Strategy-Proofness in the Stable Matching Problem with Couples

Andrew Perrault, Joanna Drummond, and Fahiem Bacchus
Department of Computer Science
University of Toronto, Toronto, CANADA
{perrault, jdrummond, fbacchus}@cs.toronto.edu

ABSTRACT

Stable matching problems (SMPs) arising in real-world markets often have extra complementarities in the participants' preferences. These complementarities break many of the theoretical properties of SMP and make it computationally hard to find a stable matching. A common complementarity is the introduction of couples in labor markets, which gives rise to the stable matching problem with couples (SMP-C). A major concern in markets is strategy-proofness since markets that are easily manipulated often unravel. In this paper we provide some key insights into the issue of strategy-proofness in SMP-C. We provide theoretical results that relate the set of resident Pareto optimal stable matchings ($\mathcal{RP}_{opt}$) admitted by an SMP-C instance to the ability of the residents to manipulate. We show that a mechanism returning an $\mathcal{RP}_{opt}$ matching is, in certain cases, strategy-proof against residents attempting to manipulate by truncating their preference lists. We provide an algorithm for finding an $\mathcal{RP}_{opt}$ matching when one exists. And finally, we study empirically the frequency of multiple stable and multiple $\mathcal{RP}_{opt}$ matchings as the market sizes grows, and under different proportions of couples in the market. Our empirical results indicate that SMP-C becomes less susceptible to manipulation as both the size of the market grows and the fraction of couples in the market shrinks.

General Terms

Economics, Algorithms, Theory

Keywords

Stable Matching; Complementarities; Strategy-proofness; SAT

1. INTRODUCTION

The Stable Matching Problem (SMP), in which two groups wish to be matched with each other, is one of the most widely-studied problems in economics [11, 7]. A number of real-world applications can be cast as SMP, with residency matching being one the most well-known [18, 22, 1]. The National Resident Matching Program (NRMP) in the US, and the Scottish Foundation Allocation Scheme (SFAS) in Scotland, are two real-world instantiations of such matching markets [17, 9]. The SMP framework however does not allow for complementarities between the participants' preferences. In resident matching an important complementarity arises from couples among the residents: a resident's preferences can depend on the match received by their partner. To address this need for couples to coordinate their placement, residency matching markets (including the NRMP and SFAS) began allowing couples to express their preferences over residency programs jointly. This extension to SMP is called the Stable Matching Problem with Couples.

As in SMP, the goal of SMP-C is to find a match such that no pair (one from each side of the market) has an incentive to defect from their assignment. A matching with this property is said to be *stable*. In standard SMP, there always exists a resident-optimal stable matching ($\mathcal{R}_{opt}$) in which every resident is as least as well off as in any other stable matching; and the simple Deferred Acceptance (DA) algorithm [6] can find this $\mathcal{R}_{opt}$ matching in polynomial time. These properties no longer hold for SMP-C. Finding a stable matching in SMP-C becomes NP-complete [20]; a stable matching is not guaranteed to exist; and even when one does exist, an $\mathcal{R}_{opt}$ matching might not exist.

While there are many strategy-proofness results for SMP, little work has investigated strategy-proofness in SMP-C. For SMP, it is known that no stable matching mechanism is strategy-proof for both sides of the market [21]. There are, however, mechanisms that are approximately stable and approximately strategy-proof where the approximation becomes better as the market size grows [10]. Furthermore, in SMP an $\mathcal{R}_{opt}$ matching always exists and a mechanism returning it is known to be strategy-proof against manipulation by the residents [6].

Since SMP-C extends SMP there also can be no stable mechanism for SMP-C that is strategy-proof for both sides of the market. Furthermore, the limit results of [8, 10] are not known to hold for SMP-C. To the authors' knowledge, no investigation of strategy-proofness against residents in SMP-C has been published. This is a gap in the literature since strategy-proofness on one side of the market is frequently cited as an important property for real world matching mechanisms; e.g., there is evidence that mechanisms lacking this property are often abandoned in favor of strategy-proof ones [2].

In this paper we investigate the question of strategy-proofness against residents in SMP-C. We show that in general, no resident strategy-proof mechanism exists for SMP-C, but that there is a mechanism that is resident strategy-proof with respect to truncations when the SMP-C instance has

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

an $\mathcal{R}_{opt}$ matching. Furthermore, we provide an algorithm for implementing this mechanism. When no $\mathcal{R}_{opt}$ matching exists this mechanism will return a resident Pareto optimal ($\mathcal{RP}_{opt}$) matching (as long as the instance has at least one stable matching). We show that residents who have a unique match among the set of $\mathcal{RP}_{opt}$ matchings cannot manipulate this mechanism by truncating their preferences. We also show, however, that no stable mechanism can be strategy-proof when the residents can use the more general manipulation of reordering their preferences, even in the setting where a unique stable matching exists.

We empirically evaluate the set of stable and $\mathcal{RP}_{opt}$ matchings in instances drawn from two different synthetic markets (previously developed by Kojima et al. and Biró et al. [13, 15]). For problem instances of similar size and percentage of couples as the NRMP data investigated by Roth and Peranson [23], we find that nearly all problem instances admit an $\mathcal{R}_{opt}$ matching, and the DA-style algorithm presented by Roth and Peranson almost always finds that $\mathcal{R}_{opt}$ matching. Thus, in conjunction with our theoretical work, we provide an alternate hypothesis for behaviour seen by Roth and Peranson [23]: any mechanism that returns a $\mathcal{R}_{opt}$ matching will be resident strategy-proof with respect to truncations. We additionally find that the theoretical result of Immorlica and Mahdian [8] for large markets appears to hold for SMP-C, even though that result was only proved for SMP.

Section 2 formally defines SMP-C and $\mathcal{RP}_{opt}$ matches. Section 3 discusses strategy-proofness in SMP-C and provides some theoretical results for this question. Section 4 describes our algorithms for finding $\mathcal{RP}_{opt}$ matchings in SMP-C. These algorithms are extensions of a prior SAT based algorithm for SMP-C [5]. Section 5 provides our empirical results, investigating properties of the set of $\mathcal{RP}_{opt}$ matchings as the size and percentage of couples in the market varies. We conclude in Section 6.

2. BACKGROUND

The stable matching problem with couples (SMP-C) can be formalized in terms of the residency matching problem [23]. In this problem doctors wish to be matched with a hospital residency program,[1] and programs wish to accept some number of residents. Both doctors and programs have preferences over who they are matched with, expressed as ranked order lists (ROLs). Some doctors are members of a couple, and these couples provide a joint ROL.[2] Both doctors' and programs' ROLs can be incomplete: any alternative not listed in their ROL is considered to be unacceptable. That is, they would rather not be matched at all than be matched to an alternative not on their ROL. The SMP-C problem is to find a *stable* matching, such that no doctor-program pair has an incentive to defect from the assigned matching.

