The Complexity of Stable Matchings under Substitutable Preferences

Yuan Deng and **Debmalya Panigrahi**
Department of Computer Science
Duke University
Durham, NC 27708, USA
{ericdy,debmalya}@cs.duke.edu

Bo Waggoner
Department of Computer Science
University of Pennsylvania
Philadelphia, PA 19104, USA
bwag@seas.upenn.edu

Abstract

In various matching market settings, such as hospital-doctor matching markets (Hatfield and Milgrom 2005), the existence of stable outcomes depends on substitutability of preferences. But can these stable matchings be computed efficiently, as in the one-to-one matching case? The algorithm of (Hatfield and Milgrom 2005) requires efficient implementation of a choice function over substitutable preferences. We show that even given efficient access to a value oracle or preference relation satisfying substitutability, exponentially many queries may be required in the worst case to implement a choice function. Indeed, this extends to examples where a stable matching requires exponential time to compute.

We characterize the computational complexity of stable matchings by showing that efficient computation of a choice function is equivalent to efficient *verification*—determining whether or not, for a given set, the most preferred subset is the entire set itself. Clearly, verification is necessary for computation, but we show that it is also sufficient: specifically, given a verifier, we design a polynomial-time algorithm for computing a choice function, implying an efficient algorithm for stable matching. We then show that a verifier can be implemented efficiently for various classes of functions, such as submodular functions, implying efficient stable matching algorithms for a broad range of settings. We also investigate the effect of ties in the preference order, which causes complications both in defining substitutes and in computation. In this case, we tightly connect the computational complexity of the choice function to a measure on the number of ties.

Introduction

Ever since the seminal work of (Gale and Shapley 1962), stable matching has been intensively studied in both economics and algorithms. Applications have ranged from school matching (Abdulkadiroglu and Sönmez 2003; Abdulkadiroğlu, Pathak, and Roth 2005; Gale and Shapley 1962), hospital-resident matching (Irving and Manlove 2009; Irving, Manlove, and Scott 2000; Roth 1996), to kidney exchange programs (Abraham, Blum, and Sandholm 2007; Roth, Sonmez, and Unver 2003; Roth, Sönmez, and Ünver 2005).

In the standard one-to-one stable matching model (Gale and Shapley 1962), there is a set of pianists and a set of violinists looking to form duets. Each agent has a preference relation over a subset of the other instrument's players and prefers to stay unmatched rather than be matched with an agent outside this subset. An algorithm takes every agent's preference list as input and outputs a matching. A matching generated by the algorithm is stable if no pair of agents prefer to match with each other over their designated partners and no matched agent prefers being unmatched. Gale and Shapley (1962) proposed the deferred-acceptance or "Gale-Shapley" algorithm to compute a stable matching in quadratic time. The algorithm (violinist-proposing version) proceeds in multiple rounds. In each round, each violinist makes a proposal to her favorite pianist who has not rejected her yet; and each pianist keeps her favorite proposal and rejects all others. The algorithm iterates until no further proposal can be made.

A more complex setting is many-to-one matching such as in hospital-doctor matching markets (Roth and Peranson 1997; Roth and Xing 1997; Niederle and Roth 2005). Here, there is a set of hospitals and a set of doctors. Each hospital has a preference list over the subsets of doctors while each doctor has a preference list over the set of hospitals. A matching is stable if for each hospital, there exists no subset of doctors that the hospital prefers over its currently allocated subset such that all those doctors prefer to move to that hospital over their assigned one. Unlike in one-to-one settings, the existence of a stable matching is not generally guaranteed in a many-to-one scenario. Substitutable preferences, introduced by Roth (1984), have been shown to be a necessary and sufficient condition to guarantee the existence of a stable outcome in many-to-one settings (Roth 1984; Hatfield and Kojima 2008; Hatfield and Kominers 2011). Substitutes, intuitively, imply that each doctor can become only less desirable when more choices become available. It is a natural assumption when, for instance, doctors have overlapping skill sets. Substitutability has also been studied extensively in consumer theory (Hanemann 1984) and auction theory (Milgrom and Strulovici 2009).

