other variants of the stable matching problem, e.g. stable matching instance with bounded-length preference lists, etc.

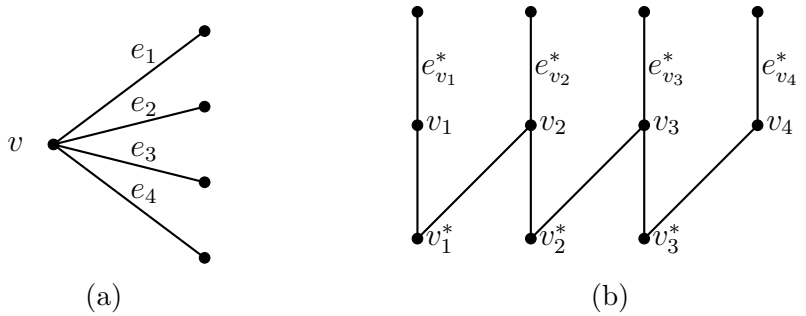
We initialize some works in the following:

4.7.1 Reduction to stable matching instance with bounded-degree

Assume that we are given a bipartite preference system $P = (V, E, <_v)$, i.e. a bipartite graph (V, E) and a set of strict total order $<_v$ for vertices $v \in V$. For each vertex $v \in V$, we denote by $E(v)$ the set of edges that are incident to v. We also define that $deg(P) := max_{v \in V} |E(v)|$.

We construct a new preference system $Q = (W, F, \prec_v)$ from P such that $deg(Q) \leq 3$ (See Figure 4.8). For each vertex $v \in V$, we make $2|E(v)|-1$ copies and partition into two groups $\{v_i | i \in \{1, 2, \cdots, |E(v)|\}\}$ and $\{v_i^* | i \in \{1, 2, \cdots, |E(v)|-1\}\}$. For each vertex $v \in V$ and each edge $e \in E(v)$, we define the rank of e in v as $rank_v(e) = 1 + |\{f \in E(v) | f <_v e\}|$. For each edge $e = (u, v) \in E$, we define $e^* = (u_{rank_u(e)}, v_{rank_v(e)})$ and the set of these edges as E^*. Notice that E and E^* is a one-to-one mapping. We also include the set of edges (v_i, v_i^*) and (v_i^*, v_{i+1}) for each $i \in \{1, 2, \cdots, |E(v)|-1\}$.

Next, we define the strict total order $\prec_v$ for each $v \in W$. For each vertex v in V and each integer i in $\{1, 2, \cdots, |E(v)|\}$, we denote by $e^*_{v_i}$ the unique edge e in E^* that is incident to v_i. We define the strict total order of vertex $v_i \in W$ for each $v \in V$ and each integer $i \in \{1, 2, \cdots, |E(v)|\}$ as follows:



(a) (b)

Figure 4.8: (a) A vertex v and the edges contains v. We assume that $e_1 <_v e_2 <_v e_3 <_v e_4$. (b) for each integer i in $\{1, 2, 3, 4\}$, we have $e^*_{v_i} = e^*_i$.

$$e^*_{v_1} \prec_{v_1} (v_1, v^*_1)$$

$$(v^*_{i-1}, v_i) \prec_{v_i} e^*_{v_i} \prec_{v_i} (v_i, v^*_i) \text{ for } 2 \leq i \leq |E(v)|-1$$

$$(v^*_{|E(v)|-1}, v_{|E(v)|}) \prec_{v_{|E(v)|}} e^*_{v_{|E(v)|}}$$

Then for each v in V and each integer i in $\{1, 2, \cdots, |E(v)|-1\}$, we define that $(v_i, v^*_i) \prec_{v^*_i} (v^*_i, v_{i+1})$.

We prove that we can construct a stable matching in P from a stable matching in Q. Suppose that we are given a stable matching M^* in Q.

Lemma 4.7.1. *For each vertex v in V and each integer i in $\{1, 2, \cdots, |E(v)|-1\}$, v^*_i is matched in M^*.*

Proof. Suppose that v^*_i is unmatched, then v_{i+1} must be matched in M^*. Otherwise we can add (v^*_i, v_{i+1}) into M^*. However, v^*_i is the best choice of v_{i+1}, i.e. $rank_{v_{i+1}}((v^*_i, v_{i+1})) = 1$. Hence, (v^*_i, v_{i+1}) blocks M^*. □

Proposition 4.7.2. *For each vertex v in V and each integer i in $\{1, 2, \cdots, |E(v)|\}$, exactly one of v_i is either unmatched in M^* or matched in $e^*_{v_i}$.*

Proof. Obvious by Lemma 4.7.1. □

Given a stable matching M^* in Q, if the edge (v_i, u_j) is in M^* for some vertices v, u in V and some integers i, j, then we include the edge (v, u) into M. We need to prove that M is a stable matching in P.

Lemma 4.7.3. *The set of edges M in P is a matching.*

Proof. Suppose that M is not a matching, then there exists a vertex v in V such that v has two neighbors u and w in M. Thus there exists integers i and j, such that v_i matches to $e^*_{v_i}$ and v_j matches to $e^*_{v_j}$ in $M*$, which contradicts with Proposition 4.7.2. □

Lemma 4.7.4. *If $e^* = (v_i, u_j)$ for some vertices v, u and integers i, j is unmatched in M^* and v_i is matched to v^*_{i-1}, then there exists v_k, $k < i$, such that $e^*_{v_k}$ is in M^*.*

Proof. We prove by induction. Base case: v_2 is matched with v^*_1. If $e^*_{v_1}$ is unmatched in M^*, then v_1 is unmatched. But (v_1, v^*_1) is a blocking pair. If v_{i+1} is matched to v^*_i, we have 3 cases. Case 1: v_i is unmatched, then (v_i, v^*_i) is a blocking pair; Case 2: $e^*_{v_i}$ is matched, we have the required edge $e^*_{v_i}$ in M^* and $i < i + 1$; Case 3: (v_i, v^*_{i-1}) is matched, by induction, we have if v_i is matched to v^*_{i-1}, then there exists v_k, $k < i$, such that $e^*_{v_k}$ is in M^*. □

Lemma 4.7.5. *The set of edges M is a stable matching.*

Proof. M is a matching by Lemma 4.7.3. Next we prove that there is no blocking pair in M. This is equivalent to say for any edge $e = (v, u)$ in $E \backslash M$, at least one of v and u, w.l.o.g, let it be v, such that v is matched in M and $(v, M(v)) <_v (v, u)$.

