13
Algorithmics of Matching Markets

Jiehua Chen[a] and David Manlove[b]

13.1 Introduction and Motivation

Chapter 1 described classical results for two-sided matching markets involving the assignment of workers to firms based on their preferences over one another. The underlying matching problem is known in the literature as the Stable Marriage problem, and in its canonical form it corresponds to Setting I from Chapter 1. The objective is to find a *stable matching*, a one-to-one assignment of workers to firms such that no worker and firm prefer one another to their assigned partners.

The Stable Marriage problem has been the focus of a great deal of attention in the literature, and one reason for this is that the classical problem and its variants feature in many practical applications. These include entry-level labor markets, school choice, and higher education admission. For example, a many-to-one extension of Stable Marriage known as the Hospitals / Residents problem (captured by Setting III in Chapter 1 models the assignment of graduating medical students to hospital posts. The National Resident Matching Program (NRMP) administers this process in the US, which involves applications from over 40,000 aspiring junior doctors per year. At the heart of the NRMP is an algorithm for the Hospitals / Residents problem.

Centralized matching schemes (also known as clearinghouses) such as the NRMP typically involve large numbers of participants, and thus it is imperative that they incorporate efficient algorithms. Chapter 1 described an efficient algorithm for the Stable Marriage problem and showed how to extend it to the Hospitals / Residents problem. In practice, however, there are often additional features of a matching market that have to be taken into consideration, which lead to generalizations of the Stable Marriage problem that have thus far not been considered in this book from an algorithmic point of view.

For example, it is very likely that a large hospital participating in the NRMP, having many applicants, may not have enough information to rank them objectively in strict order of preference. It may prefer to rank several applicants equally, in tied

[a] Institute of Logic and Computation, Faculty of Informatics, TU Wien, Austria.
[b] School of Computing Science, University of Glasgow, UK.

batches, indicating that it is indifferent between them. This is especially likely if hospitals' preference lists are derived from scores (e.g., originating from academic assessments); several applicants may have equal scores, making them essentially indistinguishable.

Another direction involves computing "fair" stable matchings. Chapter 1described the lattice structure that holds for the set of stable matchings in an instance of the Stable Marriage problem (see Definition 1.37. The Gale–Shapley Deferred Acceptance algorithm (see Page 6 of Chapter 1) computes a stable matching μ that is either at the top or at the bottom end of this lattice. That is, matching μ is either worker-optimal or firm-optimal – but in each case, optimality for the workers or the firms comes at the expense of the other set of agents, since these matchings are worst possible for the firms and workers, respectively. One may instead wish to find a stable matching that is fair to both sides of the market – one example of such a matching is an *egalitarian stable matching*, where the overall dissatisfaction of the agents, the so called egalitarian cost, is minimized.

A further extension concerns the case in which the agents involved in the market form a single set, rather than two disjoint sets as before. In this case we obtain the non-bipartite version of Stable Marriage called the Stable Roommates problem. This problem has applications in P2P networking, as well as in dormitory allocation and in pairing players for chess or tennis tournaments.

In many of these variants, computing certain types of stable matchings becomes an NP-hard problem. Given the practical significance of these matching problems, the importance of finding ways to cope with this complexity should be clear. The purpose of this chapter is to focus on two examples where finding types of "optimal" stable matchings is NP-hard, and illustrate the algorithmic techniques that typically have been applied in order to find optimal or approximate solutions.

The first problem that we focus on concerns finding a stable matching that maximizes the number of workers that are matched, given an instance of the variant of Stable Marriage in which preference lists may include ties and need not involve every member of the other side of the market. We firstly give a reduction to demonstrate the NP-hardness of this problem. Then we give an exposition of Király's approximation algorithm that achieves a performance guarantee of $\frac{3}{2}$ using a technique that subsequently has been widely applied in various matching problem scenarios.

The second problem involves finding an egalitarian stable matching in a given instance of the Stable Roommates problem. This is an NP-hard problem, and for this, we show how techniques from parameterized algorithmics give rise to fixed-parameter algorithms when the parameter is the egalitarian cost of the solution. Specifically, these methods involve kernelization and the use of bounded search trees, which are used extensively in designing fixed-parameter algorithms.

The remainder of this chapter is organized as follows. In Section 13.2, we define preliminary notation and terminology, and give formal definitions of the key problems that will be considered in this chapter. Section 13.3 focuses on the variant of Stable Marriage with ties and incomplete lists where we seek a maximum cardinality stable matching. The fixed-parameter algorithms for finding an egalitarian stable matching in an instance of the Stable Roommates problem are described in Section 13.4. We list some open problems in Section 13.5 that are related to the problems tackled in Sections 13.3 and 13.4. Finally Section 13.6 gives some chapter notes, including references for the key existing results that we rely on.

13.2 Preliminaries

13.2.1 Definitions of Key Notation and Terminology

We begin by defining notation and terminology that will be used throughout this chapter. Firstly, for each natural number t, we denote the set $\{1, 2, \ldots, t\}$ by $[t]$.

Let $V = [n]$ be a set of n agents. Each agent $i \in V$ has a subset $V_i \subseteq V$ of agents that it finds *acceptable* as a partner and has a *preference list* $\succeq_i$ on V_i (i.e., a transitive and complete binary relation on V_i). Here, $x \succeq_i y$ means that i *weakly prefers* x over y. We use $\succ_i$ to denote the asymmetric part (i.e., $x \succeq_i y$ and $\neg(y \succeq_i x)$) and $\sim_i$ to denote the symmetric part of $\succeq_i$ (i.e., $x \succeq_i y$ and $y \succeq_i x$) so that $x \succ_i y$ means that i *strictly prefers* x to y while $x \sim_i y$ means that i regards x as tied with y. We may omit the subscript in the $\succ_i$, $\succeq_i$ and $\sim_i$ notation if it is clear from the context. For two agents x and y, we call x *most acceptable* to y if x is a maximal element in the preference list of y. Note that an agent can have more than one most acceptable agent.

A *preference profile* $\mathcal{P}$ for V is a collection $(\succeq_i)_{i \in V}$ of preference lists for each agent $i \in V$. To a preference profile $\mathcal{P} = (V, (\succeq_i)_{i \in V})$, we assign an *acceptability graph* G, which has V as its vertex set, and an edge between each pair of agents who find each other acceptable. Without loss of generality, we assume that G does not contain isolated vertices, meaning that each agent has at least one agent that it finds acceptable. A preference profile $\mathcal{P}$ may have the following properties: it is *complete* if the underlying acceptability graph is complete (i.e., it contains an edge between each pair of agents); otherwise, it is *incomplete*. The profile $\mathcal{P}$ has *ties* if there is an agent $i \in V$ for which there are two agents $x, y \in V_i$ such that $x \sim_i y$; we say that x and y are *tied* by i; otherwise, if $\mathcal{P}$ has no ties, it is said to be *strict*.

When illustrating a preference profile, in a given agent's preference list and for a given indexed set S of agents, the notation $[S]$ refers to all agents in S listed in an arbitrary but fixed strict order, whilst the notation (S) indicates a tie containing all agents in S, in both cases in the position where the symbol occurs.

The *rank* of an agent i in the preference list of some agent j, denoted $\mathsf{rank}_j^{\mathcal{P}}(i)$, is the number of agents x that j strictly prefers over i:

$$\mathsf{rank}_j^{\mathcal{P}}(i) := |\{x \in V \mid x \succ_j i\}|.$$

We will omit the superscript from $\mathsf{rank}_j^{\mathcal{P}}(i)$ if the instance is clear from the context.

Given a preference profile $\mathcal{P}$ for a set V of agents, recall that a *matching* $\mu \subseteq E(G)$ is a set of disjoint pairs $\{i, j\}$ of agents. For a pair $\{i, j\}$ of agents, if $\{i, j\} \in \mu$, then the *partner* of i, denoted by $\mu(i)$, is defined to be j; otherwise we call this pair *unmatched*. If agent i has *no partner*; i.e., i is not involved in any pair in μ, we say that i is *unmatched* by μ. If no agent is unmatched by μ then μ is *perfect*.

Given a matching μ of $\mathcal{P}$, an unmatched pair $\{i, j\} \in E(G) \setminus \mu$ *is blocking* μ if each of i and j is either unmatched or prefers the other to his/her assigned partner, i.e., it holds that (i) i is unmatched by μ or $j \succ_i \mu(i)$, and (ii) j is unmatched by μ or $i \succ_j \mu(j)$. We call a matching μ *stable* if no unmatched pair is blocking μ.

13.2.2 Central Computational Problems

We now define formally the main computational problems that we will be studying in the remainder of this chapter. We begin with the Stable Roommates problem, which is defined as follows:

Stable Roommates (SRTI)
Input: A preference profile $\mathcal{P} = (V, (\succeq_i)_{i \in V})$ for a set V of n agents.
Question: Does $\mathcal{P}$ admit a stable matching?