Choice functions and many-to-one matchings. Under substitutable preferences, Hatfield and Milgrom (2005) designed a generalized Gale-Shapley algorithm to compute a stable many-to-one matching and proved the induced mechanism is incentive-compatible for doctors in the doctor-proposing version of the algorithm.

This landmark work initiated a chain of related research in Internet advertising (Edelman, Ostrovsky, and Schwarz 2007), supply chain networks (Ostrovsky 2008), and school admissions (Abdulkadiroglu et al. 2006). The algorithm (doctor-proposing version) proceeds in multiple rounds. In each round, each doctor makes an offer to her favorite hospital that has not rejected her yet; and each hospital keeps its favorite subset of doctors from its previous set and the new proposals, rejecting all others. The algorithm iterates until no further offer can be made. Hatfield and Milgrom (2005) showed that under substitutable preferences of hospitals, this algorithm converges to a stable matching in quadratically many iterations in the number of doctors and hospitals. However, they assume that an oracle to the choice function exists: in every iteration, each hospital can query the oracle to determine its favorite subset of doctors among those currently proposing to it.

In this paper, we analyze the computational complexity of implementing the choice function for a single hospital under substitutable preferences in order to understand the complexity of the generalized Gale-Shapley algorithm. We make the minimum assumption that a preference relation can be accessed such that the relative order between two sets of doctors can be compared efficiently. The complexity of the implementation is measured by the number of queries to the preference relation. Unfortunately, in the worst case, we show that evaluating a choice function requires exponential queries to a substitutable preference relation, even represented as a valuation function over the subsets of doctors.

A natural question arises: when can a choice function for substitutable preferences be computed efficiently? We show that computation is equivalent to the seemingly-simpler task of *verification*: deciding whether or not a given set of doctors is preferred to all of its subsets. We choose this terminology because such an algorithm can verify, given a set of doctors D, whether or not D is the most-preferred set among all subsets of D. Further, note that efficient verification is clearly necessary for efficient computation; we show that it is also *sufficient*. Given access to a verifier, we design an efficient algorithm for computing the choice set using $O(n^3)$ queries to the verifier and $O(n^2)$ queries to preference order when there are n doctors (hence, 2^n possible subsets). We further demonstrate how to implement the verifier efficiently for various classes of valuation functions, including submodular functions and monotone functions with efficiently checkable downward-closed constraints, e.g., matroid constraints, knapsack constraints, and various graph properties. When combined with (Hatfield and Milgrom 2005), this gives an efficient algorithm for many-to-one matching in the corresponding settings.

The key idea in our algorithm is to repeatedly search for a doctor who cannot be in the choice set, remove that doctor from consideration, and repeat. To do so, we will use substitutability: a doctor who is not in the choice set of a subset of doctors cannot be in the choice set of the entire set of doctors either. But, how do we identify such a subset and a doctor who is not in its choice set? While this cannot be done directly for all subsets of doctors, we introduce a new notion of an *almost-optimal set*: a set A whose choice set has exactly one fewer doctors, i.e. has size $|A| - 1$. Consequently, for an almost optimal set, we can identify a doctor who is not in its choice set efficiently. Finally, using a polynomial number of calls to an efficient verifier, we design a search algorithm to find an almost-optimal set.

The main application of our results is in matching, but they also contribute to a basic algorithmic question: When can we implement a choice function (aka. a "demand oracle") given access to a preference relation or "value oracle"? We address this question under the assumption of substitutability as formalized in the matching literature.

Preference relations with ties. Our previous results apply for preference relations without ties. One might expect the results to extend naturally to the case with ties. This is the case in one-to-one matching, where Irving (1994) extended the Gale-Shapley algorithm to preference relations with ties and provided a quadratic time algorithm to compute a weakly stable matching in which no pair of agents strictly prefer to match with each other than their designated partners.