We know that M^* is a stable matching in P. For each unmatched edge e in E, there is a corresponding unmatched edge e^* in F. Since M^* is stable, there must exist some vertex $v_i \in e^*$ for some v in V and integer i such that v_i is matched in M^* and $(v_i, M^*(v_i)) \prec_{v_i} e^*$. Since we know that $(v^*_{i-1}, v_i) \prec_{v_i} e^*_{v_i} \prec_{v_i} (v_i, v^*_i)$, then $M^*(v_i) = v^*_{i-1}$. By lemma 4.7.4, we know that there exists v_k, $k < i$, such that $e^*_{v_k}$ is in M^*. Hence v is matched in M and $(M(v), v) <_v e$.

□

We proved that given a stable matching in Q such that Q has degree at most 3, we can recover a stable matching in P from Q. It is natural to ask that if it is possible to recover an almost stable matching in P given an almost stable matching in Q. There is no consensus in the literature on precisely how to measure almost-stability, but typically almost-stability requires that a matching induces relatively few blocking pairs. Ostrovsky and Rosenbaum [60] gives a deterministic distributed algorithm which finds an almost stable matching in $O(\log^5 n)$ communication rounds for arbitrary preferences. The running time can be improved to $O(1)$ rounds for "almost regular" preferences.

It would be interesting to consider the almost stable matching problem such that the degree of the preference lists is at most 3 in a distributed manner.

Chapter 5

Minimal Envy Matchings in the Hospitals/Residents Problem with Lower Quotas

5.1 Introduction

In the Hospitals/Residents problem, every hospital has an upper quota that limits the number of residents assigned to it. While, in some applications, each hospital also has a lower quota for the number of residents it receives. This extension of the HR problem is referred to as the Hospitals/Residents problem with Lower Quotas (HRLQ, for short) in the literature. However, the existence of a stable matching is not always true for a given HRLQ instance. This can be easily observed by the well-known Rural Hospitals theorem [21, 66, 67] stating that each hospital is assigned the same number of residents in any stable matching for a given HR instance.

Since there might be no stable matching given a HRLQ instance, one natural approach is to weaken the requirement of stability. Envy-freeness is a relaxation of stability that allows blocking pairs involving a resident and an empty position of a hospital. Structural results of envy-free matchings were investigated in [79] showing that the set of envy-free matchings forms a lattice. Envy-free matchings with lower quotas has recently been studied in

[80]. Fragiadalis et al. [19] studies envy-free matchings called fairness in their paper when the preference lists are complete and the sum of lower quotas of all hospitals does not exceed the number of residents. In this restricted setting, envy-free matching always exists and they gave a linear-time algorithm called extended-seat deferred acceptance (ESDA) to find an envy-free matching. However, if preference lists are not complete, envy-free matching may not exist. Yokoi [80] provided a characterization of envy-free matchings in HRLQ, connecting them to stable matchings in a modified HR instance so that the existence of envy-free matching can be decided by running DA on this modified HR instance.

Given a HRLQ instance, we know that stable matchings or envy-free matchings might not exist. Hamada et al. [27] considered the problem of minimizing the number of blocking pairs among all feasible matchings (feasible matching means a matching satisfies both lower and upper quotas of each hospital). They showed hardness of approximation of the problem and provided an exponential-time exact algorithm.

In this chapter, we consider the problem of minimizing envy (defined later) among all feasible matchings given a HRLQ instance. Given a feasible matching, the envy among all residents can be measured by the number of envy-pairs or the number of residents involved in any envy-pair.

We show that for both measurements, the problem is NP-hard. We also provide an exponential-time algorithm to find a feasible matching minimizing the number of envy-pairs.

The results presented in this chapter are joint work with Vijay Garg and published in [30].

5.1.1 Related Work

Popular matching is another relaxation of stability which preserves "global" stability in the sense that no majority of residents wish to alter into another feasible matching. Popular matching always exists. Nasre and Nimbhorkar [59] proposed an efficient algorithm to compute a maximum cardinality matching that is popular amongst all the feasible matchings in an HRLQ instance. When there exists a stable matching given a HRLQ instance, it is known that every stable matching is popular.

5.2 Preliminaries

An instance $I = (G, \succ)$ of the Hospitals/Residents Problem with Lower Quotas (HRLQ for short) consists of a bipartite graph $G = (R \cup H, E)$, where R is a set of residents and H is a set of hospitals, and an edge $(r, h) \in E$ denotes that r and h are mutually acceptable, and a preference system $\succ$ such that every vertex (resident and hospital) in G ranks its neighbors in a strict order, referred to as the preference list of the vertex. If a vertex a prefers its neighbor b_1 over b_2, we denote it by $b_1 \succ_a b_2$. Each hospital h has a lower quota l_h and an upper quota u_h ($0 \leq l_h \leq u_h$). Sometimes we write $[l_h, u_h]$ to denote the lower and upper quota for a hospital h.

