Solving Hard Stable Matching Problems Involving Groups of Similar Agents

Kitty Meeks
School of Computing Science
University of Glasgow
Glasgow, UK
kitty.meeks@glasgow.ac.uk

Baharak Rastegari
Department of Computer Science
University of Bristol
Bristol, UK
baharak.rastegari@bristol.ac.uk

Abstract

Many important stable matching problems are known to be NP-hard, even when strong restrictions are placed on the input. In this paper we seek to identify simple structural properties of instances of stable matching problems which will allow the design of efficient algorithms. We focus on the setting in which all agents involved in some matching problem can be partitioned into k different *types*, where the type of an agent determines his or her preferences, and agents have preferences over types (which may be refined by more detailed preferences within a single type). This situation could arise in practice if agents form preferences based on some small collection of agents' attributes. The notion of types could also be used if we are interested in a relaxation of stability, in which agents will only form a private arrangement if it allows them to be matched with a partner who differs from the current partner in some particularly important characteristic. We show that in this setting several well-studied NP-hard stable matching problems (such as MAX SMTI, MAX SRTI, and MAX SIZE MIN BP SMTI) belong to the parameterised complexity class FPT when parameterised by the number of different types of agents, and so admit efficient algorithms when this number of types is small.

1 Introduction

Matching problems occur in various applications and scenarios such as the assignment of children to schools, college students to dorm rooms, junior doctors to hospitals, and so on. In all the aforementioned, and similar, problems, is it understood that the participants (which we will refer to as agents) have preferences over other agents, or subsets of agents. The majority of the literature assumes that these preferences are ordinal, and that is the assumption we make in this work as well. Moreover, it is widely accepted that a "good" and "reasonable" solution to a matching problem must be *stable*, where stability is defined according to the context of the problem at hand. Intuitively speaking, a stable solution guarantees that no subset of agents find it in their best interest to leave the prescribed solution and seek an assignment amongst themselves. Unfortunately, many interesting and important stable matching problems are known to be NP-hard even for highly restricted cases.

In this work, we consider a setting where agents can be grouped into k different "types", where the type of an agent specifies what the agent's preferences are and how s/he is compared against other agents. Such a setting could arise in practice if, for example, agents derive their preferences by considering some small collection of attributes of other agents. The notion of types is also useful if we are interested in a relaxation of stability, where agents are only willing to form a private arrangement with a partner who is distinctly superior to their current partner with respect to an important characteristic. We show that we can solve some of the most important hard stable matching problems efficiently from the point of view of parameterized complexity when the number of types is taken as the parameter; many of these results rely on the fixed parameter tractability of Integer Linear Programming. We also demonstrate that, by imposing further restrictions, some of these problems become polynomial-time solvable.

Perhaps the most widely studied matching problem is the *Stable Marriage problem (SM)*. In an instance of SM we have two disjoint set of agents, men and women, each having a preference ordering over the individuals of the opposite sex (candidates). A solution to this problem is a *matching*, that is an mapping from men to women where each man is matched to at most one woman and vice versa. A matching is *stable* if there are no two agents a and b who prefer each other to their assigned partners. If such a pair exists, we say that (a, b) is a *blocking pair* and both a and b are *blocking agents*. In their seminal work, Gale and Shapley [20] showed that every stable marriage problem admits a stable matching that can be found in polynomial time by their proposed algorithm (GS). Simple extensions of GS can be used to identify stable matchings in domains where agents are permitted to declare some candidates unacceptable (*Stable Marriage with Incomplete lists (SMI)*), are allowed to express preferences in which they are indifferent between two or more agents (*Stable Marriage with Ties (SMT)*), or both (*Stable Marriage with Ties and Incomplete lists (SMTI)*).

It is known that an instance of SMTI might admit stable matchings of different sizes. In many practical applications, it is important to match as many agents as possible, and thus finding a maximum cardinality stable matching (i.e., a stable matching with the largest size amongst all stable matchings) is a crucial issue.

Definition 1. MAX SMTI *is the problem of determining the maximum cardinality stable matching in an instance of SMTI.*

Depending on the application, one might be willing to tolerate a small degree of instability if that leads to larger matchings. Two different measurements for the degree of instability have been introduced in the literature: the number of blocking pairs and the number of blocking agents.

Definition 2. MAX SIZE MIN BP SMI *(respectively* MAX SIZE MIN BA SMI*) is the problem of finding a matching, out of all maximum cardinality matchings, which has the minimum number of blocking pairs (respectively, minimum number of blocking agents) in an instance of SMI.*

The *Stable Roommate problem (SR)* is a non-bipartite generalization of SM. Extensions allowing for incomplete lists and indifference in preference lists are defined the same way as for SM. An instance of SR need not admit a stable matching. There exists a polynomial time algorithm that finds a stable matching in an instance of SR, or reports that none exists [23].

Definition 3. Max SRT *is the problem of identifying a maximum cardinality stable matching in an instance of SRT, or reporting that none exists.*

Since an instance of SR may not admit a stable matching, it is of interest to find a matching with minimum number of blocking pairs.

Definition 4. Min BP SR *is the problem of identifying a matching which has the minimum number of blocking pairs in an instance of SR.*

Hospitals/Residents problem with Ties (HRT) is a famous extension of SMTI that models many practical applications, including the assignment of junior doctors to hospitals, by allowing agents on one side of the market to be assigned to multiple agents on the other side of the market. Mechanisms very similar to that proposed in [20] have been used to compute a stable assignment of residents to hospitals. The concern of computing a maximum size stable matching extends to instances of HRT.

Definition 5. Max HRT *is the problem of determining the maximum cardinality stable matching in an instance of HRT.*

All the problems defined above are known to be NP-complete [33, 25, 3, 37, 7, 24, 1]. Note that we have provided Definition 2 for instances of SMI. We chose to do so because these two problems are hard even when there are no ties in preference lists. Likewise, Definition 3 assumes that all agents find all other agents acceptable, as the hardness result holds even under this assumption. Definition 4 is defined for instances of SR, as this problem is hard even if all agents rank all the other agents in strict order of preference. In fact, the aforementioned problems are hard even when the input is heavily restricted. For example, Max SMTI is NP-complete even if each man's preference list is strictly ordered, and each woman's preference list is either strictly ordered or is a tie of length 2 [33].

Structure of the rest of the paper. In Section 2 we provide definitions for the settings we study, as well as a brief introduction to parameterised complexity and existing results on Integer Linear Programming. In Section 3 we formally define what we mean by agents having types, discuss some related work, and summarise our results. In subsequent sections we present our results on the complexity of computing a maximum cardinality stable matching in an instance of SMTI or HRT (Section 4), a maximum cardinality stable matching in an instance of SRTI (Section 5), and a maximum cardinality matching with minimum number of blocking pairs/agents (Section 6) in our restricted setting. In Section 7 we provide a definition for the Hospitals/Residents problem with Couples (where a pair of residents can send in preferences over pairs of hospitals) and present our result on the complexity of computing maximum cardinality stable matchings in such scenarios. We motivate our setting by discussing how it may arise in practice in Section 8, along with a brief description on how to extract types from ordinal preference lists. We conclude in Section 9 and provide some directions for future work.

2 Preliminaries

In this section we introduce the main concepts we use in the paper; we begin with some definitions, then provide a brief introduction to parameterised complexity, before describing existing results on Integer Programming.

2.1 Definitions

In this section we provide the key definitions for the stable matching settings we study; for further background and terminology we refer the reader to [31].

Let N denote a set of n agents, which in a bipartite matching setting (i.e. SMTI or HRT) is composed of two disjoint sets. Each hospital h in an instance of HRT is associated with a capacity $q(h)$ that denotes the number of posts it offers. When in a bipartite matching setting, we use the term *candidates* to refer to the agents on the opposite side of the market to that of an agent under consideration. In a non-bipartite settings, candidates refer to all the other agents except the one under consideration.

Each agent finds a subset of candidates acceptable and ranks them in order of preference. Preferences orderings need not to be strict, so it is possible for an agent to be indifferent between two or more candidates. We write $b \succ_a c$, or equivalently $c \prec_a b$, to denote that agent a prefers candidate b to candidate c, and $b \simeq_a c$ to denote that a is indifferent between b and c. We write $b \succeq_a c$ to denote that a either prefers b to c or is indifferent between them, and say that *a weakly prefers b to c*.

In an instance of SMTI, a *matching* M is a pairing of men and women such that no one is paired with an unacceptable partner, each man is paired with at most one woman, and each woman is paired with at most one man. In an SRTI instance, a matching is a pairing of agents such that each agent is matched with at most one other agent whom s/he additionally finds acceptable. We write $(a, b) \in M$ to say that a and b are matched in M. In an instance of HRT, a matching is a pairing of hospitals and residents such that no agent is paired with an unacceptable candidate, each resident is matched with at most one hospital, and each hospital h is matched with at most $q(h)$ residents. We use $M(a)$ to denote the agent (or the set of agents in the case of hospitals) matched to a in M. We write $M(a) = \varnothing$ if agent a is unmatched in M. We assume that every agent prefers being matched to an acceptable candidate to remaining unmatched.

Given an instance of SMTI or SRTI, a matching M is *(weakly) stable* if there is no pair $(a, b) \notin M$ where a prefers b to his current partner in M, i.e., $b \succ_a M(a)$, and vice versa. Given an instance of HRT, a matching M is stable if there is no acceptable (resident,hospital) pair (r, h) such that (i) r prefers h to $M(r)$, and (ii) either $|M(h)| < q(h)$ or h prefers r to its worst assigned resident in M.

2.2 Parameterised Complexity

In this paper we are concerned with the *parameterised complexity* of computational problems that are intractable in the classical sense. Parameterised complexity provides a multivariate framework for the analysis of hard problems: if a problem is known to be NP-hard, so that we expect the running-time of any algorithm to depend exponentially on some aspect of the input, we can seek to restrict this combinatorial explosion to one or more *parameters* of the problem rather than the total input size. This has the potential to provide an efficient solution to the problem if the parameter(s) in question are much smaller than the total input size. A parameterised problem with total input size n and parameter k is considered to be tractable if it can be solved by a so-called *fpt-algorithm*, an algorithm whose running time is bounded by $f(k) \cdot n^{\mathcal{O}(1)}$, where f can be any computable function. Such problems are said to be fixed parameter tractable, and belong to the complexity class FPT. It should be emphasised that, for a problem to be in FPT, the exponent of the polynomial must be independent of the parameter value; problems which satisfy the weaker condition that the running time is polynomial for any constant value of the parameter(s) (so that the degree of the polynomial may depend on the parameters) belong to the class XP.

For further background on the theory of parameterised complexity, we refer the reader to [15, 18].