However, some subtle challenges arise in many-to-one matching settings. In particular, if the result of the choice function is not well defined (*i.e.* if two different subsets of doctors are equally preferred), it is not *a priori* obvious how to repair the definition of substitutability to match. In particular, there are two intuitive conditions for substitutability that one might impose (Sotomayor 1999). These conditions are equivalent without ties—hence the definition of substitutability is unambiguous in this case—but distinct if there are ties. We show that these two conditions have very different implications on complexity of choice functions.

For the first condition, the equivalence of verification and computation continues to hold. As an interesting special case, we consider the widely studied class of gross substitute preferences (Kelso Jr and Crawford 1982). Leme (2014) showed that for gross substitute valuations, a demand oracle can be implemented via an "ascending greedy" algorithm. In contrast, we show that gross substitutability satisfies the first condition (indeed, both conditions) for substitutability with ties, and has an efficient verifier. This provides a new method for efficient implementation of the demand oracle: our "descending" algorithm, that removes elements and maintains a superset of the choice set, takes an interesting opposite approach to the existing ascending greedy algorithm.

In contrast to the first condition, we show that if only the second condition is satisfied, then efficient verification does *not* necessarily imply efficient computation of the choice function. We introduce the notion of a critical set as a measure of the ambiguity introduced by ties in this case, and obtain a separation of complexity via the number of critical sets in a preference relation. Intuitively, a critical set of doctors is one that a verifier denies being preferred to all of its subsets, yet due to ties, our search algorithm cannot efficiently pick a doctor to remove from candidates. When the number of critical sets is large, no algorithm implementing a choice function can avoid exponentially many queries; but when it is small, our algorithm runs efficiently. In this sense, the number of critical sets exactly captures the complexity of the choice function.

We note that our work complements the line of work focusing on the computational complexity of testing substitutability of a preference relation. In that line of research, Hatfield, Immorlica, and Kominers (2012) provided an efficient algorithm to test the substitutability of preference relations without ties, which was later extended to preference relations with ties by (Aziz, Brill, and Harrenstein 2013).

Preliminaries

Let D be a finite set of doctors. A hospital's preferences are specified by a preference relation $\{\succ, \prec\}$ defining a transitive and complete relation on 2^D, where 2^D is the set of all subsets of X. For now, we assume the preference relation has no ties: for every $A, B \in 2^D$, either $A \prec B$ or $A \succ B$. (Ties are discussed separately later.) If $\mathcal{M} \subseteq 2^D$, let $\max \mathcal{M}$ be the most preferred subset in $\mathcal{M}$:

$$\max \mathcal{M} = Y \in \mathcal{M} \ : \ \forall Z \in \mathcal{M} \ s.t. \ Z \neq Y, Y \succ Z.$$

The preference relation induces a *choice function* $\mathbf{C}$ such that, for each $X \subseteq D$, $\mathbf{C}(X) = \max 2^X$. It will also be useful to define the *neighbor set* $\mathcal{N}(A)$ to be

$$\mathcal{N}(A) = \{B \subset A \ : \ |B| = |A| - 1\}.$$

We are interested in the complexity of computing the choice function:

Definition 1 (Choice function problem without ties)**.** *In the choice function problem, an algorithm is given D and oracle access to a preference relation without ties, i.e., an oracle call* `compare`(A, B) *with $A, B \subseteq D$ and $A \neq B$ returns either $A \prec B$ or $A \succ B$. The algorithm must output $\mathbf{C}(D)$.*

One can also consider a special case of this model where preferences are given by a "value oracle" $f : 2^D \to \mathbb{R}$ with $A \prec B \iff f(A) > f(B)$. Both our positive and negative results will apply to either model. We will focus on complexity in terms of number of queries to the oracle.

Next, we define the notion of substitutable preferences.