2.1 SMP-C

More formally, let D be a set of doctors and P be a set of programs. Since there is a preference to be unmatched over an unacceptable match, we use *nil* to denote this alternative: matching a program p to *nil* indicates that p has an unfilled slot while matching a doctor d to *nil* indicates that d was not placed into any program. Let D^+ and P^+ denote the sets $D \cup \{nil\}$ and $P \cup \{nil\}$ respectively.

[1]Each residency program is in a specific medical speciality.
[2]Members of a couple often pursue different specializations and so must apply to different programs.

The doctors are partitioned into two subsets, $S \subseteq D$ and $D \setminus S$. S is the set of single doctors and $D \setminus S$ is the set of doctors who are in couple relationships. Couples are specified by a set of pairs $C \subseteq (D \setminus S) \times (D \setminus S)$. If $(d_1, d_2) \in C$ we say that d_1 and d_2 are each other's partner. We require that every doctor who is not single (i.e., every doctor in $D \setminus S$) have one and only one partner in C. Each program $p \in P$ has an integer capacity $cap_p > 0$ specifying the maximum number of doctors p can accept.

Everyone participating in the matching market has preferences over their possible matches. Each participant a specifies their preferences in a ROL, rol_a, which lists a's preferred matches from most preferred to least preferred. The ROLs of single doctors $d \in S$ contain programs from P^+; the ROLs of couples $c \in C$ contain pairs of programs from $P^+ \times P^+$; and the ROLs of programs $p \in P$ contain doctors from D^+. Every ROL is terminated by *nil* (couple ROLs are terminated by (nil, nil)) since being unmatched is always the least preferred option, but is preferred to any option not on the ROL.

The order of items on a's ROL defines a partial ordering relation where $x \succeq_a y$ indicates that a prefers x to y (x appears before y on a's ROL) or $x = y$. We define $\succ_a$, $\preceq_a$, and $\succ_a$ in terms of $\succeq_a$ and equality in the standard way. We say that x is *acceptable* to a if $x \succeq_a nil$. (Note that unacceptable matches are not ordered by $\succeq_a$ and that *nil* is always acceptable.)

We define a choice function $Ch_p()$ for programs $p \in P$. Given a set of doctors R, $Ch_p(R)$ returns the subset of R that p would prefer to accept. In our setting $Ch_p(R)$ returns the maximal subset of R such that for all $d \in Ch_p(R)$, $d \succ_p nil$, for all $d' \in R - Ch_p(R)$, $d \succ_p d'$, and $|Ch_p(R)| \leq cap_p$. That is, all doctors in $Ch_p(R)$ are acceptable, are strictly preferred to all doctors not chosen, and p's capacity is not violated. It is also convenient to give the null program a choice function as well: $Ch_{nil}(O) = O$, i.e., *nil* will accept any and all matches.

We use the notation $ranked(a)$ to denote the set of options that a could potentially be matched with. For single doctors d and programs p this is simply the ROLs of d (rol_d) and p (rol_p). For a doctor that is part of a couple (d_1, d_2), $ranked(d_1) = \{p_1 | \exists p_2.(p_1, p_2) \in rol_{(d_1,d_2)}\}$ and similarly for d_2. Note we always have that $nil \in ranked(a)$.

Finally, we will sometimes use rol_a as an indexable vector (zero-based). We define the function $rank_a(x)$ to find the index i of x in a's rol: $rank_a(x) = i$ iff $rol_a[i] = x$. When $x \notin rol_a$ we let $rank_a(x) = |rol_a|$ (an out of bounds index).

2.2 Stable Matchings

Definition 1. A **matching** μ is a mapping from D to P^+. We say that a doctor d is matched to a program p under μ if $\mu(d) = p$, and that p is matched to d if $d \in \mu^{-1}(p)$.

We want to find a **stable** matching where no doctor-program pair has an incentive to defect. We call the pairs that do have an incentive to defect blocking pairs. First we define the condition $willAccept(p, R, S)$ to mean that given the set of doctors $R \cup S$, p will accept a set that includes R: $willAccept(p, R, S) \equiv R \subseteq Ch_p(S \cup R)$.

Definition 2. Let μ be a matching.

1. A single doctor $d \in S$ and a program $p \in P$ is a **blocking pair** for μ if $p \succ_d \mu(d)$ and *willAccept*$(p, \{d\}, \mu^{-1}(p))$.
2. A couple $c = (d_1, d_2) \in C$ and a program pair $(p_1, p_2) \in P^+ \times P^+$ with $p_1 \neq p_2$ is a **blocking pair** for μ if and only if $(p_1, p_2) \succ_{(d_1,d_2)} (\mu(d_1), \mu(d_2))$, *willAccept*$(p_1, \{d_1\}, \mu^{-1}(p_1))$, and *willAccept*$(p_2, \{d_2\}, \mu^{-1}(p_2))$.
3. A couple $c = (d_1, d_2)$ and a program $p \in P$ is a **blocking pair** for μ if $(p, p) \succ_{(d_1,d_2)} (\mu(d_1), \mu(d_2))$ and *willAccept*$(p, \{d_1, d_2\}, \mu^{-1}(p))$.

Definition 3. A matching μ is **individually rational** if (a) for all $d \in S$, $\mu(d) \succeq_d nil$, (b) for all $c = (d_1, d_2) \in C$, $(\mu(d_1), \mu(d_2)) \succeq_c (nil, nil)$, (c) for all $d \in \mu^{-1}(p)$, $d \succeq_p nil$, and (d) for all $p \in P$, $|\mu^{-1}(p)| \leq cap_p$.

Definition 4. A matching μ is **stable** if it is individually rational and no blocking pairs for μ exist.

2.3 Resident Preferred Matchings

The set of stable matchings can be quite large. In SMP (where there are no couples) this set is always non-empty [6] and has a nice structure: it forms a lattice under the partial order $\succeq_{\mathcal{R}}$ defined as follows.

Definition 5. A matching μ_1 is **resident preferred** to another matching μ_2 [11], denoted by $\mu_1 \succeq_{\mathcal{R}} \mu_2$, if for all $a \in S \cup C$ we have that $\mu_1(a) \succeq_a \mu_2(a)$. We also define $\succ_{\mathcal{R}}$, $\prec_{\mathcal{R}}$, and $\preceq_{\mathcal{R}}$ in terms of $\succeq_{\mathcal{R}}$ and equality in the standard way. In particular, $\mu_1 \succ_{\mathcal{R}} \mu_2$ whenever $\mu_1 \succeq_{\mathcal{R}} \mu_2$ and for at least one $a \in S \cup C$ we have that a strictly prefers μ_1 to μ_2 ($\mu_1(a) \succ_a \mu_2(a)$). When $\mu_1 \succ_{\mathcal{R}} \mu_2$ we say that μ_1 **dominates** μ_2.