The SRTI acronym denotes the fact that preference lists may contain ties and the preference profile may be incomplete. We use SRI to refer to the special case of SRTI in which preference lists are strictly ordered (but the preference profile may be incomplete).

The bipartite restriction of Stable Roommates, called Stable Marriage, has as input two disjoint sets W and F of agents (referred to as the *workers* and *firms* respectively in Chapter 1, where $|W| + |F| = n$, such that each agent from one set has a preference list that ranks a subset of the agents from the other set. In other words, the acceptability graph of a Stable Marriage instance is a bipartite graph on W and F. We call the corresponding preference profile a *bipartite preference profile*. The notions that we have introduced for Stable Roommates can be restricted intuitively to also work for Stable Marriage. For instance, the preference profile of a Stable Marriage instance is *complete* if the underlying acceptability graph is a complete bipartite graph.

Stable Marriage (SMTI)
Input: A bipartite preference profile $\mathcal{P} = (W, F, (\succeq_i)_{i \in W \cup F})$ for two disjoint sets W and F of agents, where $|W| + |F| = n$.
Question: Does $\mathcal{P}$ admit a stable matching?

Analogously, we use SMI to refer to the restriction of SMTI where each preference list is strictly ordered (but the preference profile may be incomplete).

When ties are not present, determining whether an instance of SRI (and thus

SMI) admits a stable matching, and finding one if it does can be done in $O(n^2)$ time. Moreover, every instance of SMI is a yes instance since it always admits a stable matching. However, when preferences have ties, the problem of deciding whether a stable matching exists, given an instance of SRTI, becomes NP-complete even if the preferences are complete.

The situation for SMTI is more positive: SMTI still admits a stable matching, even if the preferences may be incomplete. But, there may be stable matchings with different cardinalities. By breaking the ties arbitrarily, one can find a stable matching in $O(n^2)$ time. However, finding one with maximum cardinality becomes NP-hard. The corresponding optimization problem, called MAX-CARD STABLE MARRIAGE, is defined as follows:

MAX-CARD STABLE MARRIAGE (MAX-SMTI)
Input: A bipartite preference profile $\mathcal{P} = (W, F, (\succeq_i)_{i \in W \cup F})$ for two disjoint sets W and F of agents, where $|W| + |F| = n$.
Output: A stable matching for $\mathcal{P}$ with the largest cardinality.

Stable matchings in an instance $\mathcal{P}$ of SRTI may not be unique. To find an "optimal" stable matching in $\mathcal{P}$, one can take agents' satisfaction towards a matching μ into account. This is formally captured by the *egalitarian cost* of μ, denoted by $\gamma(\mu)$, and defined as follows:

$$\gamma(\mu) \coloneqq \sum_{i \in \mathsf{V}(\mu)} \mathsf{rank}_i^{\mathcal{P}}(\mu(i)) + \sum_{k \in V \setminus \mathsf{V}(\mu)} |V_k|,$$

where $\mathsf{V}(\mu) \coloneqq \{i, j \in V \mid \{i, j\} \in \mu\}$ denotes the set of matched agents in μ. A stable matching μ is *egalitarian* if $\gamma(\mu)$ is minimum, taken over all stable matchings in $\mathcal{P}$. We now define a decision problem that is associated with finding an egalitarian stable matching:

EGALITARIAN STABLE ROOMMATES DECISION (EGAL-SRTI-DEC)
Input: A preference profile $\mathcal{P} = (V, (\succeq_i)_{i \in V})$ for a set V of n agents, and a non-negative integer γ.
Question: Does $\mathcal{P}$ admit a stable matching with $\gamma(\mu) \leq \gamma$?

See Figure 13.1 for an example. We let EGAL-SRI-DEC denote the special case of EGAL-SRTI-DEC in which preference lists are strictly ordered (but may be incomplete). The bipartite restriction of EGAL-SRI-DEC is denoted by EGAL-SMI-DEC. EGAL-SMI-DEC is solvable in polynomial time, but the variant in which preferences are complete and may contain ties is NP-complete. On the other hand EGAL-SRI-DEC is NP-complete even for complete lists.

In Section 13.3 we study MAX-SMTI and give a reduction to show that this problem is NP-hard even if the ties occur on one side only. We also present a simple and elegant approximation algorithm due to Király for this special case of MAX-SMTI, proving that it has a performance guarantee of $\frac{3}{2}$. In Section 13.4 we investigate

$1: 3 \succ 6 \succ 2,$ $\quad 2: 4 \succ 1 \succ 6 \succ 3,$
$3: 6 \succ 1 \succ 2 \succ 5,$ $\quad 4: 5 \succ 2 \succ 8,$
$5: 3 \succ 4 \succ 7 \succ 8,$ $\quad 6: 2 \succ 3 \succ 1,$
$7: 5,$ $\quad 8: 4 \succ 5.$

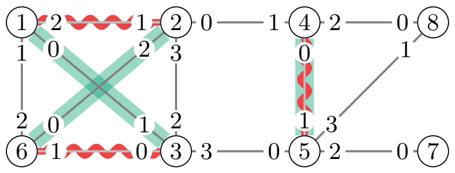

Figure 13.1 Left: An SRI instance with eight agents; it admits two stable matchings in which agents 7 and 8 are never matched. Right: The corresponding acceptability graph, with edges labeled with ranks. The stable matching marked in green and solid lines has egalitarian cost 7 while the one marked in red and curly lines has egalitarian cost 8; recall that each unmatched agent induces a cost that equals the length of her preference list.

the parameterized complexity of Egal-SRI-Dec. We show that Egal-SRI-Dec is fixed-parameter tractable with respect to the parameter “egalitarian cost” via two efficient algorithms.

13.3 Stable Marriage with Ties and Incomplete Lists: NP-hardness and Approximation

13.3.1 NP-hardness of Max-SMTI

In this section we show that Max-SMTI is NP-hard. In particular, we show that this result holds even if the ties occur in the preference lists on one side only. The result is established by proving that the following decision problem is NP-complete for this restriction on the placement of ties:

Max-Card Stable Marriage Decision (Max-SMTI-Dec)
Input: An SMTI instance $\mathcal{P} = (W, F, (\succeq_i)_{i \in W \cup F})$ and an integer $s \geq 0$.
Question: Does $\mathcal{P}$ admit a stable matching μ with $|\mu| \geq s$?

Theorem 13.1 Max-SMTI-Dec *is* NP*-complete, even if the ties occur in the preference lists on one side only.*

Proof Clearly Max-SMTI-Dec is in NP, since checking whether a given matching is stable and has cardinality at least s can be done in polynomial time. To show NP-hardness, we give a reduction from the NP-complete problem Independent Set, as follows. An instance of Independent Set comprises a graph $G = (V, E)$ and an integer $h \geq 0$, and the problem is to decide whether G has an *independent set* (i.e., a subset of vertices that are pairwise non-adjacent to each other) of size h.

The general idea behind the reduction is as follows: introduce vertex and edge agents corresponding to the vertices and edges of the graph, and h pairs of selector agents which must be matched with the vertex agents in any maximum-size stable

Workers' preference lists:

w_i :	$[T] \succ u_i$	$\forall i \in [n']$
s_j :	$u_1 \succ u_2 \succ \cdots \succ u_{n'}$	$\forall j \in [h]$
$e_j^{u_i}$:	$e_j \succ u_i \succ f_j$	$\forall e_j \in E$ with $e_j = \{v_i, v_{i'}\}$
$e_j^{u_{i'}}$:	$e_j \succ u_{i'} \succ f_j$	$\forall e_j \in E$ with $e_j = \{v_i, v_{i'}\}$.

Firms' preference lists:

u_i :	$w_i \succ [\{e_j^{u_i} \mid v_i \in e_j$ for some edge $e_j \in E\}] \succ s_1 \succ s_2 \succ \cdots \succ s_h$	$\forall i \in [n']$
t_j :	(W)	$\forall j \in [h]$
e_j :	$(\{e_j^{u_i}, e_j^{u_{i'}}\})$	$\forall e_j \in E$ with $e_j = \{v_i, v_{i'}\}$
f_j :	$[\{e_j^{u_i}, e_j^{u_{i'}}\}]$	$\forall e_j \in E$ with $e_j = \{v_i, v_{i'}\}$.

Figure 13.2 Preference lists in the instance I' of Max-SMTI-Dec constructed in the proof of Theorem 13.1. Given a set of agents A, recall that $[A]$ denotes all agents in A listed in an arbitrary but fixed strict order, whilst (A) indicates a tie containing all agents in A.

matching. The preferences of the vertex and edge agents will ensure that the vertex agents that are matched to the selector agents induce an independent set.