Definition 2 (Substitutable preferences without ties)**.** *A preference relation without ties is* substitutable *if and only if for all non-empty sets $A, B \subseteq D$ with $B \subseteq A$, the choice function $\mathbf{C}(\cdot)$ satisfies $\mathbf{C}(A) \cap B \subseteq \mathbf{C}(B)$.*

Intuitively, property (1) of this definition is that if the hospital accepts a doctor, it continues to do so when fewer doctors are available. An equivalent condition is: if the rejection function is defined as $\mathbf{R}(X) := X \setminus \mathbf{C}(X)$, then a preference relation is substitutable if and only if, for all $A \subseteq B$, $\mathbf{R}(B) \subseteq \mathbf{R}(A)$. This implies the equivalent property (2): if the hospital rejects a doctor, it continues to do so when more doctors are available (Sotomayor 1999). (We will see that these conditions are not equivalent when the preference relation has ties.)

Verification and Computation

In this section, we will first show that the choice function C cannot be computed efficiently in general, even assuming substitutable preferences. Then, we will give an algorithm to efficiently compute C given access to a *verifier*. We later discuss cases where a verifier can be efficiently implemented and implications for many-to-one stable matching.

First, we give a class of substitutable preferences whose choice functions require exponentially many queries to compute.

Example 1. *Given a fixed $S^* \subseteq D$, define $f_{S^*} : 2^D \to \mathbb{R}$ by*

$$f_{S^*}(S) = \begin{cases} |D| + 1 & S = S^* \\ |S| + Noise(S) & otherwise \end{cases}$$

where $Noise(S)$ is drawn uniformly from $(0, 0.5)$.

Lemma 1. *The preference relation induced by valuation function f in Example 1 is substitutable.*

Proof. Notice that, for all non-empty sets $A, B \subseteq D$ with $B \subseteq A$, if $S^* \not\subseteq B$, then $\mathbf{R}(B) = \emptyset \subseteq \mathbf{R}(A)$; otherwise when $S^* \subseteq B$, then $\mathbf{R}(B) = B \setminus S^* \subseteq A \setminus S^* = \mathbf{R}(A)$. Therefore, the preference relation induced by the valuation relation f is substitutable. □

We can use this family of functions to construct a choice function that cannot be efficiently computed by any algorithm.

Theorem 1. *There exists a class of substitutable preferences without ties such that any algorithm requires $\Omega(2^n)$ queries to compute its induced choice function.*

The proof of this theorem appears in the full version of this paper. To see the intuition, suppose that an algorithm is given f_{S^*} by picking S^* uniformly at random. No query to f gives any information about the choice set S^*, except for $f(S^*)$ itself. So any algorithm must query at least half the possible sets ($2^{|D|-1}$) on average before discovering S^*. Therefore, in the worst case, the algorithm requires exponentially many queries.

Theorem 1 also implies that any stable matching algorithm must make exponentially many queries in the worst case. Consider a single hospital and set of doctors all of whom would like to be matched to that hospital; the stable matching problem is identical to computing the hospital's choice function.

In order to tackle the problem, we consider a verifier of the choice function.

Definition 3 (Verifier)**.** *A verifier $\mathbb{V}(\cdot)$ of a choice function is a boolean oracle which when given a set $X \subseteq D$, outputs* TRUE *if $X = \mathbf{C}(X)$; otherwise, it outputs* FALSE*.*

In the remainder of this paper, we denote the implementation of the verifier as a *verification* problem and the implementation of the choice function as a *computation* problem, for convenience. Note that provided an oracle for the computation problem, one can implement the corresponding verifier efficiently by checking whether $X = \mathbf{C}(X)$. In other words, efficient computation implies efficient verification. The main contribution in this section is to show the converse, that efficient verification also implies efficient computation for choice functions induced by substitutable preference relations.

Designing the algorithm. First, let us see what algorithmic power we *can* get from the substitutes condition. The key

property of substitutability is that if the hospital rejects a doctor given a set of doctors to choose from, then it also rejects that doctor when more doctors are available. Formally, the definition directly implies the following.

Fact 1. *If a preference relation with choice function* $\mathbf{C}$ *is substitutable, then for any* $A \subseteq D$ *and doctor* $v \in A$*, if* $v \notin \mathbf{C}(A)$*, then* $v \notin \mathbf{C}(D)$.