A matching $M \subseteq E$ in G is an assignment of residents to hospitals such

that each resident is matched to at most one hospital, and every hospital h is matched to at most u_h residents. Let $M(r)$ denote the hospital that r is matched in M. If r is unmatched in M, we let $M(r) = \emptyset$. For any neighbor h of r, we have $h \succ_r \emptyset$ since we assume that any resident prefers to be matched over to be unmatched. We say that a hospital h is under-subscribed in M if $|M(h)| < u_h$, h is fully-subscribed if $|M(h)| = u_h$, h is over-subscribed if $|M(h)| > u_h$ and h is deficient if $|M(h)| < l_h$. A matching M is feasible in a HRLQ instance if no hospital is deficient or over-subscribed in M. The HRLQ problem is to match residents to hospitals under some optimality condition such that the matching is feasible. Envy-freeness is defined as follows.

Definition 5.2.1. Given a matching M, a resident-hospital pair (r, h) is an *envy-pair* if $h \succ_r M(r)$ and $r \succ_h r'$ for some $r' \in M(h)$.

Definition 5.2.2. Given a matching M, a resident r has *justified envy* toward r' who is matched to hospital h if $h \succ_r M(r)$ and $r \succ_h r'$.

Definition 5.2.3. A matching M is *envy-free* if there is no *envy-pair* in M. Equivalently, a matching M is envy-free if no resident has *justified envy* toward other residents in M.

In case that envy-free matchings may not exist in a given HRLQ instance. We define two other problems that minimize envy in terms of envy-pairs and envy-residents.

Minimum-Envy-Pair Hospitals/Residents Problem with Lower Quotas (Min-EP HRLQ for short) is the problem of finding a feasible match-

ing with the minimum number of envy-pairs. 0-1 Min-EP HRLQ is the restriction of Min-EP HRLQ where a quota of each hospital is either $[0, 1]$ or $[1, 1]$.

Definition 5.2.4. Given a matching M, a resident r is an envy-resident if there exists h such that $(r, h) \in E$ is an envy-pair in M.

Minimum-Envy-Resident Hospitals/Residents Problem with Lower Quotas (Min-ER HRLQ for short) is the problem of finding a feasible matching with minimum number of envy-residents. 0-1 Min-ER HRLQ is defined similarly.

We assume without loss of generality that the number of residents is at least the sum of the lower quotas of all hospitals, since otherwise there is no feasible matching. In other papers, they impose the Complete List restriction (CL-restriction for short). There always exists envy-free matchings in CL-restriction instances, and a maximum-size one can be found in polynomial time [19, 45].

5.3 Minimum-Envy-Pair HRLQ

In this section, we consider the problem of minimizing the number of envy-pairs in HRLQ. We prove a NP-hardness result for 0-1 Min-EP HRLQ in the following theorem.

Theorem 5.3.1. *0-1 Min-EP HRLQ is NP-hard.*

Proof. We give a polynomial-time reduction from the well-known NP-complete problem Vertex Cover. Below is the definition of the decision version of the Vertex Cover problem. Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, we are asked if there is a subset $C \subseteq V$ such that $|C| \leq K$, which contains at least one endpoint of each edge of G.

Reduction: Given a graph $G = (V, E)$ and a positive integer K, which is an instance of the Vertex Cover problem, we construct an instance I of 0-1 Min-EP HRLQ. Define $n = |V|$, $m = |E|$ and $l = n^2 + 1$. The set of residents is $R = C \cup F \cup S$ and the set of hospitals is $H = V \cup T$. Each set is defined as follows:

$$
\begin{aligned}
C &= \{c_i \mid 1 \leq i \leq K\} \\
F &= \{f_i \mid 1 \leq i \leq n - K\} \\
S^{i,j} &= \{s^{i,j}_{0,a} \mid 1 \leq a \leq l\} \cup \{s^{i,j}_{1,a} \mid 1 \leq a \leq l\} \ ((v_i, v_j) \in E, i < j) \\
S &= \bigcup S^{i,j} \\
V &= \{v_i \mid 1 \leq i \leq n\} \\
T^{i,j} &= \{t^{i,j}_{0,a} \mid 1 \leq a \leq l\} \cup \{t^{i,j}_{1,a} \mid 1 \leq a \leq l\} \ ((v_i, v_j) \in E, i < j) \\
T &= \bigcup T^{i,j}
\end{aligned}
$$

Each hospital in $H = V \cup T$ has a quota $[1, 1]$. Note that $|C|+|F|= |V|= n$ and $|S|= |T|= 2ml$. Thus $|H|+|R|= 2n + 4ml$, which is polynomial in n and m.

Next, we construct the preference lists of I. The preference lists of residents is shown as follows:

$$
\begin{aligned}
c_i &: [[V]] && (1 \leq i \leq K) \\
f_i &: [[V]] && (1 \leq i \leq n-K) \\
s^{i,j}_{0,1} &: t^{i,j}_{0,1} \quad v_i \quad t^{i,j}_{1,1} && ((v_i, v_j) \in E, i < j) \\
s^{i,j}_{0,2} &: t^{i,j}_{0,2} \quad v_i \quad t^{i,j}_{0,3} && ((v_i, v_j) \in E, i < j) \\
&\vdots \\
s^{i,j}_{0,l-1} &: t^{i,j}_{0,l-1} \quad v_i \quad t^{i,j}_{0,l} && ((v_i, v_j) \in E, i < j) \\
s^{i,j}_{0,l} &: t^{i,j}_{0,l} \quad v_i \quad t^{i,j}_{0,1} && ((v_i, v_j) \in E, i < j) \\
s^{i,j}_{1,1} &: t^{i,j}_{0,2} \quad v_j \quad t^{i,j}_{1,2} && ((v_i, v_j) \in E, i < j) \\
s^{i,j}_{1,2} &: t^{i,j}_{1,2} \quad v_j \quad t^{i,j}_{1,3} && ((v_i, v_j) \in E, i < j) \\
&\vdots \\
s^{i,j}_{1,l-1} &: t^{i,j}_{1,l-1} \quad v_j \quad t^{i,j}_{1,l} && ((v_i, v_j) \in E, i < j) \\
s^{i,j}_{1,l} &: t^{i,j}_{1,l} \quad v_j \quad t^{i,j}_{1,1} && ((v_i, v_j) \in E, i < j)
\end{aligned}
$$

, where $[[V]]$ denotes a fixed order of elements in V in an increasing order of indices.