Let $I = ((G = (V, E), h)$ be an instance of Independent Set, where $V = \{v_1, v_2, \ldots, v_{n'}\}$ and $E = \{e_1, e_2, \ldots, e_{m'}\}$. Construct a Max-SMTI-Dec instance I' as follows. Firstly, let $W' = W \cup S \cup E^U$ be the set of workers in I', where $W = \{w_1, w_2, \ldots, w_{n'}\}$, $S = \{s_1, s_2, \ldots, s_h\}$ and $E^U = \{e_j^{u_i}, e_j^{u_{i'}} \mid e_j = \{u_i, u_{i'}\} \in E\}$. Next, let $F' = U \cup T \cup E \cup F$ be the set of firms in I', where $U = \{u_1, u_2, \ldots, u_{n'}\}$, $T = \{t_1, t_2, \ldots, t_h\}$, $E = \{e_1, e_2, \ldots, e_{m'}\}$ and $F = \{f_1, f_2, \ldots, f_{m'}\}$.

Intuitively, for $v_i \in V$, agents u_i and w_i correspond to vertex v_i in G. For each edge $e_j = \{u_i, u_{i'}\} \in E$, agents e_j, f_j, $e_j^{u_i}$, and $e_j^{u_{i'}}$ correspond to edge e_j in G. Although the notation for the agent e_j in I' is the same as that for the edge e_j in G, the precise meaning should be clear from the context. Finally, the agents in $S \cup T$ are intended to receive partners that correspond to h vertices that are selected in an independent set in G.

The preference profile $\mathcal{P}$ in I' is described in Figure 13.2. Also see Figure 13.3 for an illustration of the reduction. We observe that the preference lists of the firms in $T \cup E$ contain ties, while the preference lists of the firms in $U \cup F$ and those of the workers are strictly ordered. To complete the construction of $I' = (\mathcal{P}, s)$, we let the target size of stable matching be $s = n' + 2m' + h$.

Clearly I' can be constructed from I in linear time, and the number of agents on each side of I' is s. We claim that I has an independent set of size h if and only if I' has a stable matching of size s. Before proving the claim, we give some further intuition for the reduction as follows. The vertices $v_i \in V$ of an independent set in G correspond to firms $u_i \in U$ that are matched to a worker in S. Since each such firm u_i prefers all workers $e_j^{u_i} \in E^U$ where v_i is an endpoint of edge $e_j = \{v_i, v_{i'}\}$,

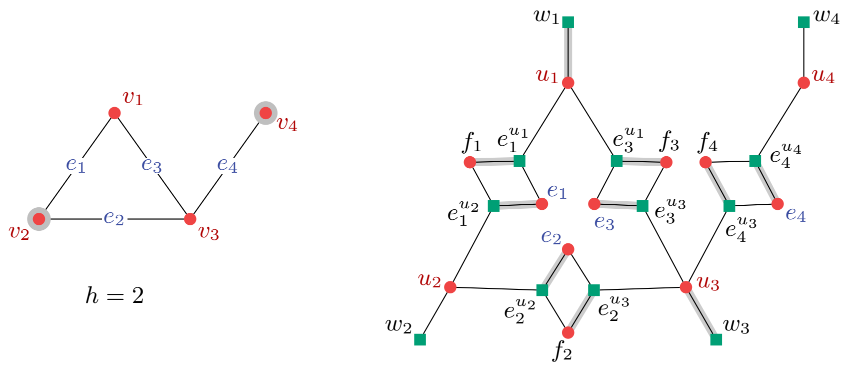

Figure 13.3 Illustration of the NP-hardness reduction for the proof of Theorem 13.1. Left: an instance of Independent Set. An independent set solution containing $V' = \{v_2, v_4\}$ is highlighted with gray circles. Right: the acceptability graph of the constructed instance; for the sake of readability, the agents from $S \cup T$ are omitted. The crucial part of the stable matching which corresponds to the independent set V' is marked in gray.

by stability each such worker $e_j^{u_i}$ must be matched to her first choice e_j, meaning that $e_j^{u_{i'}}$ must be matched to her third choice f_j. In turn, $e_j^{u_{i'}}$ prefers $u_{i'}$ to her partner, and hence by stability, $u_{i'}$ cannot be matched to a worker in S. We thus obtain that at most one endpoint of e_j corresponds to a firm matched to a worker in S, which establishes the independence property.

To prove the “only if” direction, assume that $V' := \{v_{i_1}, v_{i_2}, \ldots, v_{i_h}\}$ is an independent set in G, where $i_1 < i_2 < \ldots < i_h$. We form a matching μ of size s in I' as follows.

(a) For each $z \in [h]$, add to μ the two pairs $\{w_{i_z}, t_z\}$ and $\{s_z, u_{i_z}\}$.
(b) For each $v_i \in V \setminus V'$, add to μ the pair $\{w_i, u_i\}$.
(c) For each edge $e_j \in E$ with $e_j = \{v_i, v_{i'}\}$, where $i < i'$, if $v_i \in V'$, then add to μ the two pairs $\{e_j^{u_i}, e_j\}$ and $\{e_j^{u_{i'}}, f_j\}$; otherwise add to μ the two pairs $\{e_j^{u_{i'}}, e_j\}$ and $\{e_j^{u_i}, f_j\}$.

Please notice that in Step (c), if neither v_i nor $v_{i'}$ belong to V', adding $\{e_j^{u_i}, e_j\}$ and $\{e_j^{u_{i'}}, f_j\}$ instead of $\{e_j^{u_{i'}}, e_j\}$ and $\{e_j^{u_i}, f_j\}$ to matching μ also preserves stability. Clearly $|\mu| = 2h + (n' - h) + 2m' = s$, and μ matches all agents in I'. We claim that μ is stable in I'.

Clearly no firm in $T \cup E$ can be involved in a blocking pair of μ in I'. Neither can any firm in F, since no worker prefers a firm in F to her partner. An unmatched pair $\{w_{i'}, u_{i'}\}$, for some $i' \in [n']$, cannot block μ either, since $w_{i'}$ does not prefer $u_{i'}$ to her partner in μ. Suppose that an unmatched pair $\{s_j, u_{i'}\}$ blocks μ, for some $i' \in [n']$ and $j \in [h]$. Then, $u_{i'} = u_{i_z}$ and $\{s_z, u_{i_z}\} \in \mu$ for some $z \in [h]$, by construction of μ. As $u_{i'}$ prefers s_j to $\mu(u_{i'})$, it follows that $j < z$. By construction of μ, $\{s_j, u_{i_j}\} \in \mu$, and as $i_j < i_z$, it follows that s_j does not prefer $u_{i'}$ to $\mu(s_j)$, a contradiction.

Finally suppose that $\{e_j^{u_{i'}}, u_{i'}\}$ blocks μ, where $e_j \in E$ and $u_{i'} \in e_j$. Then, $\mu(u_{i'}) \in S$, which implies that $v_{i'} \in V'$. Thus by construction of μ, it follows that $\{e_j^{u_{i'}}, e_j\} \in \mu$. This implies that $e_j^{u_{i'}}$ has her most-preferred partner in μ, and thus $\{e_j^{u_{i'}}, u_{i'}\}$ does not block μ after all, a contradiction. Hence, μ is stable in I'.

Conversely suppose that I' admits a stable matching μ of size s. Then all agents in I' are matched in μ. Define the set $V' := \{v_i \in V \mid \mu(u_i) \in S\}$. Clearly $|V'| = h$. We claim that V' is an independent set in G. For, suppose that $e_j = \{v_i, v_{i'}\} \in E$ where $u_i \in V'$ and $u_{i'} \in V'$. Then $\mu(u_i) \in S$ and $\mu(u_{i'}) \in S$ by construction of V'. As f_j must be matched in μ, either $\{e_j^{u_i}, f_j\} \in \mu$ or $\{e_j^{u_{i'}}, f_j\} \in \mu$. In the former case $\{e_j^{u_i}, u_i\}$ blocks μ in I', whilst in the latter case $\{e_j^{u_{i'}}, u_{i'}\}$ blocks μ in I'. Both of these are a contradiction and hence the claim is established. □

13.3.2 Király's Approximation Algorithm for MAX-SMTI *with one-sided ties*

In this section we describe Király's $\frac{3}{2}$-approximation algorithm for the special case of MAX-SMTI in which the ties occur in the preference lists on one side only. We also show how to prove that the algorithm is correct and has performance guarantee $\frac{3}{2}$. In what follows, we assume without loss of generality that we are given an instance $\mathcal{P} = (W, F, (\succeq_i)_{i \in W \cup F})$ of MAX-SMTI in which the ties occur in the firms' lists only, and workers' lists do not contain ties. Henceforth we refer to this special case of MAX-SMTI as MAX-SMTI-TF.