Definition 6. We say that a matching μ is **resident optimal**, written $\mathcal{R}_{opt}(\mu)$ if μ is stable and it dominates all other stable matches. That is, for all stable matches μ' not equal to μ we have that $\mu \succ_{\mathcal{R}} \mu'$.

Note that we restrict the resident-optimal matching to be stable. It can be observed that when a matching μ is resident optimal ($\mathcal{R}_{opt}(\mu)$) then for any $a \in S \cup C$, $\mu(a)$ is the best match for a offered by *any* stable matching.

Resident optimality is generally cited as an important property for stable matching algorithms (e.g., [6, 23]). In SMP the fact that the stable matchings form a lattice under $\succeq_{\mathcal{R}}$ implies that a resident-optimal matching always exists. In the presence of couples however, stable matchings may not exist and even when they do an $\mathcal{R}_{opt}$ matching might not exist. However, the $\succeq_{\mathcal{R}}$ relation is still well defined, and for SMP-C leads to potentially multiple *resident Pareto-optimal* matchings.

Definition 7. We say that a matching μ is **resident Pareto optimal**, written $\mathcal{RP}_{opt}(\mu)$, if μ is stable and there does not exist another stable matching μ' such that $\mu' \succ_{\mathcal{R}} \mu$.

It is easy to see that in SMP-C every stable matching μ is either an $\mathcal{RP}_{opt}$ or is dominated by an $\mathcal{RP}_{opt}$ matching. This also means that an SMP-C instance has an $\mathcal{R}_{opt}$ matching if and only if it has a unique $\mathcal{RP}_{opt}$ matching.

2.4 Mechanisms and Strategy-Proofness

Definition 8. A **mechanism** for SMP-C is any algorithm that takes an SMP-C instance as input and returns a matching. A **stable mechanism** is a mechanism that always returns a stable matching if one exists and otherwise returns the empty matching (i.e., everyone is matched to *nil*).[3] A $\mathcal{P}$ **mechanism** is one that always returns a matching satisfying property $\mathcal{P}$ if one exists and the empty matching otherwise.

Definition 9. Let rol^*_α be the complete, true preferences of resident or couple α. rol^*_α consists of a number of programs or program pairs, then the *nil* program, and then the remaining programs or program pairs. α is **truthful** if their reported rol_α is the same as rol^*_α up to and including the *nil* program. α **manipulates** if they are not truthful. Any manipulation is a **reordering** of rol^*_α. A manipulation is a **truncation at rank i** if the reported rol_α is the first i elements of rol^*_α followed by *nil*.

Definition 10. A mechanism m for SMP-C is **resident strategy-proof** when for every SMP-C instance being truthful is a dominant strategy for every resident and couple. That is, no resident or couple in any SMP instance can improve their matching under m by manipulating. When no resident or couple can improve their matching using only truncation manipulations, we say that m is **resident strategy-proof against truncation**.

Definition 11. Let α be a resident (couple) in an SMP-C instance I. A program p (pair of programs) is a $\mathcal{P}$ **program** for α in I if there exists a matching μ for I such that μ satisfies property $\mathcal{P}$ and $p = \mu(\alpha)$.

3. STRATEGY-PROOFNESS IN SMP-C

The current NRMP mechanism was developed partly in response to concerns that the previous mechanism could be manipulated by residents by misreporting their ranking of programs [23]. In SMP, it is possible to achieve *strategy-proofness* for either side of the market by using a mechanism that returns the resident or program-optimal matching[4] [25]. The administrators of the NRMP perceived that residents had a much higher average difference in utility between adjacent elements on their *rol*; thus, the NRMP decided that they wanted a mechanism that was as close to resident strategy-proof as possible.

Roth and Peranson hypothesized that since there is a relatively small fraction of couples in the NRMP (4% in their historical data), the resident strategy-proof mechanism for SMP that returns the $\mathcal{R}_{opt}$ matching could be generalized to SMP-C and the result would be similar [23]. Although their mechanism was not in fact an $\mathcal{R}_{opt}$ mechanism it was quite successful empirically—less than 10 residents (out of 30,000) had an incentive to manipulate via truncation. While not all manipulations are truncations, they tested for truncations for several reasons: i) any outcome that can be achieved by manipulation in SMP can be achieved by a truncation [26] ii) profitable truncations are computationally inexpensive to check for, and iii) truncations are the kind of manipulations that can be potentially identified with the least information about others' preferences [24].

To our knowledge, no theoretical strategy-proofness results exist for SMP-C. We hypothesized that if we restricted

[3]An SMP-C instance might have no stable matchings, so we cannot define a stable mechanism to be one that **always** returns a stable matching as in [21].

[4]Strategy-proofness for both sides of the market is only possible for a small number of instances of SMP [21].

our attention to SMP-C instances in which an $\mathcal{R}_{opt}$ matching exists, then an $\mathcal{R}_{opt}$ mechanism would be resident strategy-proof. As shown in Theorem 2 below, we found that this is not the case. We did find, however, that such a mechanism is resident strategy-proof against truncations. This is useful in practice. In particular, as we will show in Sec. 5, an $\mathcal{R}_{opt}$ matching frequently exists in SMP-C instances that have a low proportion of couples. Furthermore, truncations are an attractive way to manipulate because of the computational and informational properties mentioned above. Hence, an $\mathcal{R}_{opt}$ mechanism can at least block this simpler form of manipulation in many practical cases.

We begin with a lemma that establishes a limit on the ability of truncating agents to benefit from manipulation.