The preference lists of hospitals is shown as follows:

$$
\begin{aligned}
v_i &: [[C]] \quad [[S_i]] \quad [[F]] && (1 \le i \le n) \\
t^{i,j}_{0,1} &: s^{i,j}_{0,1} \quad s^{i,j}_{0,l} && ((v_i, v_j) \in E, i < j) \\
t^{i,j}_{0,2} &: s^{i,j}_{1,1} \quad s^{i,j}_{0,2} && ((v_i, v_j) \in E, i < j) \\
&\vdots \\
t^{i,j}_{0,l-1} &: s^{i,j}_{0,l-2} \quad s^{i,j}_{0,l-1} && ((v_i, v_j) \in E, i < j) \\
t^{i,j}_{0,l} &: s^{i,j}_{0,l-1} \quad s^{i,j}_{0,l} && ((v_i, v_j) \in E, i < j) \\
t^{i,j}_{1,1} &: s^{i,j}_{0,1} \quad s^{i,j}_{1,l} && ((v_i, v_j) \in E, i < j) \\
t^{i,j}_{1,2} &: s^{i,j}_{1,1} \quad s^{i,j}_{1,2} && ((v_i, v_j) \in E, i < j) \\
&\vdots \\
t^{i,j}_{1,l-1} &: s^{i,j}_{1,l-2} \quad s^{i,j}_{1,l-1} && ((v_i, v_j) \in E, i < j) \\
t^{i,j}_{1,l} &: s^{i,j}_{1,l-1} \quad s^{i,j}_{1,l} && ((v_i, v_j) \in E, i < j)
\end{aligned}
$$

, where $[[C]]$ and $[[F]]$ are as before a fixed order of all the residents in C and F, respectively, in an increasing order of indices, $[[S_i]]$ is an arbitrary order of all the residents in S that are acceptable to v_i as determined by the preference lists of residents.

Since each hospital in $V \cup T$ has a quota $[1, 1]$, any feasible matching must be a perfect matching in I. Further, the subgraph between $C \cup F$ and V is a complete graph and there is no edge between $C \cup F$ and T in I. Thus, any feasible matching must include a perfect matching between $C \cup F$ and V.

For each edge $(v_i, v_j) \in E$ $(i < j)$, there are the set of residents $S^{i,j}$

and the set of hospitals $T^{i,j}$. We call this pair of sets a $g_{i,j}$-gadget, and write it as $g_{i,j} = (S^{i,j}, T^{i,j})$. For each gadget $g_{i,j}$, let us define two perfect matchings between $S^{i,j}$ and $T^{i,j}$ as follows:

$$M_0^{i,j} = \{(s_{0,1}^{i,j}, t_{0,1}^{i,j}), (s_{0,2}^{i,j}, t_{0,2}^{i,j}), \cdots, (s_{0,l-1}^{i,j}, t_{0,l-1}^{i,j}), (s_{0,l}^{i,j}, t_{0,l}^{i,j}),$$
$$(s_{1,1}^{i,j}, t_{1,2}^{i,j}), (s_{1,2}^{i,j}, t_{1,3}^{i,j}), \cdots, (s_{1,l-1}^{i,j}, t_{1,l}^{i,j}), (s_{1,l}^{i,j}, t_{1,1}^{i,j})\}, and$$
$$M_1^{i,j} = \{(s_{0,1}^{i,j}, t_{1,1}^{i,j}), (s_{0,2}^{i,j}, t_{0,3}^{i,j}), \cdots, (s_{0,l-1}^{i,j}, t_{0,l}^{i,j}), (s_{0,l}^{i,j}, t_{0,1}^{i,j}),$$
$$(s_{1,1}^{i,j}, t_{0,2}^{i,j}), (s_{1,2}^{i,j}, t_{1,2}^{i,j}), \cdots, (s_{1,l-1}^{i,j}, t_{1,l-1}^{i,j}), (s_{1,l}^{i,j}, t_{1,l}^{i,j})\}$$

Figure 5.1 shows $M_0^{i,j}$ and $M_1^{i,j}$ on preference lists of $S^{i,j}$.

$s_{0,1}^{i,j}$	:	$t_{0,1}^{i,j}$	v_i	$t_{1,1}^{i,j}$		$s_{0,1}^{i,j}$	:	$t_{0,1}^{i,j}$	v_i	$t_{1,1}^{i,j}$
$s_{0,2}^{i,j}$	:	$t_{0,2}^{i,j}$	v_i	$t_{0,3}^{i,j}$		$s_{0,2}^{i,j}$	:	$t_{0,2}^{i,j}$	v_i	$t_{0,3}^{i,j}$
			$\vdots$						$\vdots$	
$s_{0,l-1}^{i,j}$	:	$t_{0,l-1}^{i,j}$	v_i	$t_{0,l}^{i,j}$		$s_{0,l-1}^{i,j}$	:	$t_{0,l-1}^{i,j}$	v_i	$t_{0,l}^{i,j}$
$s_{0,l}^{i,j}$	:	$t_{0,l}^{i,j}$	v_i	$t_{0,1}^{i,j}$		$s_{0,l}^{i,j}$	:	$t_{0,l}^{i,j}$	v_i	$t_{0,1}^{i,j}$
$s_{1,1}^{i,j}$	:	$t_{0,2}^{i,j}$	v_j	$t_{1,2}^{i,j}$		$s_{1,1}^{i,j}$	:	$t_{0,2}^{i,j}$	v_j	$t_{1,2}^{i,j}$
$s_{1,2}^{i,j}$	:	$t_{1,2}^{i,j}$	v_j	$t_{1,3}^{i,j}$		$s_{1,2}^{i,j}$	:	$t_{1,2}^{i,j}$	v_j	$t_{1,3}^{i,j}$
			$\vdots$						$\vdots$	
$s_{1,l-1}^{i,j}$	:	$t_{1,l-1}^{i,j}$	v_j	$t_{1,l}^{i,j}$		$s_{1,l-1}^{i,j}$	:	$t_{1,l-1}^{i,j}$	v_j	$t_{1,l}^{i,j}$
$s_{1,l}^{i,j}$	:	$t_{1,l}^{i,j}$	v_j	$t_{1,1}^{i,j}$		$s_{1,l}^{i,j}$	:	$t_{1,l}^{i,j}$	v_j	$t_{1,1}^{i,j}$