By Fact 1, if we are able to find a doctor $v \in A$ such that $A \subseteq D$ and $v \notin \mathbf{C}(A)$, then it is safe to "reject" v. In other words, we have that $\mathbf{C}(D) = \mathbf{C}(D \setminus \{v\})$. To find $\mathbf{C}(D \setminus \{v\})$, we again find a doctor to reject until there is none. Noticing that the stopping condition can be checked by the verifier, this gives Algorithm 1, assuming a rejection algorithm $\mathbf{I}(\cdot)$ finds it a doctor we can reject.

Algorithm 1: Implementation of the choice function

Input: A set of doctors D, verifier V, rejection alg. $\mathbf{I}$;
Output: $\mathbf{C}(D)$;

Set $X := D$;
while $\mathrm{V}(X) = \textsc{False}$ **do**
  Set $v \leftarrow \mathbf{I}(X)$; `// find a v to reject`
  Set $X \leftarrow X \setminus \{v\}$;
return X;

But how can we implement the rejection algorithm $\mathbf{I}(\cdot)$ in polynomial time? If we can, then we can implement the entire choice function in polynomial time, as Algorithm 1 only makes at most $|D|$ iterations (there are only $|D|$ elements in D to reject).

To illustrate our search algorithm for a doctor we can reject, we introduce the notion of an *almost-optimal set*. Recall that the "neighbor" set $\mathcal{N}(A)$ consists of subsets of A of size $|A| - 1$.

Definition 4 (Almost-optimal set)**.** *A subset* A *of doctors is* almost-optimal *if* $\mathrm{V}(A) = \textsc{False}$ *and for all subsets* $B \in \mathcal{N}(A)$*,* $\mathrm{V}(B) = \textsc{True}$.

Almost-optimal sets are useful because their choice sets can be directly computed. This allows us to find a doctor to reject in Algorithm 2 that implements $\mathbf{I}(\cdot)$, assuming a subroutine $M(\cdot)$ finds it an almost optimal set. Recall that $\max \mathcal{N}(A)$ returns the most preferred subset in $\mathcal{N}(A)$.

Algorithm 2: $\mathbf{I}(\cdot)$: Find an element $v \notin \mathbf{C}(D)$

Input: A set of doctors X such that $\mathrm{V}(X) = \textsc{False}$, verifier V, almost-optimal alg. $\mathbf{M}(\cdot)$;
Output: A $v \in X$ with $v \notin \mathbf{C}(X)$;

Set $A \leftarrow \mathbf{M}(X)$; `// almost-optimal subset`
Set $B^* = \max \mathcal{N}(A)$;
return $A \setminus B^*$

We assert the correctness of our implementation of $\mathbf{I}(\cdot)$ in the next lemma.

Lemma 2. *For an almost-optimal set* $A \subseteq D$*, the unique element* v *in* A *but not in* $\max \mathcal{N}(A)$ *is not in* $\mathbf{C}(D)$.

Proof. For all $B \in \mathcal{N}(A)$, $\mathrm{V}(B) = \textsc{True}$, so $B \succ B'$ for all $B' \subset B$. We also have $\mathrm{V}(A) = \textsc{False} \implies A \neq \mathbf{C}(A)$. This implies that $\mathbf{C}(A) \in \mathcal{N}(A)$. Therefore, by definition of $\max \mathcal{N}(A)$, we have that $\mathbf{C}(A) = \max \mathcal{N}(A)$. Letting $v = A \setminus \max \mathcal{N}(A)$, this implies $v \notin \mathbf{C}(A)$. Finally, by the property of substitutability of Fact 1, we conclude that $v \notin \mathbf{C}(D)$. □

The only remaining problem is to implement $\mathbf{M}(\cdot)$ efficiently, i.e., find an almost-optimal subset of X. This is done in Algorithm 3.

Algorithm 3: $\mathbf{M}(\cdot)$: Find an almost optimal set A of X

Input: A set of doctors X such that $\mathrm{V}(X) = \textsc{False}$, verifier V;
Output: An almost optimal set $A \subseteq X$;

Set $A := X$;
while *there exists* $B \in \mathcal{N}(A)$ *with* $\mathrm{V}(B) = \textsc{False}$ **do**
  Let $B \in \mathcal{N}(A)$ such that $\mathrm{V}(B) = \textsc{False}$;
  Set $A \leftarrow B$;
return A;

We assert the correctness of our implementation of $\mathbf{M}(\cdot)$ in the next lemma.