Király's algorithm for MAX-SMTI-TF is similar to the classical Gale–Shapley algorithm for STABLE MARRIAGE in that it involves a series of applications from workers to firms, and possible rejections of workers by firms. A key distinction is that Király's algorithm allows a worker w_i who has been rejected by every firm on her preference list to have a "second chance" and apply to them again in a second pass through her list. During this second pass, w_i is said to be *promoted*, which means that, for any firm f_j on her list, w_i has a higher priority according to f_j than any unpromoted worker that she is tied with on f_j's list. To formalize this, we define the notion of *favors* as follows.

Definition 13.2 A firm f_j is said to *favor* a worker w_i over another worker w_k if either (i) or (ii) holds, as follows:

(i) f_j strictly prefers w_i to w_k (i.e., $w_i \succ_{f_j} w_k$), or
(ii) w_i and w_k are tied in f_j's list, and w_i is promoted whilst w_k is not.

A worker can only be promoted once: after a second pass through her list, if w_i has again been rejected by every firm on her list then w_i will not be able to apply to any firm again and will be unmatched in the final matching. We say that w_i is *exhausted* if w_i has been rejected from every firm in her preference list (either during a first pass or a second pass through her list).

Algorithm 1: Király's approximation algorithm for Max-SMTI-TF

```
Input: An Max-SMTI-TF instance $\mathcal{P} = (W, F, (\succeq_i)_{i \in W \cup F})$
1  $\mu := \emptyset$
2  foreach $w_i \in W$ do
3      $promoted(w_i)$ := false
4      $exhausted(w_i)$ := false
5  while some $w_i \in W$ is unmatched and $(!promoted(w_i)$ or $!exhausted(w_i))$ do
6      if $exhausted(w_i)$ then
7          $promoted(w_i)$ := true
8          $exhausted(w_i)$ := false
9          reactivate $w_i$      // i.e., set $w_i$ to have been rejected by no firms
10     $f_j$ := most-preferred firm on $w_i$'s list that has not yet rejected her
       // $w_i$ applies to $f_j$
11     if $f_j$ is unmatched then $\mu := \mu \cup \{\{w_i, f_j\}\}$
12     else
13         if $f_j$ favors $w_i$ over $\mu(f_j)$ then      // recall Definition 13.2
14             $f_j$ rejects $\mu(f_j)$
15             $\mu := (\mu \cup \{\{w_i, f_j\}\}) \setminus \{\{\mu(f_j), f_j\}\}$
16         else $f_j$ rejects $w_i$
17     if $w_i$ is rejected by every firm on her list then $exhausted(w_i)$ := true
18 return $\mu$
```

A pseudocode description of Király's algorithm is given in Algorithm 1. We now give an explanation of the algorithm. Initially the matching μ is empty, and booleans for each worker w_i are set to indicate that w_i has not been promoted yet and w_i is not exhausted yet. The main loop iterates as long as there is some worker w_i who is unmatched, and additionally w_i has not been promoted yet or w_i is not yet exhausted. If w_i is exhausted then, as we know w_i has not been promoted by line 5, w_i has completed only one pass through her preference list. In preparation for a second pass through her list, we then set w_i as promoted and not exhausted, and "reactivate" w_i, meaning that we now assume that no firms have rejected w_i.

Lines 10–16 of the algorithm are similar to the Gale–Shapley algorithm for Stable Marriage. That is, w_i applies to the most-preferred firm f_j on her list that has not yet rejected her. If f_j is unmatched then it accepts the application and becomes assigned to w_i. Otherwise f_j is already matched to some worker w_k in μ. If f_j favors w_i over w_k (see Definition 13.2) then f_j rejects w_k and becomes assigned to w_i instead, otherwise f_j rejects w_i and remains assigned to w_k. Notice here that if w_i is promoted, she can displace an unpromoted worker w_k assigned to f_j even if w_i and w_k are tied in f_j's list. Finally lines 17–17 ensure that a worker is set to be exhausted if she has been rejected by every firm on her list.

The following series of lemmas establish the correctness of the algorithm.

Lemma 13.3 *Given an instance $\mathcal{P}$ of* Max-SMTI-TF*, all possible executions of Király's algorithm as applied to $\mathcal{P}$ produce a stable matching μ in $\mathcal{P}$.*

Proof Let μ be the matching returned by the algorithm and suppose that $\{w, f\}$ blocks μ. Then either w is unmatched in μ and finds f acceptable, or else w is matched in μ and strictly prefers f to $\mu(f)$. In either case w applied to f and it rejected her because either (i) it was already assigned to a worker w' and it did not favor w over w', or (ii) it subsequently received an application from a worker w' whom it favored over w. In either case, either w' and w are tied in f's list, or f strictly prefers w' to w. Hence f weakly prefers w' to w. Moreover, any subsequent change of partner for f cannot cause it to become strictly worse off, so f weakly prefers $\mu(f)$ to w. Hence $\{w, f\}$ does not block w after all, a contradiction. □

Lemma 13.4 *Given an instance $\mathcal{P}$ of* Max-SMTI-TF*, any execution of Király's algorithm as applied to $\mathcal{P}$ produces a stable matching μ in $\mathcal{P}$ such that $|\mu| \geq \frac{2}{3}|\mu'|$, where μ' is any stable matching in $\mathcal{P}$.*

Proof By Lemma 13.3, Király's algorithm produces a stable matching μ. Let μ' be a maximum cardinality stable matching and let $G' = (V, E')$ be a subgraph of the acceptability graph of $\mathcal{P}$ where $V = W \cup F$ and $E' = \mu \oplus \mu'$ (where $\oplus$ denotes the symmetric difference of μ and μ'). Then the connected components of G' are paths and cycles whose edges alternate between μ and μ' (we refer to these components as alternating paths and alternating cycles, respectively). We firstly claim that G' has no alternating path of length three whose end edges belong to μ'.

For, suppose for a contradiction that $\{w', f\}$, $\{f, w\}$, $\{w, f'\}$ is an alternating path of length three, where $\{w', f\} \in \mu'$, $\{f, w\} \in \mu$ and $\{w, f'\} \in \mu'$. Then each of w' and f' is unmatched in μ. This means that w' applied to (and was rejected by) every firm on her preference list as an unpromoted and promoted worker, and f' did not receive an application from any worker. We deduce that w was never promoted, and moreover that w strictly prefers f to f', otherwise she would have applied to f' (recall that workers do not have ties in their lists).

After w' was promoted, w' applied to f. As in the proof of Lemma 13.3, since $\mu(f) = w$, f rejected w' because either (i) it was already assigned to a worker w'' and it did not favor w' over w'', or (ii) it subsequently received an application from a worker w'' whom it favored over w'. Moreover, any subsequent change of partner for f cannot cause it to become strictly worse off, so f weakly prefers $\mu(f) = w$ to w'.

Suppose that w and w' are tied in f's list. Then the same is true for w' and w'' (possibly $w'' = w$). Moreover w'' must be promoted, for otherwise f would have favored w' over w'' (recall that w' has been promoted). But f ultimately exchanges its partner for an unpromoted worker in μ, namely w, which is impossible. Hence

f strictly prefers w to w'. It follows that $\{w, f\}$ blocks μ', a contradiction. Hence the claim is established.

Now let C be any connected component of G'. If C is an alternating path whose end edges belong to μ', it follows from our preceding argument that $|C| \neq 3$, and it follows from the stability of μ that $|C| \neq 1$. Hence $|C| \geq 5$, and $|\mu \cap C| \geq \frac{2}{3}|\mu' \cap C|$. If C is an alternating cycle, an alternating path of even length, or an alternating path of odd length whose end edges belong to μ, clearly $|\mu \cap C| \geq |\mu' \cap C|$. The lemma thus follows. □

Lemma 13.5 *Given an instance $\mathcal{P}$ of* Max-SMTI-TF*, Király's algorithm as applied to $\mathcal{P}$ runs in $O(L)$ time, where L is the total length of the workers' preference lists.*

Proof Each worker applies at most twice to the same firm (once as an unpromoted worker and once as a promoted worker) so the number of iterations of the main **while** loop is $O(L)$. Each worker is reactivated at most once, and hence the total time taken for reactivation is $O(L)$. Using an array to store the ranks of workers in the firms' preference lists (allowing a firm to decide whether it favors one worker to another in $O(1)$ time), and using a stack to keep track of unmatched workers, Király's algorithm can be implemented to run in $O(L)$ time. □

Together, Lemmas 13.3 to 13.5 lead to the following conclusion.

Theorem 13.6 *Király's algorithm is a $\frac{3}{2}$-approximation algorithm for* Max-SMTI-TF.

We now give an example to illustrate the execution of Király's algorithm; the example also shows that the bound of $\frac{3}{2}$ is tight.