Figure 5.1: Matchings $M_0^{i,j}$ (left) and $M_1^{i,j}$ (right)

Lemma 5.3.2. *For a gadget $g_{i,j} = (S^{i,j}, T^{i,j})$, $M_0^{i,j}$ and $M_1^{i,j}$ are the only perfect matchings between $S^{i,j}$ and $T^{i,j}$. Furthermore, each $M_0^{i,j}$ and $M_1^{i,j}$ contains an unique envy-pair (r, h) such that $r \in S^{i,j}$ and $h \in T^{i,j}$.*

Proof. Consider the induced subgraph $G^{i,j}$ that contains only $S^{i,j}$ and $T^{i,j}$. One can see that $G^{i,j}$ is a cycle of length $4l$. Hence there are only two perfect matchings between $S^{i,j}$ and $T^{i,j}$, and they are actually $M_0^{i,j}$ and $M_1^{i,j}$. Moreover, it is easy to check that $M_0^{i,j}$ contains only one envy-pair $(s_{1,1}^{i,j}, t_{0,2}^{i,j})$ and $M_1^{i,j}$ contains only one envy-pair $(s_{0,1}^{i,j}, t_{0,1}^{i,j})$. □

We now ready to show the NP-hardness of 0-1 Min-EP HRLQ.

Lemma 5.3.3. *If G is a "yes" instance of the Vertex Cover problem, then I has a solution with at most $n^2 + m$ envy-pairs.*

Proof. If G is a "yes" instance of the Vertex Cover problem, then G has a vertex cover of size exactly K. Let this vertex cover be $V_c \subseteq V$ and let $V_f = V \backslash V_c$. We construct a matching M of I according to V_c. We match each hospital in V_c to each resident in C and each hospital in V_f to each resident in F in an arbitrary way. Since $|C \cup F| = |V| = n$, there are at most n^2 envy-pairs between $C \cup F$ and V.

Since V_c is a vertex cover of G, for each edge (v_i, v_j), either v_i or v_j is included in V_c. Thus for each gadget $g_{i,j} = (S^{i,j}, T^{i,j})$ corresponding to the edge (v_i, v_j), if $v_i \in V_c$, we use $M_1^{i,j}$ as part of our matching M, otherwise, use $M_0^{i,j}$. By constructing in this way, it is easy to see that neither v_i nor v_j can be involved in an envy-pair. Also, by Lemma 5.3.2, there is only one envy-pair between $S^{i,j}$ and $T^{i,j}$. Thus, the total number of envy-pairs is at most $n^2 + m$ w.r.t M. □

Lemma 5.3.4. *If G is a "no" instance of the Vertex Cover problem, then any solution of I has at least $n^2 + m + 1$ envy-pairs.*

Proof. It is equivalent to prove that if I admits a feasible matching M with less than n^2+m+1 envy-pairs, then G has a vertex cover of size at most K. M must be a perfect matching and must be a one-to-one correspondence between $C \cup F$ and V in order to be a feasible matching. Since all V are matched to $C \cup F$, for each gadget $g_{i,j} = (S^{i,j}, T^{i,j})$, by Lemma 5.3.2, there are only two possibilities, $M_0^{i,j}$ and $M_1^{i,j}$ and either matching admits one envy-pair within each $g_{i,j}$. In all, we have m envy-pairs between S and T.

Suppose we choose $M_0^{i,j}$ for $g_{i,j}$. If v_j is matched with a resident in F, there are l envy-pairs between v_j and $S^{i,j}$. Then we have $l + m = n^2 + m + 1$ envy-pairs, contradicting the assumption. Thus, v_j must be matched with a resident in C. By the same argument, if $M_1^{i,j}$ is chosen, v_i must be matched with a resident in C. Hence, for each edge (v_i, v_j), either v_i or v_j is matched with a resident in C. It is obvious that the set of vertices matched with residents in C is a vertex cover of size K. This completes the proof. □

Thus a polynomial-time algorithm for 0-1 Min-EP HRLQ would solve the Vertex Cover problem, implying P=NP. □

Note that the reduction above also implies NP-hardness of 0-1 Min-BP HRLQ because all envy-pairs are blocking pairs and the construction in Lemma 5.3.3 does not generate any wasteful pairs (non-envy blocking pairs). [27] gives

stronger results showing 0-1 Min-BP HRLQ is hard to approximate within the ratio of $(|H|+|R|)^{1-\epsilon}$ for any positive constant ϵ even if all preference lists are complete. While, the reduction cannot be used for 0-1 Min-EP HRLQ because there exists envy-free matchings when all preference lists are complete [19] (or a weaker requirement such that all hospitals with positive lower quotas has complete preference lists over all residents [45]) and a maximum-size envy-free matching can be found in linear time.

5.3.1 A Simple Exponential-Time Algorithm

Let I be a given instance. Starting from $k = 1$, we guess a set B of k envy-pairs. There are at most $|E|^k$ choices of B. For each choice of B, we delete each $(r, h) \in B$. Let I' be the modified instance. We apply Yokoi's algorithm to find an envy-free matching in I'. If the algorithm outputs an envy-free matching M, then it is the desired solution, otherwise, we proceed to the next guess. If we run out of all guess of B for a fixed k, we increment k by 1 and proceed as before until we find a desired solution.