Lemma 3. *If* $\mathrm{V}(X) = \textsc{False}$*, then some almost-optimal subset* $A \subseteq X$ *exists; and Algorithm 3 finds one.*

Proof. First, notice that $\mathrm{V}(\emptyset) = \textsc{True}$, so if $\mathrm{V}(A) = \textsc{False}$, then A is nonempty. By induction on $|A|$: If $|A| = 1$, then A is almost optimal, as its only neighbor is $\emptyset$. If $|A| \geq 2$: Either A is almost optimal, in which case we are done, or it has a neighbor B for which $\mathrm{V}(B) = \textsc{False}$. Since $|B| = |A| - 1$, we apply the induction hypothesis. □

We can now combine the three algorithms to claim that efficient verification implies efficient computation of choice functions with substitutable preferences without ties.

Theorem 2. *The implementation of a choice function for substitutable preferences without ties, Algorithm 1, requires at most* $|D|^3$ *queries to the verifier and* $|D|^2$ *queries to the preference relations.*

Proof. Algorithm 1 takes at most $|D|$ iterations, each requiring one call to the verifier and one call to subroutine $\mathbf{I}(X)$ with an X of size at most D. Algorithm 2 makes a single call to Algorithm 3. Then it queries preferences to find the most-preferred among a set of at most $|D|$, which requires at most $|D|$ comparisons (by maintaining a candidate maximum and comparing it to each in sequence). Algorithm 3 loops at most $|D|$ times (as $|A|$ decreases each time) calling the verifier at most $|D|$ times per loop. □

Note that, although Theorem 2 is technically stated in terms of the number of queries, all computations performed in the algorithm are efficient, hence they run in polynomial time when given an efficient verifier. Therefore, in this paper, we do not distinguish between saying that our algorithm uses a polynomial number of queries and saying that it runs in polynomial time.

Efficiently verifiable functions

In the worst case, verification requires an algorithm to exhaustively enumerate all possible subsets. In fact, Example 1 provides a concrete example, since without the knowledge of the optimal subset S^*, given an input X, any algorithm needs to enumerate every subset of X to determine $\mathrm{V}(X)$. However, for certain classes of valuation functions, verification can be done with polynomially many queries to the preference relation. In these cases, of which we give some popular examples below, the above theorem implies efficient computation.

Submodular functions: A submodular function is a set-valued function f such that for any non-empty subset A, B with $B \subseteq A$ and an element $v \notin A$, $f(A \cup \{v\}) - f(A) \leq f(B \cup \{v\}) - f(B)$.

Lemma 4. *For a submodular function f, $\mathrm{V}(A) = \textsc{True}$ if and only if $\forall B \in \mathcal{N}(A), f(A) > f(B)$.*

Proof. If $\mathrm{V}(A) = \textsc{True}$, then $f(A)$ must be larger than any other subset of A, and thus, we have $\forall B \in \mathcal{N}(A), f(A) > f(B)$.

For the converse, suppose $f(A) > f(B)$ for all $B \in \mathcal{N}(A)$; we show $f(A) > f(B')$ for any $B' \subseteq A$. Consider $v \in A, v \notin B'$. The "marginal improvement" of v is positive: $0 < f(A) - f(A \setminus \{v\})$. By submodularity, $f(A) - f(A \setminus \{v\}) \leq f(B' \cup \{v\}) - f(B')$. This implies that $B' \neq \mathbf{C}(A)$, since B' can be improved by adding v. □

Thus, given a set A, verification only requires checking the submodular function's value on A and all its neighbors.