Example 13.7 Consider the following instance $\mathcal{P}$ of Max-SMTI-TF:

$$\begin{array}{llll} w_1: & f_2 \succ f_1 & f_1: & w_1 \\ w_2: & f_2 \succ f_3 & f_2: & w_1 \sim w_2 \\ w_3: & f_3 & f_3: & w_2 \succ w_3. \end{array}$$

The following execution trace results in a matching μ_1 of cardinality 2:

- w_1 applies to f_2, $\{w_1, f_2\}$ added to μ_1;
- w_2 applies to f_2, f_2 rejects w_2;
- w_2 applies to f_3, $\{w_2, f_3\}$ added to μ_1;
- w_3 applies to f_3, f_3 rejects w_3, w_3 is exhausted;
- w_3 promoted and reactivated;
- w_3 applies to f_3, f_3 rejects w_3, w_3 is exhausted.

On the other hand, $\mathcal{P}$ admits a stable matching μ_2 of cardinality 3, comprising pairs $\{w_1, f_1\}$, $\{w_2, f_2\}$, $\{w_3, f_3\}$. Note that if w_2 applies first in the above execution trace then matching μ_2 will ultimately be returned. □

13.4 Stable Roommates without Ties: Two Parameterized Algorithms

13.4.1 Introduction

In this section, we focus on Egal-SRI-Dec. Since this problem is NP-hard, exact algorithms presumably need super-polynomial time when measured only in the input length. A way out, without resorting to randomness or approximation, is given by the framework of *Parameterized Algorithmics* in which we aim to exploit structural properties of the input, measured by so-called integer-valued *parameters*. In this way, we can design more refined exact algorithms by viewing their running time as a function of both the input size and the parameter. One central goal of Parameterized Algorithmics is to design *fixed-parameter algorithms*, which solve any instance I of a given a problem Q with respect to a parameter k in $f(k) \cdot |I|^{O(1)}$ time, where f is some computable function (usually exponential) of the parameter k and $|I|$ denotes the size of (an arbitrary encoding of) I.

In the Egal-SRI-Dec problem the parameter could be the upper bound γ on the egalitarian cost of a matching that we aim for. In this section, we provide two fixed-parameter algorithms for Egal-SRI-Dec with respect to the parameter "egalitarian cost γ", which run in $O(n^2 + (\gamma+1)^\gamma \cdot \gamma^2)$ time and $O(2^\gamma \cdot n^2)$ time, respectively.

When the preferences do not have ties, there are various structures that can be utilized for designing efficient algorithms. For instance, whenever there are two agents x and y who are each other's most acceptable agents (i.e., $\mathsf{rank}_x(y) = \mathsf{rank}_y(x) = 0$), every stable matching must contain the pair $\{x, y\}$, which has zero cost. Hence, we can safely add such pairs to an egalitarian solution matching without disturbing the egalitarian cost. After we have matched all pairs of agents with zero cost, all remaining unmatched agents induce cost at least one when they are matched. Thus, to obtain a matching with egalitarian cost at most γ, there can remain at most 2γ agents and moreover no agent can be matched to an agent with rank higher than γ.

Indeed, we can go one step further and consider an even smaller parameter than the overall egalitarian cost, namely the one where we subtract both the cost induced by the unmatched agents and by pairs that appear in every stable matching (called *fixed pairs*) from the cost that we are aiming for. Note that fixed pairs and unmatched agents are unique and can be found in polynomial time by the following structural results.

Theorem 13.8 *For each instance $\mathcal{P} = (V, (\succ_i)_{i\in V})$ of* SRI *with n agents one can in $O(n^2)$ time (i) compute the set of all pairs agents which appear in every stable matching, and (ii) partition the agent set V into two disjoint subsets V^m and V^u such that every stable matching matches every agent from V^m and none of the agents from V^u.*

To apply Theorem 13.8, we introduce six additional notions.

Definition 13.9 For an instance $(\mathcal{P}, \gamma)$ of Egal-SRI-Dec, let $\mathsf{F}(\mathcal{P})$ denote the set consisting of all fixed pairs of $\mathcal{P}$. Further, let $\mathsf{V}(\mathsf{F}(\mathcal{P})) := \{x, y \in V \mid \{x, y\} \in \mathsf{F}(\mathcal{P})\}$ and $\gamma(\mathsf{F}(\mathcal{P}))$ denote the set of agents in $\mathsf{F}(\mathcal{P})$ and the egalitarian cost induced by the fixed pairs in $\mathsf{F}(\mathcal{P})$, respectively. Let $\mathsf{V}^{\mathsf{u}}(\mathcal{P})$ denote the set consisting of all agents that are unmatched in all stable matchings. Let $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ denote the set of remaining agents *not* from $\mathsf{V}(\mathsf{F}(\mathcal{P})) \cup \mathsf{V}^{\mathsf{u}}(\mathcal{P})$. Define $\hat{\gamma} := \gamma - \gamma(\mathsf{F}(\mathcal{P})) - \sum_{z \in \mathsf{V}^{\mathsf{u}}(\mathcal{P})} |V_z|$.

By Theorem 13.8 and by Definition 13.9, the sets $V_{\mathsf{F}}(\mathcal{P})$, $\mathsf{V}^{\mathsf{u}}(\mathcal{P})$, and $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ partition the whole agent set V. Note that, since each agent in $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ must be matched by each stable matching so that the cost, together with her partner, is at least one, each stable matching has an egalitarian cost bounded as follows.

Observation 13.10 *Every stable matching μ of an* SRI *instance $\mathcal{P}$ satisfies* $\gamma(\mu) \geq \frac{|\mathsf{V}^{\mathsf{r}}(\mathcal{P})|}{2} + \gamma(\mathsf{F}(\mathcal{P})) + \sum\limits_{z \in \mathsf{V}^{\mathsf{u}}(\mathcal{P})} |V_z|$.

In the remainder of this section, we apply two well-established parameterized techniques: *kernelization* and *bounded search tree algorithms* and obtain two fixed-parameter algorithms for Egal-SRI-Dec with respect to the "reduced" parameter $\hat{\gamma}$ (see Definition 13.9).

13.4.2 Kernelization for Egal-SRI-Dec

A *kernelization* is a *polynomial-time* preprocessing algorithm that transforms an instance I of a problem Q with parameter value k into an *equivalent* instance I' of Q with parameter value k' with $|I'| + k' \leq g(k)$, where g is a computable function. The resulting instance I' together with k' is called a *kernel* and g is referred to as the *size* of the kernel. Typically, kernelization is based on several polynomial-time executable *data-reduction rules* which translate an instance to an equivalent one while ultimately shrinking the instance size.
We show that Egal-SRI-Dec admits a kernel of quadratic size for the parameter $\hat{\gamma}$.

Theorem 13.11 Egal-SRI-Dec *admits a size-$O(\hat{\gamma}^2)$ kernel with at most $4\hat{\gamma} + 2$ agents and with each preference list of size at most $\hat{\gamma}+1$. The kernel can be computed in $O(n^2)$ time. Hence,* Egal-SRI-Dec *can be solved in $O(n^2 + (\hat{\gamma}+1)^{\hat{\gamma}} \cdot \hat{\gamma}^2)$ time.*

Proof We show that given an instance $I = (\mathcal{P}, \gamma)$ of Egal-SRI-Dec with $\mathcal{P} = (V, (\succ_i)_{i \in V})$, Algorithm 2 produces a kernel with at most $4\hat{\gamma} + 2$ agents with preference list length at most $\hat{\gamma} + 1$ each; we note that each of the following blocks of lines in the algorithm can be considered as a reduction rule: line 2, lines 3-5, and lines 6–10. Briefly put, our kernelization algorithm will delete all agents in $\mathsf{V}(\mathsf{F}(\mathcal{P})) \cup \mathsf{V}^{\mathsf{u}}(\mathcal{P})$, i.e., keep all agents in $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$, and replace the deleted agents by a small number of dummy agents to maintain the egalitarian-cost structure. In the

```
Algorithm 2: Kernelization for Egal-SRI-Dec
   Input: An instance $I = (\mathcal{P} = (V = [n], (\succ_i)_{i\in V}), \gamma)$ of Egal-SRI-Dec
 1 $\hat{\gamma} \coloneqq \gamma - \gamma(\mathsf{F}(\mathcal{P})) - \sum_{x\in \mathsf{V}^{\mathsf{u}}(\mathcal{P})} |V_x|$
 2 if $|\mathsf{V}^{\mathsf{r}}(\mathcal{P})| > 2\hat{\gamma}$ then return a trivial no-instance
 3 $D \coloneqq \{d_i \mid 1 \le i \le 2(\hat{\gamma}+1)\}$ // Create dummy agents
 4 foreach $i \in [\hat{\gamma}+1]$ do
 5     Construct the preference lists $\succ_{d_{2i-1}}$ and $\succ_{d_{2i}}$ of $d_{2i-1}$ and $d_{2i}$ such that
         $\mathsf{rank}_{d_{2i-1}}(d_{2i}) = \mathsf{rank}_{d_{2i}}(d_{2i-1}) = 0$
   // Update the preference lists of the agents in $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$
 6 foreach agent $x \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$ do
 7     foreach agent $y \in V_x$ with $\mathsf{rank}^{\mathcal{P}}_x(y) \le \hat{\gamma}$ do
 8         if $\mathsf{rank}^{\mathcal{P}}_x(y) + \mathsf{rank}^{\mathcal{P}}_y(x) > \hat{\gamma}$ or $y \in \mathsf{V}(\mathsf{F}(\mathcal{P})) \cup \mathsf{V}^{\mathsf{u}}(\mathcal{P})$ then
 9             In list $\succ_x$, replace $y$ with a dummy $d \in D$, using a different dummy $d$
                 for each such $y$ to ensure that $d$'s list length is at most $\hat{\gamma}+1$, and
                 append $x$ to $\succ_d$
10     Remove all agents $y'$ from $\succ_x$ with $\mathsf{rank}^{\mathcal{P}}_x(y') > \hat{\gamma}$
11 return $(\mathsf{V}^{\mathsf{r}}(\mathcal{P}) \cup D, (\succ_i)_{i\in \mathsf{V}^{\mathsf{r}}(\mathcal{P})} + (\succ_d)_{d\in D}, \hat{\gamma})$
```

following when saying that some lines in the algorithm are *correct* we mean that the instances before and after conducting those lines are equivalent.