Theorem 5.3.5. *There is an $O(|E|^{t+1})$-time exact algorithm for Min-EP HRLQ where t is the number of envy-pairs in an optimal solution of a given instance.*

Proof. If the algorithm ends when $k < t$, then the output matching M is an envy-free matching in I' and the set B can only introduce at most k envy-pairs, contradicting that t is minimum number of envy-pairs in any feasible

matching of instance I. Consider the execution of the algorithm for $k = t$ and any optimal solution M_{opt} and let our current guess B contains exactly the t envy-pairs of M_{opt}. Then it is easy to see that M_{opt} is envy-free in I' and satisfies all the lower quotas. Hence if we apply Yokoi's algorithm, we find a matching M is envy-free and satisfies all the lower quotas. Thus M has at most t envy-pairs in the original instance I and it is our desired solution.

For the time complexity, Yokoi's algorithm runs in $O(|E|)$ time and for each k, we apply Yokoi's algorithm at most $|E|^k$ times. Therefore, the total time is at most $\sum_{k=1}^{t} O(|E|^{k+1}) = O(|E|^{t+1})$. □

5.4 Minimum-Envy-Resident HRLQ

In this section, we consider the problem of minimizing the number of envy-residents in HRLQ. We prove a NP-hardness result for Min-ER HRLQ in the following theorem.

Theorem 5.4.1. *Min-ER HRLQ is NP-hard.*

Proof. We give a polynomial-time reduction from the NP-complete problem CLIQUE. In CLIQUE, we are given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, and asked if G contains a complete graph with K vertices as a subgraph.

Reduction: Given a graph $G = (V, E)$, and a positive integer $K \leq |V|$, which is an instance of the CLIQUE problem, we construct in instance I of Min-ER HRLQ. Define $n = |V|$, $m = |E|$ and $t = n + 1$. The set of residents

is $R = C \cup F \cup E'$ and the set of hospital is $H = V \cup \{x\}$. Each set is defined as follows:

$$
\begin{aligned}
C &= \{c_i \mid 1 \leq i \leq K\} \\
F &= \{f_i \mid 1 \leq i \leq n - K\} \\
E' &= \{e^k_{i,j} \mid (v_i, v_j) \in E, 1 \leq k \leq t\} \\
V &= \{v_i \mid 1 \leq i \leq n\}
\end{aligned}
$$

Each hospital in V has a quota $[1, 1]$ and the hospital x has a quota $[mt, mt]$.

The preference lists of residents and hospitals is shown as follows:

$$
\begin{aligned}
c_i &: [[V]] && (1 \leq i \leq K) \\
f_i &: [[V]] && (1 \leq i \leq n - K) \\
e^k_{i,j} &: v_i \quad v_j \quad x && ((v_i, v_j) \in E, 1 \leq k \leq t)
\end{aligned}
$$

$$
\begin{aligned}
v_i &: [[C]] \quad [[E'_i]] \quad [[F]] && (1 \leq i \leq n)[1, 1] \\
x &: [[E']] && [mt, mt]
\end{aligned}
$$

, where $[[V]]$ are a fixed order of all hospitals in V in an increasing order of indices, $[[C]]$ and $[[F]]$ are a fixed order of all the residents in C and F,

respectively, in an increasing order of indices, $[[E'_i]]$ is an arbitrary order of all the residents in E' that are acceptable to v_i as determined by the preference lists of residents. $[[E']]$ are a fixed order of all residents in E' in an increasing order of indices.

Lemma 5.4.2. *If I is a "yes" instance of CLIQUE, then there is a feasible matching of I having at most $(m - \binom{K}{2})t + n$ envy-residents.*

Proof. Suppose that G has a clique V_c of size K. We will construct a matching M of I from V_c. We assign all the residents in C to the hospitals in V_c in an arbitrary way and all the residents in F to the hospitals in $V \backslash V_c$ in an arbitrary way. Further, we match all the residents in E' to $\{x\}$. Clearly, M is feasible. Since V_c is a clique, $(v_i, v_j) \in E$ for any pair $v_i, v_j \in V_c (i \neq j)$. There are t residents $e^k_{i,j} (1 \leq k \leq t)$ associated with the edge (v_i, v_j). Each of these residents $e^k_{i,j}$ are assigned to the hospital x in M, which is the worst in $e^k_{i,j}$'s preference list and the hospitals v_i and v_j are assigned to residents in C, better than $e^k_{i,j}$. Hence all these residents $e^k_{i,j} (v_i, v_j \in V_c, i \neq j)$ are not envy-residents. There are $\binom{K}{2}t$ such residents and total number of residents is $mt + n$. Hence there are at most $(m - \binom{K}{2})t + n$ envy-residents. □

Lemma 5.4.3. *If I is a "no" instance of CLIQUE, then any feasible matching of I contains at least $(m - \binom{K}{2} + 1)t$ envy-residents.*

Proof. Suppose that there is a feasible matching M of I that contains less than $(m - \binom{K}{2} + 1)t$ envy-residents. We show that G contains a clique of size

K. Note that M must match all the resident in E' to $\{x\}$ because $\{x\}$ has a quota $[mt, mt]$ and is only acceptable to E'. Thus the hospitals V must only match to the residents in $C \cup F$ and there is one-to-one correspondence between V and $C \cup F$ in order for M to be feasible. Define V_c be the set of hospitals matched with C. Clearly $V_c = K$. We claim that V_c is a clique.