2.3 The complexity of Integer Programming

Most of the algorithms we present in this paper make use of an algorithm for Integer Linear Programming in some way. This problem is formally stated as follows: given an $m \times k$ matrix A and two m-dimensional vectors $\mathbf{b}$ and $\mathbf{c}$ (all with coefficients in $\mathbb{Z}$), find a m-dimensional vector $\mathbf{x} \in \mathbb{Z}$ which minimizes the scalar product $\mathbf{c}^T \cdot \mathbf{x}$, subject to the m linear constraints given by $A\mathbf{x} \leq \mathbf{b}$, or else report that no vector satisfying the constraints exists. Note that we can easily translate problems in which we wish to maximise rather than minimise the objective function into this form, and also we can express constraints based on linear equalities as a combination of linear inequalities; for simplicity of presentation we will use both of these generalisations when expressing problems as instances of Integer Linear Programming.

While Integer Linear Programming is NP-hard in general, one of the most celebrated results in parameterised complexity is that this problem belongs to FPT when parameterised by the number of variables.

Theorem 1 ([14], based on [19, 26, 29])**.** *An* Integer Linear Programming *instance of size L with k variables can be solved using*

$$\mathcal{O}(k^{2.5k+o(k)} \cdot (L + \log M_x) \log(M_x M_c))$$

arithmetic operations and space polynomial in $L + \log M_x$, where M_x is an upper bound on the absolute value a variable can take in a solution, and M_c is the largest absolute value of a coefficient in the vector c.

In Section 6 we also need to solve instances of Integer Quadratic Programming, a variant of Integer Linear Programming in which the objective function is quadratic. Formally, given a $k \times k$ integer matrix Q, an $m \times k$ integer matrix A and an m-dimensional integer vector $\mathbf{b}$, our goal is to find a vector $\mathbf{x} \in \mathbb{Z}^k$ which minimises $\mathbf{x}^T Q \mathbf{x}$, subject to the m linear constraints $A\mathbf{x} \leq b$, or else report that no vector satisfying the constraints exists. As before, we note that we can easily generalise this definition to deal with maximisation problems and constraints in the form of linear equalities. Lokshtanov recently gave an fpt-algorithm for this problem.

Theorem 2 ([30]). INTEGER QUADRATIC PROGRAMMING *is in* FPT *parameterised by* $k+\alpha$, *where* α *is the maximum absolute value of any entry in the matrices* A *and* Q.

3 Our Contribution

In this section we describe our approach in more detail, and provide some context for our results. We begin in Section 3.1 by giving formal definitions of the problems we consider, before discussing some related work in Section 3.2. Finally, in Section 3.3, we summarise our results.

3.1 A typed approach

Most hardness results in the study of stable matching problems are based on the premise that agents may have arbitrary preference lists. In practice, however, agents' preferences are likely to be more structured and correlated. In this work, we consider a setting where each agent is associated with a "type" that specifies his or her preferences, as well as how s/he is perceived by the other agents. We discuss how such settings may arise in practice, and how we can efficiently extract types from ordinal preference lists, in Section 8.

3.1.1 Our first model: agents of the same type are indistinguishable

The simplest model is to assume that the agents of the same type are completely indistinguishable. That is, they have the same preference lists, and every other agent that finds their type acceptable is indifferent between them. Assume that there are k types available for agents. Let N_i denote the set of agents that are of type i. Each type i has a preference ordering over types of the candidates, which need not be complete or strict. We assume, without loss of generality, that $|N_i| > 0$ for all $i \in [k]$, and that each type finds at least one other type acceptable. We write $j \succ_i \ell$, or equivalently $\ell \prec_i j$, if agents of type i strictly prefer agents of type j to agents of type ℓ. We write $j \simeq_i \ell$ to denote that agents of type i are indifferent between agents of types j and ℓ, and $j \succeq_i \ell$ if agents of type i prefer agents of type j to those of type ℓ or are indifferent between the two. We assume that given every two agents x and y of the same type:

1. x and y have identical preference lists when restricted to $N \setminus \{x, y\}$, and
2. all other agents are indifferent between x and y.

These requirements imply that any agent either finds all agents of a given type acceptable (and is indifferent between them) or finds none of them acceptable. We say that an instance of a stable matching problem satisfying these requirements is *typed*, and refer to the standard problems with input of this form as TYPED MAX SMTI etc.

We illustrate our definition of typed instances with an example.

Example 1. *Assume we have 6 types for the agents,* $\{1, \ldots, 6\}$, *and that we are in a stable marriage setting where types* 1, 2 *and* 3 *correspond to men and the rest of the types correspond* *to women. Let the preferences over types be as follows, where the preference lists are ordered from left to right in decreasing order of preference, and the types in round brackets are tied.*

$$
\begin{array}{ll}
1: 4\ 5 & 4: 3\ (2\ 1) \\
2: (4\ 5)\ 6 & 5: 2\ 3 \\
3: 5\ 6\ 4 & 6: 3\ 2
\end{array}
$$

Assume that there seven men, $\{m_1, \ldots, m_7\}$*, and seven women ,* $\{w_1, \ldots, w_7\}$*, where men* m_1 *and* m_2 *are of type* 1*, men* m_3, m_4 *and* m_5 *are of type* 2*, and the last two men are of type* 3*. Assume that* w_1 *is of type* 4*,* w_2, w_3 *and* w_4 *are of type* 5 *and the last three women are of type* 6*. Therefore, the preferences of the agents under typed model are as follows.*

$$
\begin{array}{rlrl}
m_1, m_2: & w_1\ (w_2\ w_3\ w_4) & w_1: & (m_6\ m_7)\ (m_1\ m_2\ m_3\ m_4\ m_5) \\
m_3, m_4, m_5: & (w_1\ w_2\ w_3\ w_4)\ (w_5\ w_6\ w_7) & w_2, w_3, w_4: & (m_3\ m_4\ m_5)\ (m_6\ m_7) \\
m_6, m_7: & (w_2\ w_3)\ (w_5\ w_6\ w_7)\ (w_1) & w_5, w_6, w_7: & (m_6\ m_7)\ (m_3\ m_4\ m_5)
\end{array}
$$

3.1.2 A generalisation: agents of the same type refine their preferences in the same way

We also consider a generalisation of typed instances in which agents are no longer necessarily indifferent between two agents of the same type, however agents of the same type occur consecutively in preference lists. Moreover, we assume that being indifferent between two types essentially means being indifferent between the agents of either types. This means that for any two agents x and y of the same type i:

1. x and y have identical preference lists when restricted to $N \setminus \{x, y\}$,
2. no agent of a different type appears between x and y in any preference list, and
3. if type i is indifferent between types j and ℓ then x and y are indifferent between all agents in $(N_j \cup N_\ell) \setminus \{x, y\}$.

If an instance of a stable matching problem satisfies these slightly weaker requirements, we say that the instance is *refined-typed*, and refer to the standard problems with input of this form as Refined-Typed Max SMTI etc.

We illustrate this definition with two short examples.

Example 2. *Consider the same stable marriage setting with types as in Example 1. While Example 1 already meets the requirements of refined-typed model, there are many more permitted preference profiles in this setting, as agents are allowed to break ties within a particular type. For example,* m_4, m_5 *and* m_6 *can have the following preference list:*

$$m_3, m_4, m_5: (w_1\ w_2\ w_3\ w_4)\ w_6\ (w_5\ w_7)$$

However, the following preference list is not allowed for these men, as a tie between t_4 and t_5 requires these three men to be indifferent between all women who are of either of these two types.

$$m_3, m_4, m_5 : (w_1\ w_2)\ (w_3\ w_4)\ (w_5\ w_6\ w_7)$$

Example 3. *Assume that we are in a stable roommates setting with six agents $a, b, \ldots, f$ of type 1. Assume that type 1 finds type 1 acceptable, and let the refined preferences within type 1 be as follows:*

$$(a\ b\ c)\ (d\ e\ f)$$

The agents' preference lists will then be as listed below.

$$a : (b\ c)\ (d\ e\ f) \qquad b : (a\ c)\ (d\ e\ f) \qquad c : (a\ b)\ (d\ e\ f)$$
$$d : (a\ b\ c)\ (e\ f) \qquad e : (a\ b\ c)\ (d\ f) \qquad f : (a\ b\ c)\ (d\ e)$$

3.2 Related Work

NP-hard matching problems. As stated earlier, all the problems defined in Section 1 are NP-complete, even for highly restricted cases. The NP-completeness of Max SMTI has been shown for a variety of restricted settings, for example: (1) even if each man's list is strictly ordered, and each woman's list is either strictly ordered or is a tie of length 2 [33], (2) even if each mans preference list is derived from a strictly-ordered master list of women, and each woman's preference list is derived from a master list of men that contains only one tie [25], and (3) even if the SMTI instance has symmetric preferences; that is, for any acceptable (man, woman) pair (m_i, w_j), $rank(m_i, w_j) = rank(w_j, m_i)$ [3, 37], where $rank(a, b)$ is defined to be one plus the number of candidates that a prefers to b. As SMTI is a special case of HRT, the NP-hardness of finding a maximum stable matching in the latter follows directly from the NP-hardness of this problem in the former.

Weak SRT, the problem of deciding weather a stable matching exists in an instance of SRT, is NP-complete [24]. The NP-completeness holds even if each preference list is either strictly ordered or contains a tie of length 2 at the head. It therefore follows that Max SRT is NP-complete.

Max Size Min BP SMI and Max Size Min BA SMTI are NP-hard and very hard to approximate [7] even if each agent's preference list is of length at most 3 [8, 22], but polynomial-time solvable if agents on one side of the market have preference lists of length at most 2 [8].

Parameterized complexity. There are a limited number of works addressing fixed-parameter tractability in the stable matching problems we study in this paper. Marx and Schlotter [34] study the parameterized complexity of Max SMTI. They show that the problem is in FPT when parameterised by the total length of the ties, but is W[1]-hard when parameterised by the number of ties in the instance, even if all the men have strictly ordered preference lists.

Attributes and types. Settings in which agents are partitioned into different types, or derive their preferences based on a set of attributes assigned to each candidate, have been considered for the problems of sampling and counting stable matchings in instances of SM or SR (see, e.g., [5, 11, 12]). In [17], the authors study the problem of characterizing matchings that are rationalisable as stable matchings when agents' preferences are unobserved. They focus on a restricted setting that translates into assigning each agent a type based on several attributes, and assuming that agents of the same type are identical and have identical preferences. They remark that empirical studies on marriage typically make such an assumption [13]. Bounded agent types have been considered in [40, 4] to derive polynomial-time results for the coalition structure generation problem, an important issue in cooperative games when the goal is to partition the participants into exhaustive and disjoint coalitions in order to maximize the social welfare.

3.3 Our Results

Observe that the hardness results discussed in the previous section also hold even in the settings of typed and refined-typed models: if we let $k = n$ (so each type contains just one agent) then the definitions of our models do not place any restrictions on the preference lists. Thus we can deduce that (Refined-)Typed Max SMTI, (Refined-)Typed Max HRT, (Refined-)Typed Max SRT, (Refined-)Typed Max Size Min BP SMI and (Refined-)Typed Max Size Min BA SMI are all NP-complete when k is part of the input.