Lemma 1. *Let α be an agent who truncates their preferences at rank i in an SMP-C instance $\mathcal{I}$. Let Ω be the set of stable matchings before the truncation and let Ω' be the set of stable matchings after the truncation. For any $\mu \in \Omega'$, either i) $\mu \in \Omega$ or ii) $\mu(\alpha) = nil$.*

Proof. Let μ' be a matching in Ω' where α is not matched to *nil*. It must be the case that $\mu'(\alpha)$ is ranked above i; otherwise $\mu'(\alpha)$ would be unacceptable to α and μ' would be unstable. Since rol_α is the same as rol^*_α above rank i and the rank order lists of all of the other agents and programs in the instance are the same, μ' must also be stable before α truncates. Thus $\mu' \in \Omega$. $\square$

Theorem 1. *Let α be a resident or couple in an SMP-C instance $\mathcal{I}$. For any property $\mathcal{P}$, if α has a unique $\mathcal{P}$ program in $\mathcal{I}$, then for any $\mathcal{P}$ mechanism $y_\mathcal{P}$, α cannot improve its matching under $y_\mathcal{P}$ using only truncation manipulations (α has no incentive to manipulate under $y_\mathcal{P}$).*

Proof. The theorem says that there exists a program (or pair of programs) p such that for all $\mathcal{P}$ matchings μ we have that $\mu(\alpha) = p$, so before truncation $y_\mathcal{P}$ returns p as α's match. After truncation, by Lemma 1, $y_\mathcal{P}$ must return either p or *nil* as α's match. Neither of these improve α's match. $\square$

A corollary is that in an instance of SMP-C that has an $\mathcal{R}_{opt}$ matching, a $\mathcal{R}_{opt}$ mechanism is resident strategy-proof against truncation. However, such mechanisms are not resident strategy-proof, as reordering manipulations can still be beneficial.

Theorem 2. *Let $y_{\mathcal{R}_{opt}}$ be an $\mathcal{R}_{opt}$ mechanism for SMP-C. Then $y_{\mathcal{R}_{opt}}$ is strategy-proof against residents who manipulate via truncation. However, $y_{\mathcal{R}_{opt}}$ is not strategy-proof against residents who manipulate via reordering.*

Proof. Part I: For any SMP-C instance $\mathcal{I}$, if $\mathcal{I}$ has an $\mathcal{R}_{opt}$ matching then every resident has a unique $\mathcal{R}_{opt}$ program. Then by Theorem 1 no resident has an incentive to manipulate $y_{\mathcal{R}_{opt}}$ using truncation. Otherwise $y_{\mathcal{R}_{opt}}$ always returns the empty match, and again no resident has an incentive to manipulate.

Part II: Reordering is not strategy-proof. We provide a counterexample in Figure 1. The preferences provided show residents' and programs' true preferences. Couples are identified by their resident-resident pair, and give joint preferences as expected. Given these true preferences, only one stable matching exists: $\mu(r_0) = c$, $\mu((r_1, r_2)) = (b, e)$, and $\mu((r_3, r_4)) = (a, d)$. However, even though only one stable matching exists (and thus this matching is $\mathcal{R}_{opt}$), the single resident r_0 has an incentive to manipulate via reordering. Instead of reporting $a \succ b \succ c \succ d$ (r_0's true preferences) r_0 can be matched to b instead of c by reporting $b \succ d \succ c \succ a$. (All other participants in the market report their true preferences.) The resulting matching is $\mu'(r_0) = b$, $\mu'((r_1, r_2)) = (a, d)$, and $\mu'((r_3, r_4)) = (c, e)$. μ' was not stable under the original preferences, and single resident r_0 is better off than when they reported their true preferences. $\square$

Resident preferences	Program preferences
$r_0 : a \succ b \succ c \succ d$	$a : r_3 \succ r_0 \succ r_1$
$(r_1, r_2) : (b, e) \succ (a, d)$	$b : r_1 \succ r_0$
$(r_3, r_4) : (a, d) \succ (c, e)$	$c : r_3 \succ r_0$
	$d : r_0 \succ r_2 \succ r_4$
	$e : r_4 \succ r_2$

Figure 1: An instance that is manipulable by r_0 using reordering under any mechanism. Each program has capacity 1.

Resident preferences	Program preferences
$(r_0, r_1) : (d, b) \succ (a, c)$	$a : r_0 \succ r_2 \succ r_4$
$(r_2, r_3) : (e, c) \succ (b, d) \succ (a, c)$	$b : r_2 \succ r_1$
$(r_4, r_5) : (a, c) \succ (e, nil)$	$c : r_1 \succ r_3 \succ r_5$
	$d : r_0 \succ r_3$
	$e : r_4 \succ r_2$

Figure 2: An instance that is manipulable for truncating residents under any stable mechanism. Each program has capacity 1.

We can strengthen the latter result of Theorem 2. Using the same instance from that proof, we can show that:

Theorem 3. *No stable mechanism for SMP-C is resident strategy-proof.*

Proof. Since the instance of Figure 1 has only one stable matching before and after r_0 reorders, any stable mechanism is manipulable by r_0. $\square$

Thus, reorderings are a powerful way of manipulating in SMP-C. Reorderings can remove old stable matchings and create new ones allowing great scope for beneficial manipulations. In fact, for SMP-C, strategy-proofness is a very strong requirement due to diversity of instances. Hence, we can further strengthen Theorem 3 to show that even less general manipulations by truncations suffice to foil strategy-proofness for stable mechanisms.

Theorem 4. *No stable mechanism for SMP-C is resident strategy-proof against truncations.*

Proof. Figure 2 shows an instance that can be manipulated by couple (r_0, r_1) or couple (r_2, r_3). This instance has two stable matchings, μ and μ':

$$\mu((r_0, r_1)) = (a, c), \mu((r_2, r_3)) = (b, d), \mu((r_4, r_5)) = (e, nil)$$
$$\mu'((r_0, r_1)) = (d, b), \mu'((r_2, r_3)) = (a, c), \mu'((r_4, r_5)) = (e, nil)$$

Resident preferences	Program preferences
$(r_0, r_1) : (d, b) \succ (a, c) \succ (g, h)$	$a : r_0 \succ r_2$
$(r_2, r_3) : (a, f) \succ (b, e)$	$b : r_8 \succ r_2 \succ r_2 \succ r_7 \succ r_1$
$(r_4, r_5) : (c, e) \succ (d, f)$	$c : r_1 \succ r_4$
$(r_6, r_7) : (d, b) \succ (g, h)$	$d : r_4 \succ r_0 \succ r_6$
$(r_8, r_9) : (b, g)$	$e : r_3 \succ r_5$
	$f : r_5 \succ r_3$
	$g : r_0 \succ r_6 \succ r_9$
	$h : r_7 \succ r_1$

Figure 3: An instance where the $\mathcal{RP}_{opt}$ mechanism is weak to truncations. (r_0, r_1) manipulates and each program has capacity 1.