Corollary 1. *Implementing a verifier $\mathrm{V}(A)$ for a submodular function requires only $|A| + 1$ function evaluations.*

Monotone functions with efficiently checkable downward-closed constraints: A function f is said to be monotone with efficiently checkable downward-closed constraints if there exists a set H of subsets of D such that (1) if $X \in H$, then for all $X' \subseteq X$, $X' \in H$; (2) for all $B \subseteq A$ with $A \in H$, $f(B) < f(A)$; (3) for all $X \notin H$, $f(X) < \max_{X' \subseteq X, X' \in H} f(X')$. According to our construction, note that $\mathrm{V}(A) = \textsc{True}$ if and only if $A \in H$. Hence, if we can efficiently decide whether $A \in H$, we can also implement the verifier efficiently. Several constraints are downward-closed and efficiently checkable, such as matroid constraints, knapsack constraints, and various graph properties.

Implications for Many-to-One Matching

The above results imply complexity results for the many-to-one stable matching problem.

Recall that in this problem, we are given a set of doctors D and a set of hospitals H. Each doctor has a preference ordering over the hospitals, and each hospital has a preference order over subsets of doctors. The goal is to find an assignment of doctors to hospitals, so that each doctor is matched to at most one hospital, that is *stable*. The stability condition is that for each hospital h, there exists no subset of doctors that the hospital prefers to its assigned set such that all those doctors prefer h over their assigned hospital.

Hatfield and Milgrom (2005) proposed a generalized Gale-Shapley algorithm for this problem and showed that, when hospital preferences satisfy substitutes, it runs in a quadratic number of iterations where each iteration requires polynomially many queries to the hospital choice functions.

Notice that a special case of many-to-one stable matching is the case where there is a set D of doctors and just one hospital, which is liked by all doctors; in this case, the stable matching is the choice set of the hospital.

We summarize our findings for many-to-one stable matchings below:

Even when hospital preferences satisfy substitutes, no algorithm guarantees to find a stable matching with a subexponential number of queries to hospital preferences (Example 1, Theorem 1).

But given access to a verifier, there exists an algorithm for stable many-to-one matching running in polynomial time with a polynomial number of verifier calls (Theorem 2). In particular, polynomial-time algorithms exist in the following natural settings: submodular valuation functions and monotone valuation functions with efficiently checkable downward-closed constraints.

Preferences with Ties

In this section, we suppose a hospital's preference relation may contain ties with $A \sim B$ meaning A and B are equally preferred. In this case, the choice function is

$$\mathbf{C}(X) = \{Y \subseteq X \mid \forall Z \subseteq X, Y \succ Z \text{ or } Y \sim Z\}.$$

Because $\mathbf{C}(X)$ may be exponentially large, the computational problem is naturally phrased as finding any member of $\mathbf{C}(X)$ given oracle access to $\texttt{compare}(A, B)$ which returns $A \prec B$, $A \succ B$, or $A \sim B$. We also generalize $\max \mathcal{M}$ to mean the set of members of $\mathcal{M}$ that are preferred to or comparable to every other member of $\mathcal{M}$, i.e.

$$\max \mathcal{M} = \{B \in \mathcal{M} : \forall B' \in \mathcal{M}, B \succeq B'\}.$$

Recall that the definition of substitutes had, intuitively, two properties: (1) a hospital who rejects a doctor continues to do so when more doctors are available; (2) a hospital who accepts a doctor continues to do so when fewer doctors are available. Sotomayor (1999) observed that these two conditions are not equivalent if there are ties (see detailed examples and discussions in (Aziz, Brill, and Harrenstein 2013)), and preserved both conditions in defining substitutability over preference relations with ties.

Definition 5 (Substitutable preferences with ties)**.** *A preference relation with ties is substitutable if and only if the following two conditions hold*

1. *For all nonempty sets $A, B \subseteq D$ with $B \subseteq A$, we have for all $Y \in \mathbf{C}(B)$, there exists some $X \in \mathbf{C}(A)$, such that $X \cap B \subseteq Y$, and*
2. *For all nonempty sets $A, B \subseteq D$ with $B \subseteq A$, we have for all $X \in \mathbf{C}(A)$, there exists some $Y \in \mathbf{C}(B)$, such that $X \cap B \subseteq Y$.*

In the following discussions, we treat the above two conditions separately to illustrate their individual impact on the computability of the choice function. In fact, we observe a sharp contrast between the behavior of the choice function under these two conditions individually.