In contrast with these hardness results, we are able to provided positive results in the parameterised setting when the number of types is taken to be the parameter.

Theorem. (Refined-)Typed Max SMTI, (Refined-)Typed Max HRT, (Refined-)Typed Max SRTI, (Refined-)Typed Max Size Min BP SMTI *and* (Refined-)Typed Max Size Min BA SMTI *all belong to* FPT *when parameterised by the number of types.*

Note that in some cases our FPT result applies to a more general version than that which is known to be NP-hard. These results are presented in Sections 4, 5 and 6.

We also consider a setting where preferences over types are all strict. Certain known hardness results also carry over to this setting if we allow $k = n$: (Refined-)Typed Max Size Min BP SMI and (Refined-) Typed Max Size Min BA SMI remain NP-hard even if all preferences over types are strict.

However, this additional restriction on the input allows us to give polynomial-time algorithms (where k is taken to be part of the input) for (Refined-)Typed Max SMTI, (Refined-)Typed Max HRT and (Refined-)Typed Max SRTI.

Theorem. (Refined-)Typed Max SMTI, (Refined-)Typed Max HRT *and* (Refined-)Typed Max SRTI *are polynomial-time solvable if all preferences over types are strict.*

We prove these results in Sections 4 and 5.

4 Efficient Algorithms for Max SMTI and Max HRT

In this section we first present our fpt-algorithm for the NP-complete problem Typed Max SMTI, and then show how to extend it to solve Refined-Typed Max SMTI and to give a polynomial-time algorithm for the special case in which preferences over types are all strict. We conclude with presenting a straightforward reduction from Max HRT to Max SMTI that implies we can use our ftp-algorithms for the latter to solve the former.

4.1 An fpt-algorithm for Typed Max SMTI

Let I be a typed instance of SMTI, and let M be a matching in I. We may assume without loss of generality that every agent is matched, by creating sufficiently many dummy agents of type $k+1$ which are inserted at the end of each men's and women's (possibly incomplete) preference list. We define $\text{worst}_M(i)$ to be the type of the least desirable agent with which any agent of type i is matched in M. Note that $\text{worst}_M(i)$ would be a dummy type if an agent of type i is unmatched (i.e. matched to a dummy agent) in M. Let $\text{type}(a)$ denote the type of a given agent a.

We claim that, in order to determine whether or not M is stable, it suffices to examine the values of $\text{worst}_M(i)$ for each $i \in [k]$.

Lemma 3. *Let I be a typed instance of SMTI. Then a matching M in I is stable if and only if there is no pair $(i,j) \in [k]^{(2)}$ such that $j \succ_i \text{worst}_M(i)$ and $i \succ_j \text{worst}_M(j)$.*

Proof. Suppose first that M is not stable. In this case, by definition, there exists some pair of agents (a,b) such that a and b are not matched together but each prefers the other over their current partner. Suppose without loss of generality that a is of type i and b is of type j. Then we know that $j \succ_i \text{type}(M(a)) \succeq_i \text{worst}_M(i)$, and similarly $i \succ_j \text{type}(M(b)) \succeq_j \text{worst}_M(j)$.

Conversely, suppose that M is stable, and suppose for a contradiction that $j \succ_i \text{worst}_M(i)$ and $i \succ_j \text{worst}_M(j)$. Then there is some agent a of type i which is matched with an agent of type $\text{worst}_M(i)$, so in particular a is matched with a partner less desirable than any agent of type j. Similarly, there is some agent b of type j which is matched with an agent of type $\text{worst}_M(j)$ and hence is matched with a partner less desirable than any agent of type i. Thus a and b both prefer each other to their current partner, and so form a blocking pair. This contradicts the assumption that M is stable. □

We now use the above observation to prove our main result of this section.

Theorem 4. Typed Max SMTI *is in* FPT *parameterised by the number k of different types in the instance.*

Proof. Lemma 3 implies that every stable matching must pass the test of Lemma 3, and any matching that passes its test is stable. Note that there are at most $(k+1)^k$ possibilities for the function $\text{worst} : [k] \to [k+1]$. We say that a given function worst is *feasible* if for each type i, $\text{worst}(i)$ is either a type acceptable to type i or the dummy type. We say that a matching M *realises* a given feasible function worst if, for each $i \in [k]$, the least desirable partner any agent of type i has in M is of type $\text{worst}(i)$.

Our strategy is to consider each of the feasible possibilities for worst in turn. First we determine (using Lemma 3) whether a matching M realising worst will be stable; this can be done for any candidate function worst in time $\mathcal{O}(k^2)$.

For any feasible function worst which does give rise to a stable matching, we now want to determine what is the maximum number of agents which can be matched in any matching which realises worst. We do this by solving a suitable instance of an Integer Linear Programming formulation. For each unordered pair of distinct values $\{i,j\} \in [k+1]^{(2)}$, the variable $n_{\{i,j\}}$ represents the number of pairs in the matching consisting of one agent of type i and another of type j. Recall that N_i is the set of agents of type i. We then have the following integer linear program:

$$\begin{aligned}
\textbf{maximize} \quad & \sum_{1\leq i,j\leq k} n_{\{i,j\}} & \\
\textbf{subject to} \quad & \sum_{j\in[k+1]} n_{\{i,j\}} = |N_i|, & \forall i \in [k] \\
& \sum_{j\succeq_i \mathrm{worst}(i)} n_{\{i,j\}} = |N_i|, & \forall i \in [k] \\
& n_{\{i,\mathrm{worst}(i)\}} > 0, & \forall i \in [k] \\
\textbf{and} \quad & n_{\{i,j\}} \geq 0. & \forall i,j \in [k+1]
\end{aligned}$$

The first constraint ensures that every agent is involved in exactly one pair, perhaps with a dummy agent. The next two conditions ensure that, for each i, worst(i) is indeed the type of the least desirable partner assigned to any agent of type i. The objective function seeks to maximise the total number of pairs that do not involve dummy agents.

The above integer linear program has $3k+(k+1)^2$ constraints (where each constraint has constant length) and $(k+1)^2$ variables. The upper bound on the absolute value a variable can take is n. Therefore, by Theorem 1, this maximisation problem for any candidate function worst can be solved in time $k^{\mathcal{O}(k^2)} \log^3 n$. □

4.2 An fpt-algorithm for Refined-Typed Max SMTI

To extend the result for Typed Max SMTI to Refined-Typed Max SMTI, we need the following result.

Lemma 5. *Let I be a refined-typed instance of SMTI and suppose that M is a matching in I such that there is no pair $(i,j) \in k^{(2)}$ where $j \succ_i \mathrm{worst}_M(i)$ and $i \succ_j \mathrm{worst}_M(j)$. Then there is a stable matching M' such that, for every $(i,j) \in k^{(2)}$, both M and M' contain the same number of pairs that consist of one agent of type i and another of type j. Moreover, given M, we can compute M' in polynomial time.*

Proof. Given matching M, let $n_{\{i,j\}}(M)$ denote the number of pairs in M consisting of an agent of type i and an agent of type j. We construct a stable matching M' such that $n_{\{i,j\}}(M') = n_{\{i,j\}}(M)$. We first present the construction of M' and then prove that it is stable. Let $\bigcup_{i,j\in[k],i<j} M'_{i,j} = M'$ be a decomposition of M' where $M'_{i,j}$ is the projection of M' onto agent types i and j. The construction of M' takes place in two steps:

- **Step 1:** For all $(i, j) \in k^2$, compute $A_{i,j}$ where $A_{i,j}$ denotes the set of agents of type i that are to be matched to agents of type j.
- **Step 2:** For all $(i, j) \in k^2$, $i < j$, generate $M'_{i,j}$ given $A_{i,j}$ and $A_{j,i}$ computed in Step 1.

We next elaborate on how each of the above steps are executed.

Step 1: To start with, all agents are available. For each type i, take the candidate types in type i's decreasing order of preference with ties broken arbitrarily, $\langle i_1, i_2, \ldots i_{k_i} \rangle$, where i_s denotes the type that is ranked s'th by type i. Starting with $j = i_1$, take the topmost (from the perspective of an agent of type j) $n_{\{i,j\}}(M)$ available agents of type i, and put them in $A_{i,j}$; these agents become unavailable from now on. Note that as the preference lists of agents of type j over agents of type i may include ties, it may not be possible to determine exactly who are the topmost available $n_{\{i,j\}}(M)$ agents in type i. To be more precise, when going down the preference list of agents of type j over available agents of type i, we may reach a tie τ including $z > 1$ available agent where we need to pick $x < z$ number of them. If this happens, arbitrarily pick x agents from τ.

Step 2: We now show how to generate $M'_{i,j}$ given $A_{i,j}$ and $A_{j,i}$ computed in Step 1. We do so by computing a stable matching amongst the agents in $A_{i,j} \cup A_{j,i}$. Since agents of the same type have the same preference ordering over the candidates of the same type, our problem reduces to the problem of computing a stable matching in an instance of SMTI where all men have identical preferences, as well as all women. The latter problem is solvable in polynomial time using a straightforward greedy procedure.

By our construction of M', $n_{\{i,j\}}(M') = n_{\{i,j\}}(M)$. So it only remains to prove that M' is weakly stable. Assume for a contradiction that M' admits a blocking pair (a, b) where a , b, $M'(a)$ and $M'(b)$ are of types i, j, j' and i' respectively. Three "kinds" of blocking pairs are possible, depending on how a and b compare each others' types against the types of their partners. We examine each of them and show that M' can admit none.

- *If a prefers type j to type j', and b prefers type i to type i'.* In the assumption of the lemma we have that given M there is no pair $(i, j) \in k^{(2)}$ where $j \succ_i \text{worst}_M(i)$ and $i \succ_j \text{worst}_M(j)$. By construction of M', $\text{worst}_M(i)$ remains unchanged under M' for all types i. It thus directly follows that M' cannot admit such a blocking pair.
- *If $i = i'$ and $j = j'$.* The existence of such a blocking pair implies that $M'_{i,j}$ constructed in Step 2 is not weakly stable w.r.t. the preferences of agents in $A_{i,j} \cup A_{j,i}$, a contradiction.
- *Either $i = i'$ or $j = j'$, but not both.* Without loss of generality assume that $j = j'$ and b prefers type i to type i'. Since a prefers b to $M'(a)$, and $M'(a) \in A_{j,i}$, therefore it follows the construction in Step 1 that, since b is not in $A_{j,i}$ and is in $A_{j,i'}$, therefore any agent of type j (including b) prefers type i' to type i, a contradiction.