(r_0, r_1) prefers μ' to μ and (r_2, r_3) prefers μ to μ'. By truncating their rank-order list at rank 1, each couple can guarantee that their preferred stable matching is selected by the mechanism. Thus, any stable mechanism is manipulable by at least one of the two couples. $\square$

In this situation it is unclear what mechanism to use for SMP-C when an $\mathcal{R}_{opt}$ matching does not exist. By extending the analogy with SMP, it would be intuitive for $\mathcal{RP}_{opt}$ mechanisms to be harder to manipulate via truncations than other stable mechanisms. However, Lemma 1 states that a truncating agent a may create new stable matchings where they are matched to *nil*. A stable mechanism might return one of these matching. Therefore, a might have a disincentive to manipulate by truncation because the mechanism might now return a matching in which a is unmatched. However, these new matches of a to *nil* might not be $\mathcal{RP}_{opt}$. In this case they would not be returned by an $\mathcal{RP}_{opt}$ mechanism. Hence, with an $\mathcal{RP}_{opt}$ mechanism a might no longer have a disincentive to manipulate. Figure 3 shows an instance where this occurs. Initially, there are four stable matchings and two $\mathcal{RP}_{opt}$ matchings (not shown in the figure). By truncating, (r_0, r_1) can eliminate one of the $\mathcal{RP}_{opt}$ matchings while also creating a new stable matching μ where they are matched to (nil, nil). However, μ is not $\mathcal{RP}_{opt}$, and so (r_0, r_1) can freely manipulate an $\mathcal{RP}_{opt}$-mechanism without fear of being matched to (nil, nil). Hence there are stable mechanisms that are strategy-proof against truncations for this instance, but some $\mathcal{RP}_{opt}$ mechanisms are not.

Despite this difficulty in comparing the strategy-proofness of $\mathcal{RP}_{opt}$ mechanisms with other stable mechanisms, we will show in Sec. 5 that empirically Theorem 1 can be used to show that often a larger number of residents will have no incentive to manipulate an $\mathcal{RP}_{opt}$ mechanism than a stable mechanisms.

4. ALGORITHMS FOR SMP-C

The standard approach to finding a stable matching in SMP-C has been to extend the deferred acceptance algorithm so that it can handle couples (e.g., [23, 12, 3]). However, these extensions are incomplete: they are unable to determine whether or not a stable matching exists, and even when a stable matching does exist they might not be able to find one.

Since SMP-C is known to be NP-Complete [20] it is also possible to encode it as another NP-Complete problem. For example, it can be encoded as a SAT (satisfiability) problem or as IP (integer program) problem [5, 4]. The advantage of doing this is that SAT and IP solvers have become very advanced and are routinely able to solve large practical problems. Another advantage of these solvers is that when given an instance for which no stable matching exists, they are often able to prove this.

4.1 DA-Style Algorithms for SMP-C

The basic principle of DA algorithms [6] is that members of one side of the market propose down their ROLs while the other side either rejects those proposals or holds them until they see a better proposal: once all proposals have been made the non-rejected proposals are accepted forming a match.

Roth and Peranson develop a DA algorithm, **RP99**, capable of dealing with couples [23]. This algorithm has been used with considerable success in practice, including most famously for finding matches for the NRMP which typically involves about 30,000 doctors [16]. RP99 employs an iterative scheme. After computing a stable matching for all single doctors, couples are added one at a time and a new stable matching computed after each addition. The algorithm uses DA at each stage to find these stable matchings. Matching a couple can make previously made matches unstable and in redoing these matches the algorithm might start to cycle. Hence, cycle checking (or a timeout) is sometimes needed to terminate the algorithm.

Kojima et al. develop a simple "sequential couples algorithm," which they use to show that the probability of a stable matchings existing goes to one under certain assumptions [12]. However, this simple algorithm is not useful in practice as it declares failure under very simple conditions. Kojima et al. also provide a more practical DA algorithm, **KPR**, that they use in their experiments. The main difference between KPR and RP99 is that KPR deals with all couples at the same time—it does not attempt to compute intermediate stable matchings. Drummond et al. found that KPR was much more efficient than RP99 [5].

Finally, Ashlagi et al. extend the analysis of Kojima et al., developing a more sophisticated "Sorted Deferred Acceptance Algorithm" and analyzing its behaviour [3]. This algorithm is designed mainly to be amenable to theoretical analysis rather than for practical application.

4.2 Solving SMP-C via SAT

Drummond et al. developed SAT-E, an effective encoding of SMP-C into SAT, and evaluated its performance on synthesized SMP-C problem instances [5]. They found that their encoding, used in conjunction with a state-of-the art SAT solver, scaled well, outperformed an IP encoding they also developed, and could find solutions to problem instances that the DA-style algorithms could not.

Given an SMP-C instance $\langle D, C, P, ROLs\rangle$, where $ROLs$ is the set of all participant ROLs, SAT-E($\langle D, C, P, ROLs\rangle$) can be viewed as a function that returns a SAT encoding in CNF (conjunctive normal form), which is the input format taken by modern SAT solvers.

We highlight three important things about SAT-E:

1. For any SMP-C instance $\mathcal{I} = \langle D, C, P, ROLs\rangle$, the satisfying models of SAT-E stand in a one-to-one correspondence with the stable models of $\mathcal{I}$.
2. SAT-E includes the set of propositional variables $m_d[p]$.

ALGORITHM 1: SAT-$\mathcal{RP}_{opt}$. Given an SMP-C instance return a $\mathcal{RP}_{opt}$ matching or the empty match if none exists.

```
   Input: $\mathcal{I} = \langle D, C, P, ROLs \rangle$ an SMP-C instance.
   Output: An $\mathcal{RP}_{opt}$ matching for $\mathcal{I}$.
1  CNF $\leftarrow$ SAT-E($\mathcal{I}$)
2  $\mu \leftarrow \emptyset$
3  while true do
4      (sat?,$\pi$) $\leftarrow$ SatSolve(CNF)
       /* SatSolve returns the status (sat or unsat) and a
           satisfying model $\pi$ if sat                    */
5      if not sat? then
6          return $\mu$                // Return last match found.
7      $\mu \leftarrow$ stable matching corresponding to $\pi$
8      $c \leftarrow \{\neg m_d[p] \mid \mu(d) = p\}$            // Block this match
9      CNF $\leftarrow$ CNF $\cup \{c\}$
10     for $d \in S$ do    //$d$ must get an equally good match
11         $c_d \leftarrow \{m_d[p] \mid p \succeq_d \mu(d)\}$
12         CNF $\leftarrow$ CNF $\cup \{c_d\}$
13     for $c \in C$ do    //$c$ must get an equally good match
14         $c_c \leftarrow \{m_c[i] \mid i = rank_c(\mu(c))\}$
15         CNF $\leftarrow$ CNF $\cup \{c_c\}$
```

In any satisfying model, π, $m_d[p]$ is true if and only if $\mu(d) = p$ in the stable matching μ corresponding to π.

3. SAT-E also includes the set of propositional variables $m_c[i]$. In any satisfying model, π, $m_c[i]$ is true if and only if in μ, the stable matching corresponding to π, c is matched to a program pair they rank i or above in their ROL.