First, the correctness of line 2 is given by Observation 13.10 and the definition of $\hat{\gamma}$. Second, the introduction of $2\hat{\gamma}+2$ dummy agents in lines 3–5 does not contribute any egalitarian cost to any stable matching; hence, these lines are correct.

Third, in lines 6–10, we update the preference lists of all original agents that will stay in the kernel. These are those agents that belong to $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ (see line 6). To see why the inner loop in lines 7–9 is correct, let us consider an arbitrary agent x with $x \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$ and one of her acceptable agents from V_x, say y. In order to obtain a stable matching with egalitarian cost at most γ, agent x cannot be assigned to y if the sum of their respective ranks exceeds $\hat{\gamma}$ (see the first condition in line 8) due to Observation 13.10. Moreover, by Theorem 13.8, no stable matching will match x with y if $y \in \mathsf{V}(\mathsf{F}(\mathcal{P})) \cup \mathsf{V}^{\mathsf{u}}(\mathcal{P})$. Hence, we can safely replace y with some dummy agent in line 9 if y satisfies one of the two conditions given in line 8.

Finally, by the same reasoning as above, it is also correct in line 10 to remove in the preference list of $x \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$ all agents y' that have $\mathsf{rank}^{\mathcal{P}}_x(y')$ higher than $\hat{\gamma}$. It remains to show that the updated preference lists are symmetric, meaning that an agent x remains in the preference list of another agent y if and only if y remains in the preference list of x. Towards a contradiction, suppose that x and y are two agents with $x, y \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$ such that y remains in the preference list of x, while x does not remain in the preference list of y. Then, by lines 8 and 10 it follows that

$\mathsf{rank}^{\mathcal{P}}_x(y)+\mathsf{rank}^{\mathcal{P}}_y(x) \le \hat{\gamma}$. This further implies that $\mathsf{rank}^{\mathcal{P}}_y(x) \le \hat{\gamma}$, and hence x must also remain in the preference list of y, a contradiction.

Altogether, for each agent $x \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$, her updated preference list has at most $\hat{\gamma}+1$ agents, each of which is from $D \cup \mathsf{V}^{\mathsf{r}}(\mathcal{P})$.

It remains to bound the size of the kernel. The kernel has $|\mathsf{V}^{\mathsf{r}}(\mathcal{P})|$ original agents and $2\hat{\gamma}+2$ dummy agents. By line 2 we know that $|\mathsf{V}^{\mathsf{r}}(\mathcal{P})| \le 2\hat{\gamma}$. Thus, the kernel has at most $4\hat{\gamma}+2$ agents. As for the lengths of the preference lists, by line 10, each remaining original agent has at most $\hat{\gamma}+1$ agents in her list. This means that the total length of the preference lists of the remaining original agents is at most $2\hat{\gamma}\cdot(\hat{\gamma}+1)$. Since we have introduced $2\hat{\gamma}+2$ dummy agents, each ranking some unique dummy agent in the first place, there remain $2\hat{\gamma}\cdot(\gamma+1)$ entries in the preference lists of all dummy agents to be filled up with original agents. By line 9, we ensure that each dummy agent has at most $\hat{\gamma}+1$ agents in her preference list.

As for the running time of Algorithm 2, note that $|V| = n$. By Theorem 13.8, computing $\mathsf{F}(\mathcal{P})$, $\mathsf{V}^{\mathsf{u}}(\mathcal{P})$, and $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ takes $O(n^2)$ time. Line 2 can be conducted in $O(n)$ time. Constructing the dummy agents in lines 3–5 takes $O(\hat{\gamma})$ time. The number of iterations in the two loops in lines 6–7 and in line 10 is $O(n\cdot\hat{\gamma})$. Using an array to store the ranks of the agents in the preference list of each agent and an array to mark whether an agent is in $\mathsf{V}(\mathsf{F}(\mathcal{P}))$, $\mathsf{V}^{\mathsf{u}}(\mathcal{P})$, or $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$, we can test the condition in line 8 in $O(1)$ time. Using an array to store the preference list of $\succ_x$ and a counter to mark a dummy agent d_i whose preference list has length less than $\hat{\gamma}+1$, we can perform the replacement in line 9 in $O(1)$ time.

Thus, in total, Algorithm 2 takes $O(n^2)$ time. The claimed running time in the third statement can be shown for instance via an exhaustive brute-force search of all $(\hat{\gamma}+1)^{\hat{\gamma}}$ possible matchings on the remaining agents from $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ to check whether one of them is stable in $O(\hat{\gamma}^2)$ time. Note that the depth of the search tree is bounded by $\hat{\gamma}$ since we are building at most $\hat{\gamma}$ pairs. $\square$

Example 13.12 To illustrate Algorithm 2, consider the instance I given in Figure 13.1. One can verify that it has exactly two stable matchings $\mu_1 = \{\{1,3\},\{2,6\},\{4,5\}\}$ and $\mu_2 = \{\{1,2\},\{3,6\},\{4,5\}\}$. Observe that $\mathsf{F}(\mathcal{P}) = \{\{4,5\}\}$. According to Theorem 13.8, the agent set can be partitioned into $\mathsf{V}(\mathsf{F}(\mathcal{P})) = \{4,5\}$, $\mathsf{V}^{\mathsf{u}}(\mathcal{P}) = \{7,8\}$, $\mathsf{V}^{\mathsf{r}}(\mathcal{P}) = \{1,2,3,6\}$. The egalitarian costs of μ_1 and μ_2 are $\gamma(\mu_1) = 7$ and $\gamma(\mu_2) = 8$, respectively.

Let $\gamma = 7$. Then, by Definition 13.9, we obtain that $\hat{\gamma} = 3$. The preference lists returned by the algorithm could look as follows, where $D = \{d_1, d_2, \ldots, d_8\}$.

$$\begin{array}{llll}
1: 3 \succ 6 \succ 2, & 2: d_1 \succ 1 \succ 6 \succ d_2, & 3: 6 \succ 1 \succ d_1 \succ d_2, & 6: 2 \succ 3 \succ 1, \\
d_1: d_2 \succ 2 \succ 3, & d_2: d_1 \succ 2 \succ 3, & \forall i \in \{2,3,4\}\colon\ d_{2i-1}: d_{2i}, & d_{2i}: d_{2i-1}.
\end{array}$$

13.4.3 Bounded Search Tree Algorithms for Egal-SRI-Dec

Besides kernelization, another simple but useful parameterized technique is the use of *bounded search tree algorithms*. They are a restricted variant of exhaustive search algorithms, which use the parameter to cut the branches in the search tree. Roughly speaking, they recursively apply the following branching steps until an equivalent and "easy-to-solve" instance is found. In each branching step on an input instance I,

(i) we identify a small set of elements so that at least one element in the set belongs to a solution,
(ii) and then branch into considering all possible "smaller" instances, each of which is obtained by fixing one element of the subset as part of the solution.

If in each branching step the size of the identified subset is bounded by $f(k)$, and the depth of the branching is bounded by $g(k)$, where f and g are two computable functions depending only the parameter k, then the execution of such an algorithm results in a search tree with $f(k)^{g(k)}$ nodes. If the identification of each subset in the branching runs in polynomial time, i.e., $|I|^{O(1)}$ time, then we obtain a fixed-parameter algorithm.

Now, we show that Egal-SRI-Dec can be solved by a simple *bounded search tree* algorithm to obtain the following result.