Notice that we have not considered a scenario where $j \simeq_i j'$ $(j \neq j')$ or $i \simeq_j i'$ $(i \neq i')$. Assume that $j \simeq_i j'$, $j \neq j'$. Then, by our assumption that the instance is refined-typed, any agent of type i, and hence a, is indifferent between all agents who are of type j or type j'. Therefore, $M(a) \simeq_a b$ and (a, b) cannot be a blocking pair. A similar argument holds for when $i \simeq_j i'$, $i \neq i'$. □

Let I be a refined-typed instance of SMTI and let I' be a typed instance of SMTI that is obtained from I by ignoring the refined preferences within each type (i.e. every agent is indifferent between the candidates of the same type). It follows from the definition of stability that every matching that is stable in I is also stable in I'. Lemma 5 implies that for any stable matching M in I', there exists a stable matching M' in I of the same cardinality as M. Thus, in order to find a maximum cardinality matching in a refined-typed instance of SMTI, it suffices to (1) solve the typed problem (i.e. ignore the refined preferences within each type) and then (2) use the algorithm provided in the proof of Lemma 5 to convert the solution to a matching of the same cardinality that is stable in the instance I.

Theorem 6. Refined-Typed Max SMTI *is in* FPT *parameterised by the number k of different types in the instance.*

4.3 An fpt-algorithm for Max HRT

A typed instance of Max HRT can be reduced to an instance of Max SMTI as follows. For each type i of hospitals, let N_i denote the set of posts offered by the hospitals of type i, and hence $|N_i|$ the total number of posts offered by the hospitals of type i (i.e. the sum of the capacities of the hospitals of type i). To reduce an instance of Refined-Typed Max HRT to an instance of Refined-Typed Max SMTI, additionally let $A_{i,j}$ denote the set of posts of type i that are matched to residents of type j (if i is a hospital type) or the set of residents of type i that are matched to a post of type j (if i is a resident type). The next result then straightforwardly follows from Theorems 4 and 6.

Corollary 7. Typed Max HRT *and* Refined-Typed Max HRT *are in* FPT *parameterised by the number k of different types in the instance.*

4.4 Strict preferences over types

In the preceding sections we have assumed that agents can be indifferent between the agents of two or more types. Here we investigate the implication of requiring strict preferences over types on the complexity of the stable matching problems we have studied. We show that the problems (Refined) Typed Max SMTI and (Refined) Typed Max HRT are polynomial-time solvable if agents have strict preferences over types; the argument is based on a private communication with David Manlove.

Theorem 8. *When preferences over types are strict, both* Typed Max SMTI *and* Refined-Typed Max SMTI *are polynomial-time solvable. Furthermore, all stable matchings are of the same size.*

Proof. Let I be a typed instance of SMTI, and assume that preferences over types are strict. Let I' be a refined-typed instance of SMTI that is obtained from I by breaking all ties arbitrarily in a consistent way (recall that agents of the same type have identical preferences). In fact I' will be an instance of SMI, as preferences over types are strict.

By Lemma 5, if M is a stable matching in I, then there exists a stable matching M' in I' of the same size. This implies that the size of a maximum cardinality matching in I is the same as the size of some stable matching in I'. However I' is an instance of SMI and all stable matchings of I' are of the same size. Therefore, to solve Max SMTI for I, it is enough to find a stable matching in I', which can be done easily in polynomial time by a simple extension of GS [20]. Additionally, all stable matchings of I are of the same size.

Now let I be a refined-typed instance of SMTI and assume that preferences over types are strict. Let I' be a typed instance of SMTI that is obtained from I by ignoring the refined preferences within each type (i.e. every agent is indifferent between the candidates of the same type). We have proved, in the preceding paragraph, that all stable matchings of I' are of the same size. It thus follows Lemma 5, using a similar argument as in the preceding paragraph, that all stable matchings in I are also of the same size, and thus the maximum cardinality stable matching of I can be found in polynomial time by breaking ties arbitrarily and applying a simple extension of GS [20]. □

The above result, combined with Corollary 7 gives us the following result.

Corollary 9. *When preferences over types are strict,* Typed Max HRT *and* Refined-Typed Max HRT *is polynomial-time solvable. Furthermore, all stable matchings are of the same size.*

5 Efficient Algorithms for Max SRTI

Notice that, in the proofs of Lemma 3, Lemma 5, and Theorems 4 and 6, we have not made use of the fact that SMTI is a bipartite matching problem, except that we have implicitly assumed that no type finds itself acceptable. Therefore these proofs will go through immediately for MAX SRTI if we impose such an assumption. To allow for a type to find itself acceptable, we need to make some modifications, which are explained next.

5.1 An fpt-algorithm for Typed Max SRTI

Let I be a typed instance of MAX SRTI, and let M be a matching in I. We may assume without loss of generality that every agent is matched, using the same argument as in Section 4.1. We define $\text{worst}_M(i)$ as in Section 4.1, and additionally define $\text{second\_worst}_M(i)$ to be the type of the second least desirable agent with which any agent of type i is matched in M. If there is only one agent of type i, then $\text{second\_worst}_M(i)$ is undefined, in which case let $\text{second\_worst}_M(i) = \varnothing$. Note that it is possible to have $\text{second\_worst}_M(i) = \text{worst}_M(i)$.

We claim that, in order to determine whether or not M is stable, it suffices to examine the values of $\text{worst}_M(i)$ and $\text{second\_worst}_M(i)$ for each $i \in [k]$.

Lemma 10. *Let I be a typed instance of SRTI. Then a matching M is stable if and only if (1) there is no pair $(i, j) \in [k]^{(2)}$, $i \neq j$, such that $j \succ_i \text{worst}_M(i)$ and $i \succ_j \text{worst}(j)_M$, and (2) there is no pair (i, i), $i \in [k]$, such that there is at least two agents of type i and $i \succ_i \text{second\_worst}_M(i)$.*

Proof. The proof is similar to the proof of Lemma 3 with slight modification concerning the second case in the statement of the current lemma.

Suppose first that M is not stable. In this case, by definition, there exists some pair of agents (a, b) such that a and b are not matched together but each prefers the other over their current partner. If the agents are of different types then the argument is the same as in Lemma 3. Assume that a and b are both of the same type i. Then we know that type i likes type $\text{type}(M(a))$ or type $\text{type}(M(b))$ at least as well as type $\text{second\_worst}_M(i)$. Without loss of generality assume that $\text{type}(M(a)) \succeq_i \text{second\_worst}_M(i)$. Since (a, b) is a blocking pair, $\text{type}(b) \succ_i \text{type}(M(a))$, and therefore $i \succ_i \text{second\_worst}_M(i)$.

Conversely, suppose that M is stable. The first case where $i \neq j$ is the same as in the proof of Lemma 3. Suppose, for a contradiction, that there exists a type i corresponding to at least two agents where $i \succ_i \text{second\_worst}_M(i)$. Then there are two agents a and b of type i who are matched to an agent of type $\text{worst}_M(i)$ and an agent of type $\text{second\_worst}_M(i)$, respectively. Thus a and b both prefer each other to their current partner, and so form a blocking pair. This contradicts the assumption that M is weakly stable. □

Theorem 11. Typed Max SRTI *is in* FPT *parameterised by the number k of different types in the instance.*

Proof. The proof is similar to the proof of Theorem 4, with some modifications. Lemma 10 implies that every stable matching must pass the test of Lemma 10, and any matching that passes the test of this lemma is stable. Note that there are at most $(\frac{(k+1)(k+2)}{2})^k$ possibilities for the pair of functions $\text{worst} : [k] \to [k+1]$ and $\text{second\_worst} : [k] \to [k+2]$. We say that a given pair of functions worst and second_worst are feasible if for each type i, (1) $\text{worst}(i)$ is either a type acceptable to type i or the dummy type, (2) $\text{second\_worst}(i)$ is either a type acceptable to type i or a dummy type, or $\varnothing$ (only if there is only one agent of type i), and (3) if $\text{worst}(i) = \text{second\_worst}(i)$ then there exists at least two agents of type $\text{worst}(i)$. We say that a matching M *realises* functions worst and second_worst if, for each $i \in [k]$, the least and the second least desirable partner any agent of type i has in M are of types $\text{worst}(i)$ and $\text{second\_worst}(i)$ respectively.

Our strategy is to consider each of the feasible possibilities for worst and second_worst in turn. First we determine (using Lemma 10) whether a matching realising worst and second_worst will be stable; this can be done for any candidate functions worst and second_worst in time $\mathcal{O}(k^2)$. Note that it is possible that none of the candidates functions passes the test of Lemma 10, as not all instances of SRTI necessarily admit a stable matching.

For any pair of feasible functions worst and second_worst which does give rise to a stable matching (if any), we now want to determine what is the maximum number of agents which can be matched in any matching which realises worst and second_worst. We do this by solving a suitable instance of an Integer Linear Programming. The formulation is exactly the same as the one provided in the proof of Theorem 4, with the exception of the following two sets of constraints introduced to ensure that $\text{second\_worst}(i)$ is indeed the type of the second least desirable partner assigned to any agent of type i.

$$\sum_{\text{second\_worst}(i) \succ_i j \succ_i \text{worst}(i)} n_{\{i,j\}} = 0, \qquad \forall i \in [k] \text{ such that } |N_i| > 1$$

$$n_{\{i,\text{worst}(i)\}} + n_{\{i,\text{second\_worst}(i)\}} > 1, \qquad \forall i \in [k] \text{ such that } |N_i| > 1$$

$$n_{\{i,\text{worst}(i)\}} > 1 \qquad \forall i \in [k] \text{ such that } \text{worst}(i) = \text{second\_worst}(i)$$

The above integer linear program has $5k+(k+1)^2$ constraints and $(k+1)^2$ variables. The upper bound on the absolute value a variable can take is n. Therefore, by Theorem 1, this maximisation problem for any candidate function worst can be solved in time $2^{\mathcal{O}(k^2)} \log^3 n$. □

5.2 An fpt-algorithm for Refined-Typed Max SRTI

Using a similar argument as in Section 4.2, and with the aid of the following lemma, we can employ the machinery used to solve Typed Max SRTI to solve Refined-Typed Max SRTI.

Lemma 12. *Let I be a refined-typed instance of SRTI and suppose that M is a matching in I such that (1) there is no pair $(i, j) \in [k]^{(2)}$, $i \neq j$, such that $j \succ_i \text{worst}_M(i)$ and $i \succ_j \text{worst}_M(j)$, and (2) there is no pair (i, i), $i \in [k]$, such that there is at least two agents of type i and $i \succ_i \text{second\_worst}(i)$. Then there is a stable matching M' such that, for every $(i, j) \in [k]^{(2)}$, both M and M' contain the same number of pairs that consist of one agent of type i and another of type j. Moreover, given M, we can compute M' in polynomial time.*

Proof. The proof is similar to that of Lemma 5 with the following modifications.

We let $\bigcup_{i,j \in [k], i \leq j} M'_{i,j} = M'$, to allow for the inclusion of $M'_{i,i}$. Then in Step 1, if $j = i$, we take the topmost $2 \cdot n_{\{i,i\}}(M)$ available agents of type i, and put them in $A_{i,i}$. In Step 2, to generate $M'_{i,i}$ given $A_{i,i}$ we create a complete stable matching amongst the agents in $A_{i,i}$ using the following greedy procedure: Until all agents are matched, take the topmost available tie in type i and match the agents in it together. If the number of agents in the topmost tie is odd, match the remaining agent to one of the agents in the second topmost available tie. Label all the matched agents as unavailable.