The total number of residents is $mt + n$. Since we assume that there are less than $(m - \binom{K}{2} + 1)t$ envy-residents, there are more than $n + (\binom{K}{2} - 1)t$ non-envy-residents (obviously an non-envy-resident is a resident that is not an envy-resident). Since $|C|+|F|= n$, there are more than $(\binom{K}{2} - 1)t$ non-envy-residents in E'. Consider the following partition of E' into t subsets: $E'_k = \{e^k_{i,j} \mid (v_i, v_j) \in E\}$ for each $1 \leq k \leq t$. There must exists a k such that E'_k contains at least $\binom{K}{2}$ non-envy-residents by Pigeonhole principle. In order for $e^k_{i,j}$ to be a non-envy-resident, we note that both v_i and v_j must match to some resident in C because if v_i or v_j is matched to F, we have an envy-pair contains $e^k_{i,j}$. Thus any pair of vertices in V_c causes such a non-envy-resident, implying that V_c is a clique. □

Because $t = n + 1$, we have $(m - \binom{K}{2} + 1)t > (m - \binom{K}{2})t + n$. Hence by Lemma 5.4.2 and Lemma 5.4.3, Min-ER HRLQ is NP-hard. □

5.5 Conclusion and Open Problems

In this chapter, we give NP hardness results of minimizing envy in terms of envy-pairs and envy-residents in the Hospitals/Resident problem with Lower

Quota. Hamada et al. [27] showed hardness of approximation for the problem of minimizing the number of blocking pairs among all feasible matchings. It would be interesting to show hardness of approximation for minimizing envy in the HRLQ problem.

Bibliography

[1] Atila Abdulkadiroğlu, Parag A Pathak, and Alvin E Roth. The new york city high school match. *American Economic Review*, 95(2):364–367, 2005.

[2] Atila Abdulkadiroğlu, Parag A Pathak, Alvin E Roth, and Tayfun Sönmez. The boston public school match. *American Economic Review*, 95(2):368–371, 2005.

[3] Atila Abdulkadiroğlu and Tayfun Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

[4] David Abraham, Ning Chen, Vijay Kumar, and Vahab S Mirrokni. Assignment problems in rental markets. In *International Workshop on Internet and Network Economics*, pages 198–213. Springer, 2006.

[5] David J Abraham, Katarína Cechlárová, David F Manlove, and Kurt Mehlhorn. Pareto optimality in house allocation problems. In *International Symposium on Algorithms and Computation*, pages 3–15. Springer, 2004.

[6] David J Abraham, Robert W Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045,

2007.

[7] Nima Anari and Vijay V Vazirani. Planar graph perfect matching is in NC. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 650–661. IEEE, 2018.

[8] Garrett Birkhoff et al. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, 1937.

[9] Xujin Chen, Guoli Ding, Xiaodong Hu, and Wenan Zang. The maximum-weight stable matching problem: duality and efficiency. *SIAM Journal on Discrete Mathematics*, 26(3):1346–1360, 2012.

[10] Richard Cole and Uzi Vishkin. Approximate and exact parallel scheduling with applications to list, tree and graph problems. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 478–491. IEEE, 1986.

[11] Stephen A Cook, Yuval Filmus, and Dai Tri Man Le. The complexity of the comparator circuit value problem. *ACM Transactions on Computation Theory (TOCT)*, 6(4):15, 2014.

[12] Brian Dean and Rommel Jalasutram. Factor revealing LPs and stable matching with ties and incomplete lists. In *Proceedings of the 3rd International Workshop on Matching Under Preferences*, pages 42–53, 2015.

[13] Xiaotie Deng, Christos Papadimitriou, and Shmuel Safra. On the complexity of equilibria. In *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, pages 67–71, 2002.

[14] Yuri Faenza, Telikepalli Kavitha, Vladlena Powers, and Xingyu Zhang. Popular matchings and limits to tractability. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2790–2809. SIAM, 2019.

[15] Tomás Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45(2):233–284, 1992.

[16] Tomás Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319, 1994.

[17] Stephen Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 754–763. ACM, 2016.

[18] Tamás Fleiner, Robert W Irving, and David F Manlove. Efficient algorithms for generalized stable marriage and roommates problems. *Theoretical computer science*, 381(1-3):162–176, 2007.

[19] Daniel Fragiadakis, Atsushi Iwasaki, Peter Troyan, Suguru Ueda, and Makoto Yokoo. Strategyproof matching with minimum quotas. *ACM Transactions on Economics and Computation (TEAC)*, 4(1):1–40, 2016.

[20] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[21] David Gale and Marilda Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.

[22] Peter Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioral Science*, 20(3):166–173, 1975.

[23] Vijay Kumar Garg and Changyong Hu. Improved paths to stability for the stable marriage problem. *arXiv preprint arXiv:2007.07121*, 2020.

[24] Sushmita Gupta, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Popular matching in roommates setting is NP-hard. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2810–2822. SIAM, 2019.

[25] Dan Gusfield and Robert W Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

[26] Philip Hall. On representatives of subsets. In *Classic Papers in Combinatorics*, pages 58–62. Springer, 2009.

[27] Koki Hamada, Kazuo Iwama, and Shuichi Miyazaki. The hospitals/residents problem with lower quotas. *Algorithmica*, 74(1):440–465, 2016.

[28] Changyong Hu and Vijay K Garg. NC algorithms for popular matchings in one-sided preference systems and related problems. In *2020 IEEE*

International Parallel and Distributed Processing Symposium (IPDPS), pages 759–768. IEEE, 2020.

[29] Changyong Hu and Vijay K. Garg. Characterization of super-stable matchings. In Anna Lubiw and Mohammad Salavatipour, editors, *Algorithms and Data Structures*, pages 485–498, Cham, 2021. Springer International Publishing.

[30] Changyong Hu and Vijay K Garg. Minimal envy matchings in the hospitals/residents problem with lower quotas. *arXiv preprint arXiv:2110.15559*, 2021.

[31] Chien-Chung Huang and Telikepalli Kavitha. An improved approximation algorithm for the stable marriage problem with one-sided ties. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 297–308. Springer, 2014.

[32] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political economy*, 87(2):293–314, 1979.

[33] Robert W Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48(3):261–272, 1994.

[34] Robert W Irving, Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna Paluch. Rank-maximal matchings. In *Proceedings of the*

fifteenth annual ACM-SIAM symposium on Discrete algorithms, pages 68–75. Society for Industrial and Applied Mathematics, 2004.