4.3 Finding $\mathcal{RP}_{opt}$ matchings

To implement an $\mathcal{RP}_{opt}$ mechanism we need to find an $\mathcal{RP}_{opt}$ matching. Using SAT-E and the above three facts, we provide SAT-$\mathcal{RP}_{opt}$ for finding an $\mathcal{RP}_{opt}$ matching. This algorithm will also return an $\mathcal{R}_{opt}$ matching if one exists (as an $\mathcal{R}_{opt}$ matching exists if and only if there is there is only one $\mathcal{RP}_{opt}$ matching).

SAT-$\mathcal{RP}_{opt}$ takes an SMP-C instance as input and constructs the SAT-E encoding for that instance. It finds a stable matching μ and then tries to find a new matching that dominates μ. This is accomplished by adding a *blocking clause* c to the SAT encoding. This clause c is a disjunction that says that no future solution is allowed to return the same stable model (one of the mappings $\mu(d) = p$ of all future solutions must be different, i.e., one of the variables $m_d[p]$ made true by μ must be false in every future matching). Along with the blocking clause, other clauses are added to ensure that in the next matching no doctor or couple receives a worse matching. For single doctors d a clause is added that says that d must be matched to a program it ranks at least as high as $\mu(d)$, and for every couple c a (unit) clause that says that c must be matched to a pair of programs it ranks at least as highly as $\mu(c)$. This causes the new match to be $\succeq_{\mathcal{R}} \mu$, and since the new match cannot be equal it must be $\succ_{\mathcal{R}} \mu$. That is, the new match must dominate μ. If no dominating match can be found, the current match is $\mathcal{RP}_{opt}$ and we return it. Note that an instance has no $\mathcal{RP}_{opt}$ matching if and only if it has no stable matching. So if the very first SAT call fails, there is no $\mathcal{RP}_{opt}$ matching and the empty matching will be returned.

ALGORITHM 2: SAT-ENUM. Find all stable models of an inputted SMP-C instance

```
   Input: $\mathcal{I} = \langle D, C, P, ROLs \rangle$ an SMP-C instance
   Output: Find all stable models of $\mathcal{I}$
1  CNF $\leftarrow$ SAT-E($\mathcal{I}$)
2  while true do
3      (sat?,$\pi$) $\leftarrow$ SatSolve(CNF)
4      if sat? then
5          $\mu \leftarrow$ stable matching corresponding to $\pi$
6          output($\mu$)
7          $c \leftarrow \{\neg m_d[p] \mid \mu(d) = p\}$
8          CNF $\leftarrow$ CNF $\cup \{c\}$                // Block $\mu$
9      else
10         return              // All stable matchings found.
```

In our experiments (Sec. 5) we also need to find all the stable matchings of an instance. This can be accomplished with SAT-ENUM. SAT-ENUM uses a sequence of SAT calls each one returning a stable match. After each stable match is found we ensure that no future stable match is the same by adding a blocking clause to the SAT encoding as described for Algorithm 1.

5. EMPIRICAL RESULTS

We use SAT-ENUM to find all stable matchings for two distributions of synthetic data, which we call: i) impartial culture with geography (IC-GEOG), and ii) SFAS-tuned with geography (SFAS-GEOG). IC-GEOG generates heterogeneous preference profiles; each resident has an equal chance of drawing any program as her first choice. SFAS-GEOG generates more homogeneous profiles; certain programs are much more likely to be in a given resident's top 10. The models have slightly different procedures for generating couples preferences, but both models assign each program a geographic region and require that, for any program pair in a couple's ROL, the programs are in the same region. We describe the models in detail below. We then evaluate how frequently these instances admit $\mathcal{R}_{opt}$ and unique stable matchings for various parameters of the markets.

5.1 Statistical Models of SMP-C Instances

IC-GEOG is the model presented Kojima et al. [12, 13]. It draws residents' and programs' preferences from the impartial culture model (i.e., i.i.d. uniform). All residents draw i.i.d. an ROL of size 10. For singles, this is their final ROL after appending *nil*. To generate a couple's joint ROL, all 11^2 pairs are scored via a Borda-like scoring function, breaking ties according to an arbitrarily chosen member of the couple's preferences. However, as a couple's preferences are constrained by geography, a pair of programs is only added to the joint ROL if both programs are in the same geographic region (or one is *nil*). Each program is randomly assigned one of 5 regions. Each program ranks each resident that ranks them, again with preferences i.i.d. uniform.

SFAS-GEOG is a variant of the model presented by Biró et al. to mimic SFAS [15]. All residents draw i.i.d. a ROL of size 10 from a Plackett-Luce model [19, 14], where the most popular program (resp., resident) is five times more popular than the least popular program (resp., resident) as seen in the SFAS market. All other programs' popularity are linearly interpolated between the most and least popu-

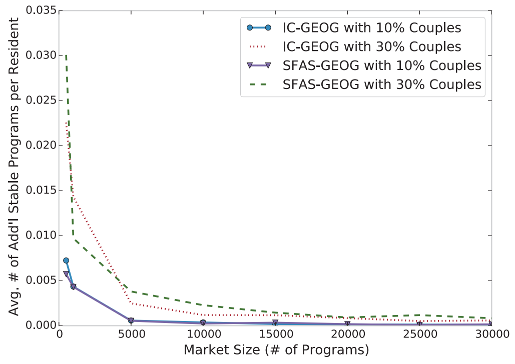

Figure 4: Average number of additional stable programs per resident as market size increases, only including satisfiable instances.

lar programs. Formally, given a reference ranking over all programs (resp., residents), the value of the ith element in the corresponding scoring vector is $m + 4i$, where m is the number of programs (resp., residents) in the match. For singles, this is their final ROL. For couples, all program-program pairs in the same geographic region are ranked in reverse lexicographic ordering, with arbitrary tie-breaking. Each program is randomly assigned one of 5 regions.

In order to ensure that the underlying preference distribution was responsible for any variance shown in the experiments, SFAS-GEOG is modified from Biró et al.'s original description to mirror IC-GEOG in the following ways: i) we fix all residents' ROL to be length 10; ii) we allow programs to rank all residents who rank them (again, drawing from a Plackett-Luce model with the scoring vector as described above); and iii) we impose the same geographical restriction on couples' joint ROLs as Kojima et al. do for IC-GEOG. In addition to being more directly comparable to IC-GEOG, imposing the geographical restriction on the Biró et al. model allows us to better capture couples' real-world preferences in larger markets. (The SFAS market has roughly 800 program positions; we test up to 30,000 program positions.)

In both models, we set the number of positions as 87% of the number of residents to match the 2015 NRMP data [17]. Each data point represents the average of 50 instances.