To prove that M' is stable, we only additionally need to take care of one type of blocking pair in which two agents a and b of the same type i prefer each other to their partners. In the assumption of the lemma we have that given M there is no pair $(i, i) \in [k]^{(2)}$ where $i \succ_i \text{second\_worst}_M(i)$. By the construction of M', $\text{second\_worst}_M(i)$ remains unchanged under M' for all types i. It thus directly follows that M' cannot admit such a blocking pair. □

Theorem 13. Refined-Typed Max SRTI *is in* FPT *parameterised by the number k of different types in the instance.*

5.3 Strict preferences over types

A similar argument as in the proof of Theorem 8, with small modifications, shows that the non-bipartite versions are also polynomial-time solvable if preferences over types are strict.

Theorem 14. *When preferences over types are strict,* Typed Max SRTI *and* Refined-Typed Max SRTI *are polynomial-time solvable. Furthermore, all stable matchings (if any exists) are of the same size.*

Proof. Let I be a typed instance of SRTI, and assume that preferences over types are strict. Let I' be a refined-typed instance of SRTI that is obtained from I by breaking all ties arbitrarily in a consistent way (recall that agents of the same type have identical preferences). In fact I' will be an instance of SRI, as preferences over types are strict.

Let MAX_SIZE(I) and MAX_SIZE(I') denote the size of the maximum cardinality stable matching in I and I' respectively (which is zero if no stable matching exists). It follows from the definition of stability that every matching that is stable in I' (if any) is also stable in I, implying that MAX_SIZE(I) $\geq$ MAX_SIZE(I'). Lemma 12 implies that for any stable matching M (if any exists) in I, there exists a stable matching M' in I' of the same cardinality as M, implying that MAX_SIZE(I') $\geq$ MAX_SIZE(I). Therefore, MAX_SIZE(I') = MAX_SIZE(I), and in order to find a maximum cardinality matching in I, or reporting that none exits, it is enough to solve Max SRI for I'. The latter problem is polynomial-time solvable [21]. Furthermore, all stable matchings of I' (if I' admits any) are of the same size [21], and hence all stable matchings of I (if I admits any) are also of the same size. □

6 Maximum Cardinality Matchings with Minimum Instability

As stated in Section 1, in some settings the size of the matching takes priority over the stability criteria. That is, the mechanism designers are willing to tolerate a small degree of instability if that leads to a matching of larger size. We can extend the methods from previous sections to prove the following results.

Theorem 15. Typed Max Size Min BP SMTI *belongs to* FPT *when parameterised by the number k of different types in the given instance.*

Theorem 16. Typed Max Size Min BA SMTI *belongs to* FPT *when parameterised by the number k of different types in the given instance.*

We begin by considering the problem of minimising the total number of blocking pairs. For the rest of this section, we assume that types 1 to k' are types of women, and types $k'+1$ to k are types of men. We further assume that all preference lists are extended to include the dummy type as their least desirable acceptable type. If an agent a is unmatched in M, we say that $M(a)$ is of type $k+1$, the dummy type. As a first step, we translate the definition of a blocking pair into the setting of typed instances.

Proposition 17. *Let x and y be agents of type i and type j respectively, and suppose that $M(x)$ is of type j' and $M(y)$ is of type i'. Then (x, y) is a blocking pair in M if and only if $j \succ_i j'$ and $i \succ_j i'$.*

Given this observation, we can obtain an expression for the total number of blocking pairs in M that are comprised of an agent of type i and an agent of type j.

Lemma 18. *The number of blocking pairs (x, y) in M such that x is a woman of type i and y is a man of type j is given by*

$$\left(\sum_{i' \prec_j i} n_{\{i',j\}}\right)\left(\sum_{j' \prec_i j} n_{\{i,j'\}}\right) = \sum_{1 \le i',j' \le k+1} n_{\{i',j\}} n_{\{i,j'\}} \left(\mathbb{1}_{i' \prec_j i} \mathbb{1}_{j' \prec_i j}\right),$$

where $\mathbb{1}_{a \prec_b c}$ is an indicator function that returns one if $a \prec_b c$ and zero otherwise.

Proof. It is easy to verify that the left-hand side and the right-hand side of the equation are equal. For the remainder of the proof, we focus on the left-hand side of the equation. By Proposition 17, we know that the set of blocking pairs consisting of one agent of type i and another of type j is precisely

$$\{(x, y) : \text{type}(M(x)) \prec_i t_j \text{ and } \text{type}(M(y)) \prec_j t_i\}.$$

The cardinality of this set is thus equal to the number of agents of type i that are matched to an agent of type inferior to type j ($\sum_{j' \prec_i j} n_{\{i,j'\}}$) multiplied by the number of agents of type j that are matched to an agent of type inferior to type i ($\sum_{i' \prec_j i} n_{\{i',j\}}$). The result follows immediately. □

Summing over all possibilities for i and j gives the following result.

Lemma 19. *The total number of blocking pairs in M is given by*

$$\sum_{1 \le i,j \le k} \sum_{1 \le i',j' \le k+1} n_{\{i',j\}} n_{\{i,j'\}} \left(\mathbb{1}_{i' \prec_j i} \mathbb{1}_{j' \prec_i j}\right).$$

We can now prove Theorem 15.

Proof of Theorem 15. We begin by computing, in polynomial time, the cardinality $C_{\max}$ of a maximum matching in our instance. Our strategy then is to formulate Max Size Min BP SMTI as an instance of Integer Quadratic Programming. Our goal is to minimise the following objective function

$$\begin{aligned}
&\sum_{1 \le i,j \le k} \sum_{1 \le i',j' \le k+1} n_{\{i',j\}} n_{\{i,j'\}} \left(\mathbb{1}_{i' \prec_j i} \mathbb{1}_{j' \prec_i j}\right) \\
&\qquad = \sum_{\substack{1 \le i,i' \le k' \\ k' < j,j' \le k}} \left(\mathbb{1}_{i' \prec_j i} \mathbb{1}_{j' \prec_i j} + \mathbb{1}_{i \prec_{j'} i'} \mathbb{1}_{j \prec_{i'} j'}\right) \\
&\qquad\qquad + \sum_{\substack{1 \le i \le k' \\ k' \le j,j' \le k}} n_{\{k+1,j\}} n_{\{i,j'\}} \mathbb{1}_{j' \prec_i j} \\
&\qquad\qquad + \sum_{\substack{1 \le i,i' \le k' \\ k' \le j \le k}} n_{\{i',j\}} n_{\{i,k+1\}} \mathbb{1}_{i' \prec_j i} \\
&\qquad\qquad + \sum_{\substack{1 \le i \le k' \\ k' < j \le k}} n_{\{k+1,j\}} + n_{\{i,k+1\}},
\end{aligned}$$

subject to the constraints

$$\forall i \in [k] : \sum_{j \in [k+1]} n_{\{i,j\}} = |N_i|$$
$$\sum_{1 \leq i,j \leq k} n_{\{i,j\}} = C_{\max}.$$

To see that the right-hand side of the objective function equation is equal to the left-hand side of the equation, notice that every pair $(\{i,j\},\{i',j'\}) \in [k]^{(2)}$ appears twice in the summation on the left-hand side: once with $\{i,j\}$ coming from the first sum and $\{i',j'\}$ coming from the second sum, and once the other way around. The former counts the number of blocking pairs (x,y) where x is of type i, y is of type j, and $M(x)$ and $M(y)$ are of types j' and i' respectively. The latter counts the number of blocking pairs (x,y) where x is of type i', y is of type j', $M(x)$ is of type j, and $M(y)$ is of type i. Pairs $(\{i,j\},\{i',j'\})$ where at least one of i,i',j,j' is equal to the dummy type $k+1$ are dealt with separately (in this case there can be at most one blocking pair).

The linear constraints enforce that every agent is involved in exactly one pair (perhaps with a dummy agent), and that the number of pairs that do not involve dummy agents is equal to the maximum possible cardinality of a matching. We can write our objective function in the form x^TQx where x is the vector $(n_{\{1,1\}}, n_{\{1,2\}}, \ldots, n_{\{k,k\}})^T$ and the entry of Q corresponding to $n_{\{i,j'\}}$ and $n_{\{i',j\}}$ is equal to either 0, 1 or 2 depending on how many of the following conditions hold:

1. $j \succ_i j'$ and $i \succ_j i'$, and
2. $j' \succ_{i'} j$ and $i' \succ_{j'} i$.

Thus, by Theorem 2, we have an fpt-algorithm to solve our instance of Max Size Min BP SMTI. □

We now consider the problem of minimising the number of agents which are involved in at least one blocking pair. We start by characterising the conditions under which an agent of a particular type can belong to one or more blocking pairs; this characterisation follows immediately from the definition of a blocking pair.

Lemma 20. *Let x be an agent of type i and assume that $M(x)$ is of type j (which would be a dummy type if x is unmatched). Then x belongs to a blocking pair if and only if there is some agent y of type j' who is paired with an agent of type i' (which would be a dummy type if y is unmatched) such that $i \succ_{j'} i'$ and $j' \succ_i j$.*

Using this characterisation, we can now prove Theorem 16.

Proof of Theorem 16. For any matching M, we can define a collection of at most $k(k+1)/2$ boolean variables $v_{i,j}$ (for $1 \leq i < j \leq k+1$, where type $k+1$ is a dummy type), so that $v_{i,j}$ is true if and only if the matching contains at least one pair involving an agent of type i and an agent of type j (and unmatched agents are considered to be matched with agents of the dummy type $k+1$). For a given matching M, this collection of variables defines a vector $\mathbf{v}_M$ in $\{0,1\}^{k(k+1)/2}$, which we call the *type-signature* of the matching M.

Note that there are at most $2^{k(k+1)/2} = \mathcal{O}(2^{k^2})$ possible type-signatures for a matching; we will consider each possible type-signature $\mathbb{v}$ in turn and determine the minimum number of agents which can be involved in blocking pairs in a maximum matching which has type-signature $\mathbb{v}$ (if a maximum matching with this type-signature exists). Minimising over this set of optimal solutions will give the desired answer.

We now describe how to compute the minimum number of agents involved in blocking pairs in a maximum matching with type-signature $\mathbf{v}$ or else to report that no such maximum matching exists. Our strategy is to encode the problem as an instance of Integer Linear Programming.