Theorem 13.13 Egal-SRI-Dec *can be solved in* $O(2^{\hat{\gamma}} \cdot n^2)$ *time.*

Proof Let $I = (\mathcal{P} = (V, (\succ_i)_{i \in V}), \gamma)$ be an instance of Egal-SRI-Dec. Let $\mathsf{F}(\mathcal{P})$, $\mathsf{V}^{\mathsf{u}}(\mathcal{P})$, $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$, and $\hat{\gamma}$ be as defined in Definition 13.9. By Observation 13.10, we assume that $\hat{\gamma} \geq 0$, otherwise we halt, reporting that I is a no instance.

First, we set $\mu \coloneqq \mathsf{F}(\mathcal{P})$ because all stable matchings must contain all fixed pairs from $\mathsf{F}(\mathcal{P})$. Our branching algorithm will extend the matching μ to find a stable one with egalitarian cost at most γ (or report that no such matching exists) and works as follows.

- As long as $\hat{\gamma} > 0$ and there remains a not-yet-matched agent u from $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$ with $|V_u^*| \geq 1$, where $V_u^* \coloneqq \{v \in V_u \cap \mathsf{V}^{\mathsf{r}}(\mathcal{P}) \mid \mathsf{rank}_u(v) + \mathsf{rank}_v(u) \leq \hat{\gamma}\}$, pick an arbitrary such agent. Further, let $\mathsf{best}(u)$ denote the unique agent $v \in V_u^*$ with $\mathsf{rank}_v(u) = 0$ (if she exists) such that $\mathsf{rank}_u(v)$ is the smallest, i.e.,

$$\mathsf{best}(u) \coloneqq \begin{cases} \arg\min_{v \in V_u^*} \{\mathsf{rank}_u(v) \mid \mathsf{rank}_v(u) = 0\}, & \text{if some } v \text{ has } \mathsf{rank}_v(u) = 0, \\ \perp, & \text{otherwise.} \end{cases}$$

- Branch into all possible ways of matching u as follows, distinguishing between two cases.
  - If $\mathsf{best}(u) \neq \perp$, then for each $v \in V_u^*$ with $\mathsf{rank}_u(v) \leq \mathsf{rank}_u(\mathsf{best}(u))$, branch into adding $\{u, v\}$ to μ; note that, by definition, u cannot be matched with an agent with rank higher than $\mathsf{rank}_u(\mathsf{best}(u))$ as otherwise $\{u, \mathsf{best}(u)\}$ forms a blocking pair.
  - Otherwise, for each $v \in V_u^*$, branch into adding $\{u, v\}$ to μ.

For each of the branches, update $\mathsf{V}^{\mathsf{r}}(\mathcal{P}) := \mathsf{V}^{\mathsf{r}}(\mathcal{P}) \setminus \{u, v\}$ and decrease the remaining budget $\hat{\gamma} := \hat{\gamma} - \mathsf{rank}_u(v) - \mathsf{rank}_v(u)$, and continue as follows:

(i) If $\hat{\gamma} < 0$ or there exists an agent $u' \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$ with $|V^*_{u'}| = 0$, then stop and reject the current μ.

(ii) If $\hat{\gamma} > 0$ and $\mathsf{V}^{\mathsf{r}}(\mathcal{P}) \neq \emptyset$, then recurse with an arbitrary agent $u' \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$.

(iii) If $\hat{\gamma} = 0$, then check whether the matching μ is stable for $\mathcal{P}$. Accept if μ is stable, otherwise reject the current μ.

Correctness. It is straightforward to see that $\mathcal{P}$ has a stable matching of egalitarian cost at most γ if and only if one of the leaves (Case (iii)) in the produced search tree accepts.

Running time. The running time of the algorithm is bounded by the number of nodes in the search tree multiplied by the time used for each node.

The time used for each node is bounded by the time required for computing $\mathsf{F}(\mathcal{P})$, $\mathsf{V}^{\mathsf{u}}(\mathcal{P})$, $\mathsf{V}^{\mathsf{r}}(\mathcal{P})$, $\mathsf{best}(u)$, and V^*_u for all $u \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$, which is $O(n^2)$ (see also Theorem 13.8).

In order to bound the number of nodes in the search tree, we first bound the number of leaves in the tree. To this end, let $\mathsf{N}(k)$ and $\mathsf{L}(k)$ denote the upper bounds on the number of nodes and leaves, respectively, in a search tree in relation to its height k. Clearly, if a search tree algorithm solves a problem instance with parameter value k and calls itself recursively on problem instances with parameter values at most $k - d_1$, $k - d_2$, ..., $k - d_q$ with $1 \leq d_1 \leq d_2 \leq \cdots \leq d_q \leq k$, then an upper bound on the number of leaves in the built search tree is given by the following linear recurrence $\mathsf{L}(k) \leq \mathsf{L}(k - d_1) + \mathsf{L}(k - d_2) + \cdots + \mathsf{L}(k - d_q)$. Assuming that $\mathsf{L}(k) = \lambda^k$ (where λ is a positive constant), the recurrence is satisfied if the following holds:

$$\lambda^{d_q} - \lambda^{d_q - d_1} - \lambda^{d_q - d_2} - \cdots - \lambda^{d_q - d_q} = 0. \tag{13.1}$$

Using standard analysis, we know that the left hand side of (13.1), called the *characteristic polynomial* of the recurrence, has a unique positive root λ_0 such that $\mathsf{L}(k) = \lambda_0^k$.

To find the unique positive root λ_0, we analyze the decrease of the budget (the parameter) in each call of our algorithm, distinguishing between two cases:

- If $\mathsf{best}(u) \neq \perp$, then $\mathsf{rank}_u(\mathsf{best}(u)) \leq \hat{\gamma}$ and the recursive procedure makes at most $\mathsf{rank}_u(\mathsf{best}(u)) + 1$ recursive calls. Accordingly, the budget in these calls is decreased by at least $1, 2, \ldots, \mathsf{rank}_u(\mathsf{best}(u))$, $\mathsf{rank}_u(\mathsf{best}(u))$, respectively; note that $\mathsf{best}(u) \in V^*_u$. To see why the budgets in the first $\mathsf{rank}_u(\mathsf{best}(u))$ calls are updated in this way, we observe that for each agent $u \in \mathsf{V}^{\mathsf{r}}(\mathcal{P})$ and each acceptable agent $v' \in V^*_u$ with $\mathsf{rank}_u(v') < \mathsf{rank}_u(\mathsf{best}(u))$, the definition of $\mathsf{best}(u)$ implies that $\mathsf{rank}_{v'}(u) \geq 1$.
- If $\mathsf{best}(u) = \perp$, then $|V^*_u| \leq \hat{\gamma}$ and the recursive procedure makes $|V^*_u|$ recursive calls. The budget in these calls is decreased by at least $1, 2, \ldots, |V^*_u|$, respectively.

To see why the budgets are updated in this way, we observe that $\mathsf{best}(u)$ does not exist, so each acceptable agent $v' \in V_u^*$ has $\mathsf{rank}_{v'}(u) \geq 1$.

The characteristic polynomials of the recurrence (13.1) in the two cases thus are:

$\lambda^{\hat{\gamma}} - \lambda^{\hat{\gamma}-1} - \lambda^{\hat{\gamma}-2} - \cdots - \lambda^{\hat{\gamma}-\hat{\gamma}} - \lambda^{\hat{\gamma}-\hat{\gamma}}$ and $\lambda^{\hat{\gamma}} - \lambda^{\hat{\gamma}-1} - \cdots - \lambda^{\hat{\gamma}-\hat{\gamma}}$, respectively.

Since the unique positive root of the first polynomial is 2, while the unique positive root of the second polynomial is less than 2, our search tree has $\mathsf{L}(\hat{\gamma}) \leq 2^{\hat{\gamma}}$ leaves since its height is bounded by $\hat{\gamma}$.

Now, to bound the number of nodes in the tree, observe that $\mathsf{N}(1) = 1$ and $\mathsf{N}(\hat{\gamma}) = \mathsf{N}(\hat{\gamma} - 1) + \mathsf{L}(\hat{\gamma})$. This implies that $\mathsf{N}(\hat{\gamma}) \leq 2 \cdot 2^{\hat{\gamma}} - 1$ since $\mathsf{L}(\hat{\gamma}) \leq 2^{\hat{\gamma}}$. All together, our algorithm runs in $O(2^{\hat{\gamma}} \cdot n^2)$ time. □

13.5 Selected Open Questions

In Section 13.3 we presented a simple and elegant $\frac{3}{2}$-approximation algorithm for Max-SMTI-TF. In fact, a stronger, but more complex, approximation algorithm for this problem is known, with performance guarantee $(1 + \frac{1}{e}) \approx 1.3679$, and this is the best current upper bound at the time of writing. The best current lower bound for this problem is $\frac{5}{4} - \varepsilon$, for any $\varepsilon > 0$, assuming the Unique Games Conjecture (UGC) holds. For the general Max-SMTI problem (where ties can be on both sides), the best current approximation algorithm has performance guarantee $\frac{3}{2}$, whilst the best current lower bound is $\frac{4}{3} - \varepsilon$, for any $\varepsilon > 0$, assuming UGC. It remains open to close these gaps for both Max-SMTI and Max-SMTI-TF by providing improved approximation algorithms or stronger inapproximability results, leading to tighter upper and lower bounds.