5.2 Results and Discussion

We begin by investigating the ability of residents to benefit by truncating their preferences in SMP-C as the size of the market gets large. Roth and Peranson found that, when using RP99 on historical NRMP data, very few residents had an incentive to truncate [23]. They conjectured that this was due to two reasons: i) as market size grows large relative to ROL length, the size of the set of all stable matches gets smaller, and ii) as in SMP, residents have no incentive to manipulate when a market has a unique stable matching. In Theorem 2, we prove the latter point for truncations, but show that it is false for reorderings. Immorlica and Mahdian proved the former point for SMP [8], and we address it empirically for SMP-C in this section.

Figure 4 shows the average number of *additional* stable programs per resident (i.e., in excess of one) for 10% and 30% residents in couples in IC-GEOG and SFAS-GEOG models as the market size increases.[5] Note that every resident has

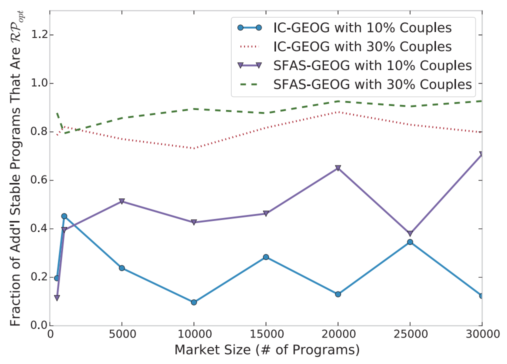

Figure 5: Average number of additional $\mathcal{RP}_{opt}$ matchings per resident divided by average number of additional stable matchings per resident as market size increases, only including satisfiable instances.

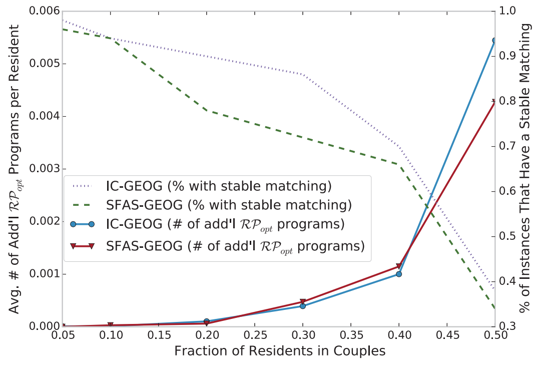

Figure 6: Average number of additional $\mathcal{RP}_{opt}$ programs per resident, and overall percentage of satisfiable instances, as fraction of residents in couples increases. 20,000 programs.

at least one stable program because we restrict our attention to SMP-C instances that are satisfiable. The models with a larger percentage of couples have more stable programs per resident, but the differences between IC-GEOG and SFAS-GEOG are small. We see that the average number of stable programs per resident drops rapidly as the market size increases, irrespective of model, which shows that an analog of Immorlica and Mahdian's result may apply in this setting.

In addition, we find that the residents' ability to truncate is even more strongly curtailed under an $\mathcal{RP}_{opt}$ mechanism. Figure 5 shows the average number of additional $\mathcal{RP}_{opt}$ programs per resident relative to the average number of additional stable programs per resident. We see that generally residents have much fewer additional $\mathcal{RP}_{opt}$ programs than additional stable programs. Since Figure 4 shows that the average number of additional stable programs is well below 1, this means that many residents have a unique stable program and even more will have a unique stable $\mathcal{RP}_{opt}$ program. Hence, by Theorem 1 there will be more residents with no incentive to manipulate an $\mathcal{RP}_{opt}$ mechanism than there are residents with no incentive to manipulate a stable mechanism.

Figure 6 provides a more detailed picture of what happens

[5]Standard error in this graph is 0.003-0.005% for the 30% couples models with market size under 1000 programs and less than 0.001% otherwise.

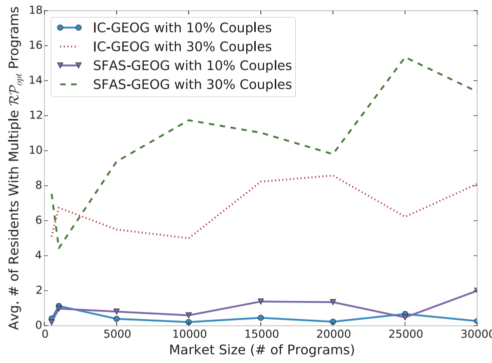

Figure 7: Average number of residents with multiple $\mathcal{RP}_{opt}$ programs as market size increases, only including satisfiable instances.

to the number of additional $\mathcal{RP}_{opt}$ programs per resident as the fraction of residents in couples increases for a fixed market size of 20,000. We see that the two models behave similarly, showing a superlinear growth. On the other axis, we see that although the average number of $\mathcal{RP}_{opt}$ programs per resident increases rapidly as the fraction of couples increases, the percentage of instances with a stable matching decreases. With 5% couples, almost all instances had at least one stable matching for IC-GEOG and SFAS-GEOG, while with 50% couples, fewer than 40% do. Thus, mechanism designers in environments with a high fraction of couples will have to contend with both the lack of existence of stable matchings and high incentives to truncate among residents.

We can conclude from Figure 5 that, under an $\mathcal{RP}_{opt}$ mechanism, residents will have substantially fewer programs they can potentially be matched to as the result of a truncation. Combining this result with Figure 6, this effect is particularly strong for low percentages of couples in the market. We find that both RP99 and KPR behave like $\mathcal{RP}_{opt}$ mechanisms in instances with a fraction of couples similar to that found in the NRMP in the 1990s (4% [23]). In particular, for instances with 20,000 programs and 5% couples and multiple stable matchings, KPR found an $\mathcal{RP}_{opt}$ matching 100% of the time and RP99 found an $\mathcal{RP}_{opt}$ matching more than 93% of the time.[6] Thus, i) RP99 is nearly an $\mathcal{RP}_{opt}$ mechanism, ii) $\mathcal{RP}_{opt}$ mechanisms can rarely be manipulated by resident truncations on markets with a low percentage of couples, and iii) when they can, only an extremely small number of residents can benefit. We thus hypothesize that the number of residents who could benefit by truncating observed by Roth and Peranson was made lower by their exploitation of $\mathcal{RP}_{opt}$ matchings rather than a small number of stable matchings as they conjectured.