First we define the constraints. As usual, we need to ensure that every agent is involved in exactly one pair (potentially involving a dummy agent), and as in the proof of Theorem 15 we need to enforce that the number of pairs that do not involve dummy agents is equal to the maximum cardinality of any matching in our instance. Moreover, we need to make sure that our matching does indeed have type-signature equal to $\mathbf{v}$. This gives rise to the following linear constraints.

$$\begin{aligned}
\forall i \in [k] : \sum_{j \in [k+1]} n_{\{i,j\}} &= |N_i| \\
\sum_{1 \leq i < j \leq k} n_{\{i,j\}} &= C_{\max} \\
\forall 1 \leq i < j \leq k+1 \text{ with } v_{i,j} = 1 : n_{\{i,j\}} &> 0 \\
\forall 1 \leq i < j \leq k+1 \text{ with } v_{i,j} = 0 : n_{\{i,j\}} &= 0.
\end{aligned}$$

Finally, we define our objective function, which captures the number of agents which are involved in at least one blocking pair. By Lemma 20, we know that an agent of type i matched with an agent of type j belongs to a blocking pair if and only if there exist $i \succ_{j'} i'$ and $j' \succ_i j$ such that $v_{i',j'} = 1$. Thus, for a given type-signature $\mathbf{v}$, we can compute for each $1 \leq i, j \leq k+1$ the indicator variable $b_{i,j}$ which takes the value 1 if an agent of type i matched with an agent of type j in a matching with type-signature $\mathbf{v}$ will belong to a blocking pair, and takes the value 0 otherwise. It is now clear that the total number of agents that are involved in at least one blocking pair in the matching is

$$\sum_{1 \leq i < j \leq k+1} n_{\{i,j\}}(b_{i,j} + b_{j,i}).$$

This is our linear objective function. □

We further observe that Theorems 15 and 16 can easily be extended to refined typed instances.

Corollary 21. Refined-Typed Max Size Min BP SMTI *and* Refined-Typed Max Size Min BA SMTI *belong to* FPT *when parameterised by the number of types.*

Proof. We use the same strategy for both problems. First, we solve the problem for the corresponding typed instance (ignoring the more refined preferences within types). Note that the number of blocking pairs or blocking agents achieved in this simplification of the problem clearly gives a lower bound on the minimum number that can be achieved if we take into account all information; we will argue that in fact we can always obtain a matching which does not increase either quantity when we take into account the full preference lists.

To do this, we follow the method described in Lemma 5. Given the number of pairs $n_{\{i,j\}}$ of type i and j for each $1 \leq i < j \leq k$, this method allows us to construct a matching M where, for each $1 \leq i < j \leq k$ we have exactly $n_{\{i,j\}}$ pairs involving one agent of type i and one of type j, and there is no blocking pair (x, y) such that x is currently matched to an agent of the same type as y. Thus the only blocking pairs in M are those of the form (x, y) where x is of type i and y of type j, and $j \succ_i \text{type}(M(x))$ and $i \succ_j \text{type}(M(y))$. But these are precisely the blocking pairs that occur in the relaxation to a typed instance.

Thus we can indeed obtain a solution to Max Size Min BP SMTI or Max Size Min BA SMTI by applying the appropriate algorithm to find the number of pairs of each type under the relaxation to a typed instance, and then use the method of Lemma 5 to extend this to a matching which does not introduce any additional blocking pairs when the full preference lists are taken into consideration. □

Finally, we consider extending these results to other matching problems. We obtain one immediate corollary.

Corollary 22. (Refined-)Typed Max Size Min BP HRT *and* (Refined-)Typed Max Size Min BA HRT *belong to* FPT *when parameterised by the total number of types.*

To generalise to the non-bipartite case takes just slightly more care: if two agents of type i which are both matched to agents of type j form a blocking pair, then the total number of blocking pairs that results is $n_{\{i,j\}}(n_{\{i,j\}} - 1)/2$ rather than $n^2_{\{i,j\}}$. Otherwise, exactly the same method works. Thus we obtain the following corollary.

Corollary 23. (Refined-)Typed Max Size Min BP SRTI *and* (Refined-)Typed Max Size Min BA SRTI *belong to* FPT *when parameterised by the total number of types.*

It is also fairly straightforward to modify the IQP and ILP in the proofs of Theorem 15 and 16 to solve MIN BP SRTI and MIN BA SRTI respectively. We only need to remove the constraint that enforces the matching to be of size $C_{\max}$.

Corollary 24. (Refined-)Typed Min BP SRTI *and* (Refined-)Typed Min BA SRTI *belong to* FPT *when parameterised by the total number of types.*

A related problem to MIN BP SR is EXACT BP SR which given an instance I of SR and an integer Z, decides whether I admits a matching with exactly Z blocking pairs. Even this problem is NP-hard [1] in general. If I is a (Refined-)Typed instance, to solve EXACT BP SRTI we only have to move the objective function in the IQP of Theorem 15 to the set of constraints, and enforce it to be equal to Z.

Corollary 25. (Refined-)Typed EXACT BP SRTI *belongs to* FPT *when parameterised by the total number of types.*

7 Hospitals/Residents problem with Couples

The job market for medical residents underwent a change since mid 1970's, due to married couples seeking posts in nearby hospitals. Subsequently, the central matching systems had to be adapted to take into account *couple's preferences*, as otherwise couples would seek to arrange their own matches outside the centralized clearinghouse. The *Hospitals/Residents problem with Couples (HRC)* models such settings, where couples can send in preferences over pairs of hospitals. The size of a given matching in an instance of HRC is equal to the number of residents matched in the matching.

Definition 6. STABLE HRC *is the problem of deciding whether an instance of HRC admits a stable matching.*

Definition 7. MAX HRC *is the problem of identifying a maximum cardinality stable matching in an HRC instance, or reporting that none exists.*

STABLE HRC is NP-complete, and as a corollary so is MAX HRC. The result holds even if each hospital has capacity 1 and there are no single residents [38]. The parameterized complexity of STABLE HRC has been studied in [35, 6], where [35] also studies MAX HRC. STABLE HRC is W[1]-hard when the problem is parametrized by the number of couples [35]. The result holds even if each hospital has capacity 1. However, STABLE HRC belongs to FPT when the problem is parametrized by the number of couples in the instance and the hospitals' list are derived from a strictly-ordered master list of residents [6].

In the remainder of this section, we first provide a stability definition for instances of HRC, then extend our TYPED and REFINED-TYPED models to such instances, and conclude with presenting an fpt-algorithm for TYPED MAX HRC.

7.1 Stability in HRC

In an instance of HRC, the set of residents R includes an even size subset R' consisting of those residents who belong to couples, where every resident in R' belongs to exactly one couple. Let C denote the set of ordered pairs (r_i, r_j) where r_i and r_j form a couple. Single residents and hospitals find a subsets of candidates acceptable and rank them in strict order of preference. Every couple $(r_i, r_j) \in C$ has a joint strict preference ordering over an acceptable subset of ordered hospital pairs.

Different stability definitions for an instance of HRC have been provided in the literature, most of them distinct from one another (see, e.g., [6, 28, 35, 36, 9, 16, 27, 39]). In this paper we adopt the definition of [36] and provide it as it has been given in [31]. We say that a hospital h_j is undersubscribed if $|M(h_j)| < q(h_j)$. Given an instance of HRC, a matching M is stable if it admits no blocking pair, where a blocking pair satisfies one of the following properties:

1. it involves a single resident r_i and a hospital h_j where (a) r_i prefers h_j to $M(r_i)$ and (b) h_j is undersubscribed or prefers r_i to its worst assigned resident in M;

2. it involves a couple $(r_i, r_j) \in C$ and a hospital h_k such that either

(a) (r_j, r_j) prefers $(h_k, M(r_j))$ to $(M(r_j), M(r_k))$, and h_k is undersubscribed or prefers r_i to a resident in $M(h_k) \setminus \{r_j\}$; or

(b) (r_j, r_j) prefers $(M(r_i), h_k)$ to $(M(r_j), M(r_k))$, and h_k is undersubscribed or prefers r_j to a resident in $M(h_k) \setminus \{r_i\}$;

3. it involves a couple $(r_i, r_j) \in C$ and a pair of (not necessarily distinct) hospitals h_k, h_ℓ such that $h_k \neq M(r_i)$, $h_\ell \neq M(r_j)$, (r_j, r_j) prefers (h_k, h_ℓ) to $(M(r_j), M(r_k))$, and either

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

   (b) $h_k = h_\ell$ and $q(h_k) - |M(h_k)| \geq 2$; or

   (c) $h_k = h_\ell$ and $q(h_k) - |M(h_k)| \geq 1$ and h_k prefers at least one of r_i, r_j to one of its assigned residents; or

   (d) $h_k = h_\ell$, $q(h_k) = |M(h_k)|$, h_k prefers r_i to some resident $r_s \in M(h_k)$, and prefers r_j to some resident in $M(h_k)$ distinct from r_s.

7.2 Typed Instances

In an instance of HRC, each type i that is associated with a single resident or a hospital has a preference ordering over types of the candidates, similarly to that defined earlier for instances of HRT. We assume, without loss of generality, that each pair of residents that form a couple are ordered such that the type of the first one is no larger than the type of the second one. That is, for each $(x, y) \in C$ where x is of type i and y is of type j, it is the case that $i \leq j$. Each pair of types (i, j) (with $i \leq j$) associated with a couple has a joint preference ordering over ordered pairs of "hospital" types.

The same conditions outlined in Section 3.1.1 apply to single residents and hospitals. That is, (1) every two single residents or hospitals of the same type have identical preference lists, and (2) all single residents are indifferent between hospitals of the same type, and all hospitals are indifferent between residents of the same type.

These notions extend naturally to the preference lists of couples:

1. if (x, y) and (x', y') are couples of the same type (i, j) then (x, y) and (x', y') have identical preference lists, and

2. all couples are indifferent between (h_s, h_t) and (h_k, h_ℓ) if h_s and h_k are hospitals of the same type, say i, and h_t and h_ℓ are hospitals of the same type, say j.

7.3 An fpt-algorithm for Typed MAX HRC

We adapt the method used to prove Theorems 4 and 11 in order to show that Typed Max HRC is in FPT when parameterised by the number of different types in the instance. The idea, once again, is to (1) consider a number of possibilities for the matching, where this number of possibilities is bounded by a function of k, (2) determine for each such possibility whether a matching which meets the conditions will be stable, and then (3) express the maximisation problem associated with each set of stable conditions as an instance of Integer Linear Programming.

As in previous sections, we assume without loss of generality that there is at least one agent of each type; however we cannot be sure that there will always be a *single* resident of a given type, or a couple of a specific pair of types. Also, as in previous sections, we assume that every agent is matched by creating sufficiently many dummy agents of type $k+1$ which are inserted at the end of each single resident's and hospital's (possibly incomplete) preference list; we insert $(k+1, k+1)$ at the end of each couple's preference list.

We start by providing conditions that are necessary and sufficient for a matching to be stable in a given typed instance of HRC. Given a matching M (in which every agent is matched, perhaps to a dummy agent), we define three functions worst_M, second_worst_M and assigned_M. The function worst_M is defined as follows:

- for any resident type i of which there is at least one single agent, $\text{worst}_M(i)$ is the type of the least desirable hospital (with respect to the preference list for i) to which any *single* resident of type i is assigned in M;
- for any pair of resident types i and j (with $i \leq j$) such that at least one couple has type (i, j), $\text{worst}_M(i, j)$ is the least desirable pair of hospital types (with respect to the joint preference list for (i, j)) to which any couple of types i and j are assigned in M;
- for any hospital type p, $\text{worst}_M(p)$ is the type of the least desirable resident (with respect to the preference list for p) assigned to a hospital of type p in M.