[35] Robert W Irving, Paul Leather, and Dan Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM)*, 34(3):532–543, 1987.

[36] Kazuo Iwama, Shuichi Miyazaki, Yasufumi Morita, and David Manlove. Stable marriage with incomplete lists and ties. In *International Colloquium on Automata, Languages, and Programming*, pages 443–452. Springer, 1999.

[37] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. A 25/17-approximation algorithm for the stable marriage problem with one-sided ties. *Algorithmica*, 68(3):758–775, 2014.

[38] Joseph JáJá. *An introduction to parallel algorithms*, volume 17. Addison-Wesley Reading, 1992.

[39] Richard M Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[40] Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E Paluch. Strongly stable matchings in time O(nm) and extension to the hospitals-residents problem. *ACM Transactions on Algorithms (TALG)*, 3(2):15–es, 2007.

[41] Telikepalli Kavitha and Meghana Nasre. Optimal popular matchings. *Discrete Applied Mathematics*, 157(14):3181–3186, 2009.

[42] Tamás Király and Júlia Pap. Total dual integrality of Rothblum's description of the stable-marriage polyhedron. *Mathematics of Operations Research*, 33(2):283–290, 2008.

[43] Zoltán Király. Linear time local approximation algorithm for maximum stable marriage. *Algorithms*, 6(3):471–484, 2013.

[44] Donald Ervin Knuth. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. American Mathematical Soc., 1997.

[45] Prem Krishnaa, Girija Limaye, Meghana Nasre, and Prajakta Nimbhorkar. Envy-freeness and relaxed stability: hardness and approximation algorithms. In *International Symposium on Algorithmic Game Theory*, pages 193–208. Springer, 2020.

[46] Adam Kunysz. An algorithm for the maximum weight strongly stable matching problem. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[47] Adam Kunysz, Katarzyna Paluch, and Pratik Ghosal. Characterisation of strongly stable matchings. In *Proceedings of the twenty-seventh annual*

ACM-SIAM symposium on Discrete algorithms, pages 107–119. SIAM, 2016.

[48] Chi-Kit Lam and C Gregory Plaxton. A (1+1/e)-approximation algorithm for maximum stable matching with one-sided ties and incomplete lists. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2823–2840. SIAM, 2019.

[49] Gavriela Freund Lev, Nicholas Pippenger, and Leslie G Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE transactions on Computers*, 100(2):93–100, 1981.

[50] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.

[51] David F Manlove. Stable marriage with ties and unacceptable partners. Technical report, Technical Report TR-1999-29, University of Glasgow, 1999.

[52] David F Manlove. The structure of stable marriage with indifference. *Discrete Applied Mathematics*, 122(1-3):167–181, 2002.

[53] Alberto Marchetti-Spaccamela, Umberto Nanni, and Hans Rohnert. Maintaining a topological order under edge insertions. *Information Processing Letters*, 59(1):53–58, 1996.

[54] Ernst W Mayr and Ashok Subramanian. The complexity of circuit value and network stability. *Journal of Computer and System Sciences*, 44(2):302–323, 1992.

[55] Eric McDermid. A 3/2-approximation algorithm for general stable marriage. In *International Colloquium on Automata, Languages, and Programming*, pages 689–700. Springer, 2009.

[56] Eric McDermid and Robert W Irving. Popular matchings: structure and algorithms. *Journal of combinatorial optimization*, 22(3):339–358, 2011.

[57] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[58] Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[59] Meghana Nasre and Prajakta Nimbhorkar. Popular matching with lower quotas. *arXiv preprint arXiv:1704.07546*, 2017.

[60] Rafail Ostrovsky and Will Rosenbaum. Fast distributed almost stable matchings. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 101–108, 2015.

[61] Katarzyna Paluch. Faster and simpler approximation of stable matchings. *Algorithms*, 7(2):189–202, 2014.

[62] David J Pearce and Paul HJ Kelly. Online algorithms for topological order and strongly connected components. Technical report, Citeseer, 2003.

[63] David James Pearce. *Some directed graph algorithms and their application to pointer analysis*. PhD thesis, University of London, 2005.

[64] András Radnai. Approximation algorithms for the stable matching problem. *Eötvös Lorand University*, 2014.

[65] Alvin E Roth. Incentive compatibility in a market with indivisible goods. *Economics letters*, 9(2):127–132, 1982.

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

[67] Alvin E Roth. On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica: Journal of the Econometric Society*, pages 425–427, 1986.

[68] Alvin E Roth and Andrew Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, 1977.

[69] Alvin E Roth, Uriel G Rothblum, and John H Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research*, 18(4):803–828, 1993.

[70] Uriel G Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54(1-3):57–67, 1992.

[71] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[72] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[73] Sandy Scott. *A study of stable marriage problems with ties*. PhD thesis, University of Glasgow, 2005.

[74] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[75] Boris Spieker. The set of super-stable marriages forms a distributive lattice. *Discrete applied mathematics*, 58(1):79–84, 1995.

[76] Ashok Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Stanford University, 1990.

[77] Chung-Piaw Teo and Jay Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998.

[78] John H Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

[79] Qingyun Wu and Alvin E Roth. The lattice of envy-free matchings. *Games and Economic Behavior*, 109:201–211, 2018.

[80] Yu Yokoi. Envy-free matchings with lower quotas. *Algorithmica*, 82(2):188–211, 2020.

[81] Xiong Zheng and Vijay K Garg. Parallel and distributed algorithms for the housing allocation problem. In *23rd International Conference on Principles of Distributed Systems (OPODIS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[82] Lin Zhou. On a conjecture by Gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, 1990.

Vita

Changyong Hu was born in Hunan, China. He received his Bachelor of Engineering degree in Electrical Engineering from Zhejiang University in July, 2015. Thereafter, he started pursuing a Ph.D. at UT Austin.

Email Address: colinhu9@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.