However, there is a peculiar caveat to these results: the number of residents per instance with multiple stable or $\mathcal{RP}_{opt}$ programs does not seem to be affected by market size. Figure 7 shows the average number of residents with multiple $\mathcal{RP}_{opt}$ programs for 10% and 30% residents in couples in IC-GEOG and SFAS-GEOG models as the market size increases.[7] We see that the number of residents with multiple $\mathcal{RP}_{opt}$ programs does not decrease as market size increases. We observed similar behaviour for the number of residents with multiple stable matchings. Comparing IC-GEOG and SFAS-GEOG, we see that SFAS-GEOG has more residents with multiple $\mathcal{RP}_{opt}$ programs, which is consistent with the higher fraction of $\mathcal{RP}_{opt}$ programs in SFAS-GEOG that we previously observed. Unsurprisingly, the number of residents with multiple stable or $\mathcal{RP}_{opt}$ programs is higher when the fraction of couples is increased, since an instance with a higher fraction of couples tends to have a larger number of stable and $\mathcal{RP}_{opt}$ matchings.

Since it only requires a single resident with multiple $\mathcal{RP}_{opt}$ programs to prevent the existence of a $\mathcal{R}_{opt}$ matching, we likewise find that the frequency of instances that have $\mathcal{R}_{opt}$ matchings is not affected by market size. $\mathcal{R}_{opt}$ matchings exist 94.2% and 70.1% of the time for IC-GEOG with 10% and 30% couples and 91.1% and 62.9% of time for SFAS-GEOG with 10% and 30% couples.

These results suggest that Roth and Peranson's observation that few residents could benefit by truncating, was not affected by the size of the market, further suggesting their observation was only dependent on the percentage of residents in couples. While there may be some residents that can benefit from truncating irrespective of market size, we expect the fact that the number of them is so small to be a strong disincentive to truncating in practice. It requires considerable effort (e.g., learning the preferences of others) to be able to truncate effectively, and these efforts will be very unlikely to yield any benefit in sufficiently large instances. In addition, truncating carries the risk that a resident will become unmatched even under an $\mathcal{RP}_{opt}$ mechanism.

6. CONCLUSIONS

In this paper we examined strategy-proofness for SMP-C, showing that in general, no resident strategy-proof stable mechanism exists. We do show that, for certain problem instances, a stable matching mechanism that always returns an $\mathcal{RP}_{opt}$ matching is strategy-proof w.r.t. residents truncating their preferences, and we extend a previously developed SAT encoding to provide an $\mathcal{RP}_{opt}$ mechanism. We empirically show, on two very different preference distributions based on real-world markets, that when a low percentage of couples exist in the market, the $\mathcal{RP}_{opt}$ mechanism is frequently resistant to truncations, suggesting a new hypothesis for why there is little incentive for residents to truncate their preferences in the NRMP.

Furthermore, our empirical results suggest new avenues for theoretical research: we hypothesize that an analogous SMP-C result exists for the Immorlica and Mahdian large markets result for SMP [8]; we additionally hypothesize that if the percentage of couples in the market grows slower than the size of the market, an $\mathcal{R}_{opt}$ matching exists with high probability (an analogous result to the Ashlagi et al. large markets result [3]). We also wish to further use our extension to SAT-E to allow us to empirically explore different properties of SMP-C, providing further insight into possible theoretical results.

Acknowledgements

We acknowledge the support of NSERC. Drummond and Perrault were supported by OGS. We thank the reviewers for helpful suggestions.

[6]There was a problem instance where an $\mathcal{R}_{opt}$ matching existed, and RP99 did not find it.

[7]Standard error in this graph is 0.3-0.5 for the 10% couples models and 2.5-3.5 for the 30% couples models.

REFERENCES

[1] A. Abdulkadiroglu, P. Pathak, A. E. Roth, and T. Sönmez. The Boston public school match. *American Economic Review*, 95(2):368–371, 2005.

[2] A. Abdulkadiroglu, P. Pathak, A. E. Roth, and T. Sonmez. Changing the Boston school choice mechanism. Technical report, National Bureau of Economic Research, 2006.

[3] I. Ashlagi, M. Braverman, and A. Hassidim. Stability in large matching markets with complementarities. *Operations Research*, 62(4):713–732, 2014.

[4] P. Biró, D. F. Manlove, and I. McBride. The hospitals/residents problem with couples: Complexity and integer programming models. In *Experimental Algorithms*, pages 10–21. Springer, 2014.

[5] J. Drummond, A. Perrault, and F. Bacchus. SAT is an effective and complete method for solving stable matching problems with couples. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015.

[6] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.

[7] D. Gusfield and R. W. Irvine. *The stable marriage problem: structure and algorithms*. Foundations of Computing Series. The MIT Press, 1989.

[8] N. Immorlica and M. Mahdian. Marriage, honesty, and stability. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 53–62, 2005.

[9] R. Irving. Matching practices for entry-labor markets - scotland. 2011. Accessed: 2015-11-14.

[10] S. Kannan, J. Morgenstern, A. Roth, and Z. S. Wu. Approximately stable, school optimal, and student-truthful many-to-one matchings (via differential privacy). In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1890–1903, 2015.

[11] D. E. Knuth. *Marriages stables*. Les Presses de I'Universite de Montreal, 1976.

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

[13] F. Kojima, P. Pathak, and A. E. Roth. Online appendix matching with couples: Stability and incentives in large markets. *Quarterly Journal of Economics*, 128(4), 2013.

[14] R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

[15] I. McBride and D. F. Manlove. The hospitals / residents problem with couples: Complexity and integer programming models. *CoRR*, abs/1308.4534, 2013.

[16] National Resident Matching Program. National resident matching program, results and data: 2013 main residency match$^{®}$, 2013.

[17] National Resident Matching Program. National resident matching program, results and data: 2015 main residency match$^{®}$. 2015.

[18] M. Niederle, A. E. Roth, and T. Sonmez. Matching and market design. In S. N. Durlauf and L. E. Blume, editors, *The New Palgrave Dictionary of Economics (2nd Ed.)*, volume 5, pages 436–445. Palgrave Macmillan, Cambridge, 2008.

[19] R. Plackett. The analysis of permutations. *Applied Statistics*, 24:193–202, 1975.

[20] E. Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11(2):285–304, 1990.

[21] A. E. Roth. The economics of matching: Stability and incentives. *Mathematics of Operations Research*, 7(4):617–628, 1982.

[22] 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.

[23] A. E. Roth and E. Peranson. The redesign of the matching market for American physicians: Some engineering aspects of economic design. *The American Economic Review*, 89(1):748–780, September 1999.

[24] A. E. Roth and U. G. Rothblum. Truncation strategies in matching markets—in search of advice for participants. *Econometrica*, 67(1):21–43, 1999.

[25] A. E. Roth and M. Sotomayor. Chapter 16. two-sided matching. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory Volume 1*, pages 485–541. Elsevier, 1992.

[26] A. E. Roth and J. H. Vande Vate. Random paths to stability in two-sided matching. *Econometrica*, 58(6):1475–1480, 1990.