The function second_worst_M is only defined for hospital types where the total capacity of hospitals of the type is at least two: $\text{second\_worst}_M(p)$ is the type of the second least desirable resident assigned to any hospital of type p in M. The boolean function assigned_M is defined for each combination of a pair of resident types and a pair of hospital types: $\text{assigned}_M((i, j), (\ell, p)) = 1$ if and only if there is at least one couple involving residents of type i and j who are assigned to hospitals of types ℓ and p respectively.

For the remainder of this section, we assume that types 1 to k' are types of residents, and types $k'+1$ to k are types of hospitals. Let $(k' : k]$ denote the set $\{k'+1, \dots k\}$. Let $|N_i|$ (with $i \in [k']$) denote the number of single residents of type i, $|N_p|$ (with $p \in (k' : k]$) denote the total capacity of hospitals of type p, and $|N_{i,j}|$ (with $i, j \in [k']$ where $i \leq j$) the number of couples where the first resident is of type i and the second one is of type j.

Lemma 26. *Let I be a typed instance of HRC. Then a matching M is stable if and only if none of the following holds:*

1. *There is a pair of types (i, p), $i \in [k'], p \in (k' : k]$ such that $|N_i| > 0$, $p \succ_i \text{worst}_M(i)$ and $i \succ_p \text{worst}_M(p)$.*
2. *There is a couple-type (i, j) with $|N_{i,j}| > 0$ and a hospital type ℓ such that either*
   - *(a) $i \succ_\ell \text{worst}_M(\ell)$ and there is a pair of hospital types (p, q) such that $\text{assigned}_M((i, j), (p, q)) = 1$ and $(\ell, q) \succ_{i,j} (p, q)$, or*
   - *(b) $j \succ_\ell \text{worst}_M(\ell)$ and there is a pair of hospital types (p, q) such that $\text{assigned}_M((i, j), (p, q)) = 1$ and $(p, \ell) \succ_{i,j} (p, q)$.*

3. *There is a couple-type* (i, j) *with* $|N_{i,j}| > 0$ *and two hospital types* ℓ *and* p *(where* $\ell \neq p$*) such that (i)* $(\ell, p) \succ_{i,j} \text{worst}_M(i, j)$*, (ii)* $i \succ_\ell \text{worst}_M(\ell)$*, and (iii)* $j \succ_p \text{worst}_M(p)$*;*

4. *There is a couple-type* (i, j) *with* $|N_{i,j}| > 0$ *and a hospital type* ℓ *such that (i)* $(\ell, \ell) \succ_{i,j} \text{worst}_M(i, j)$*, (ii)* $i, j \succ_\ell \text{worst}_M(\ell)$*, and either*
   - *(a)* $i \succ_\ell \text{second\_worst}(\ell)$*, or*
   - *(b)* $j \succ_\ell \text{second\_worst}(\ell)$*.*

Proof sketch. It is straightforward to prove this claim using a similar approach as in the proof of Lemma 3 and the definition of a stable matching for an instance of HRC. □

Theorem 27. Typed Max HRC *is in* FPT *when parameterised by the number* k *of different types in the instance.*

Proof. Lemma 26 gives a necessary and sufficient condition for a matching realising the functions worst, second_worst and assigned to be stable. We consider each of the feasible possibilities for the functions worst, second_worst and assigned in turn and determine, using Lemma 26, whether a matching realising them will be stable; this can be done for any set of candidate functions in time $\mathcal{O}(k^3)$.

For any trio of feasible functions worst, second_worst and assigned that give rise to a stable matching (if any), we need to determine the maximum number of residents that can be matched in any matching that realises the trio, and we do this (as in the rest of the paper) by solving an instance of Integer Linear Programming.

We define variables $n_{i,p}$ and $n_{(i,j),(p,q)}$ as follows. For each pair of types (i, p), $i \in [k'], p \in (k' : k]$, we have a variable $n_{i,p}$ which denotes the number of *single* residents of type i matched to hospitals of type p. Additionally, for each combination of a pair of resident types (i, j) (with $i \leq j$) and a pair of hospital types (p, q) (where p and q may or may not be distinct), we have a variable $n_{(i,j),(p,q)}$ which denotes the number of couples consisting of residents of types i and j respectively such that the resident of type i is assigned to a hospital of type p and the resident of type j is assigned to a hospital of type q. Our objective is to maximise the size of the matching hence to maximise:

$$\sum_{i \leq k', k' < p \leq k} n_{i,p} + \sum_{i,j \leq k', k' < p,q \leq k} 2 \cdot n_{(i,j),(p,q)}$$

Our first set of constraints ensures that every agent is involved in exactly one pair. The first constraint handles single agents, the second one the hospitals, and the last one couples.

$$\sum_{k' < p \leq k} n_{i,p} = |N_i|, \qquad \forall i \in [k']$$

$$\sum_{i \leq k'} n_{i,p} + \sum_{i,j \leq k', k' < q \leq k} (n_{(i,j),(p,q)} + n_{(i,j),(q,p)}) = |N_p|, \qquad \forall p \in (k' : k]$$

$$\sum_{k' < p,q \leq k} n_{(i,j),(p,q)} = |N_{i,j}|, \qquad \forall i, j : i \leq j \leq k'$$

The next set of constraints ensures that function worst complies with its definition.

$$\sum_{p \succeq_i \text{worst}(i)} n_{i,p} = |N_i|, \qquad \forall i \in [k'] : |N_i| > 0$$

$$n_{i,\text{worst}(i)} > 0, \qquad \forall i \in [k'] : |N_i| > 0$$

$$\sum_{i \succeq_p \text{worst}(p)} n_{i,p} + \sum_{\substack{i,j \succeq_p \text{worst}(p) \\ k' < q \leq k}} (n_{(i,j),(p,q)} + n_{(i,j),(q,p)}) = |N_p|, \qquad \forall p \in (k' : k]$$

$$n_{\text{worst}(p),p} > 0, \qquad \forall p \in (k' : k]$$

$$\sum_{(p,q) \succeq_{(i,j)} \text{worst}(i,j)} n_{(i,j),(p,q)} = |N_{i,j}|, \qquad \forall i, j : i \leq j \leq k' : |N_{i,j}| > 0$$

$$n_{(i,j),\text{worst}(i,j)} > 0, \qquad \forall i, j : i \leq j \leq k' : |N_{i,j}| > 0$$

The following set of constraints ensures that function second_worst for the hospitals complies with its definition.

$$\sum_{\text{second\_worst}(p) \succ_p i \succ_p \text{worst}(p)} n_{i,p} = 0, \qquad \forall p \in (k' : k] : |N_p| > 1$$

$$n_{\text{worst}(p),p} + n_{\text{second\_worst}(p),p} > 1, \qquad \forall p \in (k' : k] : |N_p| > 1$$

$$n_{\text{worst}(p),p} > 1 \qquad \forall p \in (k' : k] : |N_p| > 1 \text{ and } \text{second\_worst}(p) = \text{worst}(p)$$

And finally, the last set of constraints ensure that boolean variables assigned are set correctly.

$$n_{(i,j),(p,q)} > 0, \qquad \forall i, j \in [k'] \forall p, q \in (k', k] \text{ such that } \text{assigned}((i,j)(p,q)) = 1$$

The above integer linear program has $\mathcal{O}(k^4)$ variables and $\mathcal{O}(k^2)$ constraints. The upper bound on the absolute value a variable can take is n. Therefore, by Theorem 1, this maximisation problem for any trio of candidate functions worst, second_worst and assigned can be solved in time $2^{\mathcal{O}(k^2)} \log^3 n$. □

8 Types in practice

In this section we discuss some issues relating to the applicability of our models to real world matching problems. We begin in Section 8.1 by discussing how an instance with relatively few types could arise if agents derive their preference lists from a small set of attributes of other agents. In Section 8.2 we discuss how the same ideas can be applied to consider a relaxation of the stability requirement in a more realistic setting in which agents only feel strongly about a small collection of attributes, but may refine these preferences arbitrarily. Finally, in Section 8.3, we explain how we can efficiently identify agents of the same type if we only have access to the agents' ordinal preference lists.

8.1 Attributes

It seems believable in many settings (particularly those involving very many agents) that agents will derive their preferences by considering some collection of attributes of the other agents, where the number of attributes considered is likely to be much smaller than the total number of agents. For example, in a centralised job allocation scheme, employers might rank applicants based on a number of criteria from the application form e.g. exam grade, interview score etc. In the setting of hospitals/residents allocation, a junior doctor might rank hospitals based on the programs they offer, their reputation on the quality of their programs, their geographic location etc. A similar observation has been made in [5, 11, 12, 17].

We assume that there is a set $\mathcal{B} = \{B_1, \dots, B_r\}$ of attributes and each attribute B_i can take one of u_i distinct possible values (perhaps including a "not applicable" value if different subsets of attributes are relevant for different agents); we write u for $\max_{1 \leq i \leq r} u_i$. The *attribute profile* of an agent is an r-tuple which lists its value for each of the r attributes that is relevant to the agent. Note that there are at most $\prod_{i=1}^{r} u_i \leq u^r$ possible attribute profiles for an agent.

Since agents form their preferences solely on the basis of the attributes of other agents, every agent is necessarily indifferent between two agents which have the same attribute profile. Thus we can express each agent's preferences as an ordering of the possible attribute profiles, allowing for some attribute profiles to be declared unacceptable. This implies that (up to the reordering of agents involved in ties) there are at most $u^r \cdot (u^r)!$ possible preference lists. As we can capture all relevant information about an agent by specifying its attribute profile and its ordering of the possible attribute profiles, we can therefore obtain a typed instance with at most $u^{2r} \cdot (u^r)!$ *types*, by grouping together agents who have the same attribute profile and preferences over other attribute profiles.

In certain models, it might be reasonable to make a stronger assumption, namely that an agents' preferences are also determined by their own attributes: for example, in a stable roommates problem, agents might be looking for partners who are similar to themselves in certain ways. If agents rank others based solely on their own attributes, this then means that the number of types is bounded by u^r rather than $u^r \cdot (u^r)!$.

8.2 A relaxation of the stability requirement

While attributes may well play an important role in determining agents' preferences, the situation described in Section 8.1, where only a very small number of attributes (taking few distinct values) are relevant, might be too simplified to reflect reality in many situations. In this section we discuss how the same ideas may be applied in a much wider range of settings if we relax our notion of stability.

It is reasonable to assume that a certain amount of effort is required by both agents in a blocking pair to make a private arrangement outside the matching, and so agents are unlikely to make this effort for a very small improvement in their utility. We say that a blocking pair is *dangerous* if both agents can make a significant improvement in their utility (where we will define what it means to make a significant improvement later). The goal is then to find a maximum cardinality matching which contains no dangerous blocking pairs.