In Section 13.4, we have seen that Egal-SRI-Dec admits a polynomial kernel, and hence is fixed-parameter tractable with respect to the egalitarian cost γ. When the preferences may have ties, Egal-SRTI-Dec still admits a fixed-parameter algorithm with running time $2^{O(\gamma \log \gamma)} \cdot (n \log n)^3$. It would be interesting to know whether Egal-SRTI-Dec also admits a polynomial kernel when ties are present. The running time given in Theorem 13.13 is tight in the sense that there exists no $2^{o(\gamma)} \cdot n^{O(1)}$-time algorithm for Egal-SRI-Dec for the case without ties, unless the *Exponential Time Hypothesis* fails. Another question is whether the $2^{O(\gamma \log \gamma)} \cdot (n \log n)^3$-time algorithm for the case with ties is tight.

13.6 Bibliographic Notes

We note that our definition of a blocking pair is consistent with that of Gusfield and Irving (1989), but the definition given in Chapter 1 for Setting II (see Definition 1.19) is slightly different. Nevertheless, both versions lead to the same notion of a stable matching. In the presence of ties, our stability definition is also referred to as *weak stability* in the literature; we note here that two stronger stability definitions (so-called *strong stability* and *super-stability*) have been considered also, but the set of matchings satisfying either of these criteria may be empty (see Manlove (2013, Chapter 3)).

Gale and Shapley (1962) gave an $O(n^2)$ algorithm to find a stable matching in an instance of SMI, whilst Irving's algorithm (Irving, 1985) finds a stable matching or reports that none exists, given an instance I of SRI. We note that all stable matchings in I have the same size, and match the same set of agents (Gusfield and Irving, 1989, Section 4.5.2). Ronn (1990) proved that deciding whether a stable matching exists, given an instance of SRTI, is NP-complete even for complete preference lists.

Irving (1994) showed that a stable matching in an instance I of SMTI can be found in $O(n^2)$ time, whilst Manlove et al. (2002) observed that stable matchings in I can have different cardinalities. The NP-hardness of MAX-SMTI for the special case in which the ties occur on one side only was first established by Manlove et al. (2002). However, the proof of Theorem 13.1 incorporates a new reduction from INDEPENDENT SET that has not appeared previously in the literature.

Approximation algorithms for MAX-SMTI were surveyed by Cechlárová et al. (2019). The approximation algorithm for MAX-SMTI-TF, described in Section 13.3.2, is due to Király (2011). The correctness and efficiency of the algorithm (i.e., it always returns a stable matching, it has a performance guarantee of $\frac{3}{2}$ and it can be implemented to run in linear time), and the example showing that the performance guarantee is tight, were all given by Király (2011). Further details regarding the data structures required for efficient implementation of Király's algorithm can be found in Gusfield and Irving (1989, Section 1.2.3).

Theorem 13.8(i) was observed by Gusfield (1988, Lemma 7.1). Theorem 13.8(ii) was established by Gusfield and Irving (1989, Theorem 4.5.2). The running time in Theorem 13.8 is obtained by using Irving's algorithm (Irving, 1985) to solve SRI.

The NP-hardness of EGAL-SRI-DEC for complete preference lists was established by Feder (1992), who also gave a 2-approximation algorithm for minimizing the egalitarian cost. Cseh et al. (2019) provided a dichotomy result regarding the maximum length ℓ of the preference lists, and showed that it is polynomial-time solvable if $\ell = 2$, and is NP-hard for $\ell \geq 3$. The polynomial-time algorithm for EGAL-SMI-DEC is due to Irving et al. (1987), whilst the NP-completeness result for the extension of EGAL-SMI-DEC to the case where preference lists are complete but can include ties is due to Manlove et al. (2002).

Notice from the definition of egalitarian cost that an unmatched agent contributes the length of her preference list to this measure. In the absence of ties, the value of this contribution is irrelevant since the set of unmatched agents is the same across all stable matchings in $\mathcal{P}$. For the case with ties, the situation changes, and we could for example have defined the contribution of an unmatched agent to the egalitarian cost to be n. However, since finding an egalitarian stable matching is NP-hard even for instances of SRI with complete preference lists, any contribution to the egalitarian cost for an unmatched agent results in the same hardness result. The choice of value only has significance when reasoning about parameterized complexity in SRTI: giving unmatched agents an egalitarian cost n only makes devising an FPT algorithm easier. Chen et al. (2018) also investigated the parameterized complexity for the cases when the egalitarian cost of an unmatched agent is a constant value or zero.

The polynomial kernel and the fixed-parameter tractability result for EGAL-SRI-DEC parameterized by the egalitarian cost γ are due to Chen et al. (2018). Chen et al. (2018) also showed that EGAL-SRTI-DEC is fixed-parameter tractable with respect to the parameter γ. Discussion on the Exponential Time Hypothesis can be found in Impagliazzo et al. (2001). For more information about the UGC, mentioned in Section 13.5, see Khot (2002).

Further algorithmic results for matching markets can be found in the monograph of Manlove (2013). More specifically, parameterized algorithms and complexity results for other matching problems under preferences can be found in Gupta et al. (2018) and Chen (2019). More techniques from Parameterized Algorithmics can be found in the following textbooks: Niedermeier (2006); Cygan et al. (2015).

Acknowledgements

The authors would like to thank Tamás Fleiner and Zhiyi Huang for valuable comments on an earlier version of this chapter that helped to improve the presentation. Jiehua Chen was supported by the WWTF research project (VRG18-012). David Manlove was supported by grant EP/P028306/1 from the Engineering and Physical Sciences Research Council.

References

Cechlárová, Katarínà, Cseh, Ágnes, and Manlove, David. 2019. Selected open problems in Matching Under Preferences. *Bulletin of the EATCS*, **128**, 14–38.

Chen, Jiehua. 2019. *Computational Complexity of Stable Marriage and Stable Roommates and Their Variants.* Tech. rept. arXiv:1904.08196 [cs.GT].

Chen, Jiehua, Hermelin, Danny, Sorge, Manuel, and Yedidsion, Harel. 2018. How hard is it to satisfy (almost) all roommates? Pages 35:1–35:15 of: *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP '18).*

Cseh, Ágnes, Irving, Robert W., and Manlove, David F. 2019. The Stable Roommates Problem with Short Lists. *Theory of Computing Systems*, **63**(1), 128–149.

Cygan, Marek, Fomin, Fedor V., Kowalik, Lukasz, Lokshtanov, Daniel, Marx, Dániel, Pilipczuk, Marcin, Pilipczuk, Michal, and Saurabh, Saket. 2015. *Parameterized Algorithms.* Springer.

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

Gale, David, and Shapley, Lloyd S. 1962. College admissions and the stability of marriage. *American Mathematical Monthly*, **69**, 9–15.

Gupta, Sushmita, Roy, Sanjukta, Saurabh, Saket, and Zehavi, Meirav. 2018. Some Hard Stable Marriage Problems: A Survey on Multivariate Analysis. Pages 141–157 of: Neogy, S.K., Bapat, Ravindra B., and Dubey, Dipti (eds), *Mathematical Programming and Game Theory.* Indian Statistical Institute Series. Singapore: Springer Singapore.

Gusfield, Dan. 1988. The Structure of the Stable Roommate Problem: Efficient Representation and Enumeration of All Stable Assignments. *SIAM Journal on Computing*, **17**(4), 742–769.

Gusfield, Dan, and Irving, Robert W. 1989. *The Stable Marriage Problem–Structure and Algorithms.* Foundations of Computing Series. MIT Press.

Impagliazzo, Russell, Paturi, Ramamohan, and Zane, Francis. 2001. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, **63**(4), 512–530.

Irving, Robert W. 1985. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, **6**(4), 577–595.

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

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

Khot, Subhash. 2002. On the Power of Unique 2-Prover 1-Round Games. Pages 767–775 of: *Proceedings of STOC '02: the Thirty-Fourth Annual ACM Symposium on Theory of Computing.* Association for Computing Machinery.

Király, Zoltán. 2011. Better and simpler approximation algorithms for the stable marriage problem. *Algorithmica*, **60**, 3–20.

Manlove, David F. 2013. *Algorithmics of Matching Under Preferences.* Series on Theoretical Computer Science, vol. 2. World Scientific.

Manlove, David F., Irving, Robert W., Iwama, Kazuo, Miyazaki, Shuichi, and Morita, Yasufumi. 2002. Hard variants of stable marriage. *Theoretical Computer Science*, **276**(1-2), 261–279.

Niedermeier, Rolf. 2006. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press.

Ronn, Eytan. 1990. NP-complete stable matching problems. *Journal of Algorithms*, **11**(2), 285–304.