In our attribute-based model (with potentially many different attributes), there are two natural ways to quantify the relative utility of two partners: one is to consider which attributes can be improved by switching partners, and another is to consider by how much these attributes can be improved. It seems reasonable to assume that:

1. each agent considers some (small) subset of attributes to be *important*, and will only make a private arrangement to improve the value of an important attribute, and

2. the possible values for each attribute can split into a small number of *groups* of similar values, and an agent will only make a private arrangement to obtain a partner whose attribute value comes from a superior group.

If we further assume that there are relatively few attributes that are important to one or more agents (in particular we do not have a large number of agents with disjoint sets of important attributes), we can define a simpler problem instance which captures coarse information about the original instance, and crucially contains enough information to determine whether there will be any dangerous blocking pairs in a matching. This idea can be formalised as follows.

We denote by $\mathcal{A}_I$ the set of t attributes which are important to at least one agent, and for each $A_i \in \mathcal{A}_I$ we split the values into groups $G^i_1, \dots, G^i_{s_i}$. We write $s = \max_{A_i \in \mathcal{A}_I} s_i$.

We now define the *coarse attribute profile* of an agent to be a t-tuple which lists the group to which the agent's value belongs for each attribute in $\mathcal{A}_I$. Based on our assumptions above, we know that an agent will only make a private arrangement with a new partner whose coarse attribute profile is preferable to that of their current partner. Note that there are at most s^t possible coarse attribute profiles, and that there are at most $(s^t)!$ possible rankings of coarse attribute profiles. We now group agents into types so that two agents are assigned to the same type if and only if they have the same coarse attribute profile and the same preferences over coarse attribute profiles; the number of types is at most $s^t(s^t)!$.

If we simplify the instance I by assuming that all agents are in fact indifferent between any two agents with the same attribute profile, we obtain a typed instance I'. Of course, a matching which is stable in this simplified instance I' need not be stable in the original instance I; however, it is easy to check that a matching M is weakly stable in I' if and only if M contains no *dangerous* blocking pair in I. Thus, if we are willing to accept matchings which are not stable but which contain no dangerous blocking pairs, we can use the machinery developed to deal with typed instances in this more general setting.

8.3 Identifying types from preference lists

In order to make use of the algorithms we have developed for typed and refined-typed instances, we need to know the type of each agent, but in practice we may only be presented with lists of ordinal preferences for each agent. In this section we describe how we can compute the coarsest partition into types which meet the conditions of either of the models.

8.4 Finding types in a typed instance

We identify the type of each agent in two stages.

In the first step, we sort the list of agents by their preference lists; this makes it easy to identify which agents have identical preference lits, which is one of the requirements for agents having the same type. We write $x \sim_{\text{pref}} y$ if agents x and y have the same preference list.

Next, we consider each pair of agents (x, y) such that $x \sim_{\text{pref}} y$ (that is, pairs (x, y) where we have not yet found evidence that x and y cannot have the same type). We then consider the preference lists of all other agents, and for each list check whether x and y belong to the same tie. If there is any agent which strictly prefers x to y (or vice versa) then x and y cannot have the same type; otherwise we write $x \sim y$.

The types are then taken to be the equivalence classes with respect to the equivalence relation $\sim$.

8.5 Finding types in a refined-typed instance

The process of identifying types for refined-typed instances is only slightly more complicated. Again, we require that all agents of the same type have identical preferences, so we can once again begin by sorting preference lists to determine the pairs for which $x \sim_{\text{pref}} y$. We call each equivalence class with respect to $\sim_{\text{pref}}$ a *group*. We now refine the partition as follows by making a number of passes through the list of preference lists.

On every pass, we consider each preference list in turn, and check for each group whether its members occur consecutively on the preference list. For every group where this is not the case, we split the group into the maximal subsets which occur consecutively in the preference list being considered, then move onto the next preference list with this new, refined partition. We continue this process until we complete a pass through the preference lists in which we do not split any group; when this occurs we stop with the current partition. At this point, it is clear that we have obtained a partition that satisfies the requirements that agents of the same type have identical preferences and appear consecutively in all other agents' lists.

It remains to observe that we do not have to complete too many passes to find such a partition. Note that, on each pass in which we do split some group, we must increase the number of parts in the partition by at least one. Therefore, if the instance can be split into a set of k types satisfying the conditions for a refined-typed instance, we cannot possibly make more than k passes.

9 Summary and Future Work

We considered a setting in which agents are partitioned into k different *types*, and the type of an agent determines his or her preferences, as well as how s/he is compared against other agents. Agents have preferences over types, which may be refined by more detailed preferences within a single type, and agents of the same type have identical preferences. We showed that in this setting several important NP-hard stable matching, namely MAX SMTI, MAX HRT, MAX SRTI, MAX SIZE MIN BP/BA SMTI, MAX SIZE MIN BP/BA SRTI, and MIN BP SRTI belong to the parameterised complexity class FPT when parameterised by the number of different types of agents, and so admit efficient algorithms when this number of types is small. We were further able to prove that MAX SMTI, MAX HRT and

MAX SRTI are polynomial-time solvable when agents have strict preferences over types. Additionally, we were able to show MAX HRC is in FPT parameterised by the number of different types of agents, when agents are indifferent between the agents of the same type.

It would be interesting to investigate whether our approach might yield fpt-algorithms for other NP-hard stable matching problems. so that agents of the same type have "similar" preference lists, but not necessarily identical. Another intriguing question would be to understand how the complexity of the stable matching problems we have studied changes when agents on only one side of the market are associated with types.

Acknowledgements. The first author is supported by a Personal Research Fellowship from the Royal Society of Edinburgh (funded by the Scottish Government). Both authors are extremely grateful to David Manlove for his insightful comments on a preliminary version of this manuscript.

References

[1] D.J. Abraham, P. Biró, and D.F. Manlove. "Almost stable" matchings in the Roommates problem. In *Proceedings of WAOA '05: the 3rd Workshop on Approximation and Online Algorithms*, volume 3879 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[2] D.J. Abraham, A. Levavi, D.F. Manlove, and G. O'Malley. The Stable Roommates Problem with Globally-Ranked Pairs. In *Proceedings of WINE '07: the 3rd International Workshop on Internet and Network Economics*, volume 4858 of *Lecture Notes in Computer Science*, pages 431–444. Springer, 2007.

[3] D.J. Abraham, A. Levavi, D.F. Manlove, and G. O'Malley. The Stable Roommates Problem with Globally-Ranked Pairs. *Internet Mathematics*, 5(4):493–515, 2008. Preliminary version appeared as [2].

[4] H. Aziz and B. de Keijzer. Complexity of coalition structure generation. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '11, pages 191–198, 2011.

[5] N. Bhatnagar, S. Greenberg, and D. Randall. Sampling stable marriages: why spouse-swapping won't work. In *Proceedings of SODA '08: the 19th ACM/SIAM Symposium on Discrete Algorithms*, pages 1223–1232. ACM-SIAM, 2008.

[6] P. Biró, R.W. Irving, and I. Schlotter. Stable matching with couples: an empirical study. *ACM Journal of Experimental Algorithmics*, 16, 2011. Section 1, article 2, 27 pages.

[7] P. Biró, D.F. Manlove, and S. Mittal. Size versus stability in the marriage problem. Technical Report TR-2008-283, University of Glasgow, Department of Computing Science, 2008.

[8] P. Biró, D.F. Manlove, and S. Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411:1828–1841, 2010.

[9] D. Cantala. Matching markets: the particular case of couples. *Economics Bulletin*, 3(45):1–11, 2004.

[10] P. Chebolu, L.A. Goldberg, and R. Martin. The complexity of approximately counting stable matchings. In *Proceedings of APPROX '10: the 13th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 6302 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2010. Full version available at `http://arxiv.org/abs/1004.1836`.

[11] P. Chebolu, L.A. Goldberg, and R. Martin. The complexity of approximately counting stable matchings. *Theoretical Computer Science*, 437:35–68, 2012. Preliminary version appeared as [10].

[12] P. Chebolu, L.A. Goldberg, and R. Martin. The complexity of approximately counting stable roommate assignments. *Journal of Computer and System Sciences*, 78(5):1579–1605, 2012.

[13] E. Choo and A. Siow. Who marries whom and why. *Journal of Political Economy*, 114(1):175–201, 2006.

[14] M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.

[15] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer London, 2013.

[16] B. Dutta and J. Massó. Stability of matchings when individuals have preferences over colleagues. *Journal of Economic Theory*, 75:464–475, 1997.

[17] F. Echenique, S. Lee, M. Shum, and M. B. Yenmez. The revealed preference theory of stable and extremal stable matchings. *Econometrica*, 81(1):153–171, 2013.

[18] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[19] A. Frank and É Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[20] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

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

[22] K. Hamada, K. Iwama, and S. Miyazaki. An improved approximation lower bound for finding almost stable maximum matchings. *Information Processing Letters*, 109(18):1036–1040, 2009.

[23] R.W. Irving. On the stable room-mates problem. Technical Report CSC/86/R5, University of Glasgow, Department of Computing Science, 1986.

[24] R.W. Irving and D.F. Manlove. The Stable Roommates Problem with Ties. *Journal of Algorithms*, 43:85–105, 2002.

[25] R.W. Irving, D.F. Manlove, and S. Scott. The stable marriage problem with master preference lists. *Discrete Applied Mathematics*, 156(15):2959–2977, 2008.

[26] R. Kannan. Minkowskis convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[27] B. Klaus, F. Klijn, and T. Nakamura. Corrigendum: Stable matchings and preferences of couples. *Journal of Economic Theory*, 144(5):2227–2233, 2009.

[28] F. Kojima, P.A. Pathak, and A.E. Roth. Matching with couples: Stability and incentives in large matching markets. *Quarterly Journal of Economics*, 128(4):1585–1632, 2013.

[29] H.W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[30] D. Lokshtanov. Parameterized integer quadratic programming: Variables and coefficients. arXiv:1511.00310 [cs.DS], 2015.

[31] D.F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[32] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. Technical Report TR-1999-43, University of Glasgow, School of Computing Science, 1999.

[33] D.F. Manlove, R.W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002. Full version available as [32].

[34] D. Marx and I. Schlotter. Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica*, 58(1):170–187, 2010.

[35] D. Marx and I. Schlotter. Stable assignment with couples: parameterized complexity and local search. *Discrete Optimization*, 8:25–40, 2011.

[36] E.J. McDermid and D.F. Manlove. Keeping partners together: Algorithmic results for the hospitals / residents problem with couples. *Journal of Combinatorial Optimization*, 19(3):279–303, 2010.

[37] G. O'Malley. *Algorithmic Aspects of Stable Matching Problems*. PhD thesis, University of Glasgow, Department of Computing Science, 2007.

[38] E. Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11:285–304, 1990.

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

[40] T. Shrot, Y. Aumann, and S. Kraus. On agent types in coalition formation problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, pages 757–764, 2010.