Condition 1

If the preference relation satisfies condition 1, we can generalize Fact 1 as follows.

Fact 2. *For any $A \subseteq D$ and doctor $v \in A$, if there exists $Y \in \mathbf{C}(A)$ such that $v \notin Y$, then there exists $X \in \mathbf{C}(D)$ such that $v \notin X$.*

Therefore, we can modify our previous approach slightly by arbitrarily setting B^* as an element in $\max \mathcal{N}(A)$ in Algorithm 2, and all our results for substitutable preference relations without ties applies to preference relations with ties, provided it satisfies condition 1 of substitutability.

Gross substitutes. As an application, we consider the widely studied class of gross substitute preferences. Gross substitutability applies in an economy comprising a set $[n]$ of indivisible goods. Each agent has a valuation $v : 2^{[n]} \to \mathbb{R}$. Given a price vector $p \in \mathbb{R}^n$ and a set $S \subseteq [n]$, the utility function for each agent is $u(S, p) = v(S) - \sum_{i \in S} p_i$. A demand oracle $\mathbf{D}$ takes a valuation v and a price vector p as input, and outputs a set of subsets of $[n]$ that maximize the utility of the agent for this price vector of the algorithm.

This setting can be viewed as a generalization of our original setting, which is the case where all prices must be set to either zero or (positive) infinity. In that case, the demand oracle will not take any item having infinite price, so it corresponds to the choice function applied to the subset having price zero.

Definition 6 (Gross substitutability)**.** *A valuation function satisfies the gross substitute property if for any price vectors $p \in \mathbb{R}^n$ and set $S \in \mathbf{D}(v, p)$, if p' is a price vector with $p \leq p'$, then there is a set $S' \in \mathbf{D}(v, p')$ such that $S \cap \{j \mid p_j = p'_j\} \subseteq S'$.*

The standard implementation of a demand oracle in this setting is an "ascending greedy" algorithm (Leme 2014) that starts with an empty set and adds doctors, one at a time, to the choice set. In contrast, we give a new implementation of the demand oracle using a "descending" algorithm based on our search procedure that starts with all doctors and discards doctors who are not in the choice set one at a time. This implementation follows from showing that gross substitutability implies condition 1 (indeed, condition 2 as well) of substitutability with ties. This allows us to invoke our search algorithm, since gross substitutability implies submodularity (Gul and Stacchetti 1999), which in turn implies efficient verification. The next theorem, whose proof appears in the full version, formalizes these claims.

Theorem 3. *Given a fixed price vector p, the preference relation (possibly with ties) induced by the utility function $u(\cdot, p)$ satisfies condition 1 (and condition 2) of substitutability with ties.*

Condition 2

We have shown that condition 1, on its own, is sufficient for our algorithm to apply. But, what if only condition 2 is satisfied? Unfortunately, in this case, Algorithm 1 does not work since for an almost optimal set A, $\max \mathcal{N}(A)$ may not be unique and Algorithm 2 is unable to find a doctor to discard from further consideration. In fact, this failure of the algorithm is fundamental: it turns out that verification is *strictly* easier than computation in this case. Recall that we showed previously that we can implement a verifier efficiently for submodular valuation functions. Nevertheless, we can show the following negative result:

Theorem 4. *There exists a class of preferences with ties induced by a submodular function and satisfying substitutability (condition 2) such that any implementation of its induced choice function requires an exponential number of queries to the submodular function.*

The instance that establishes the above theorem, given in the full version along with the proof of the theorem, turns out to have an exponential number of what we call *critical* sets: almost optimal sets A for which $|\max \mathcal{N}(A)| > 1$. Intuitively, these are cases that flummox our descending search algorithm by presenting ties at almost optimal sets.

Let $\mathcal{C}$ denote the collection of all critical sets. In the full version, we give a modification of our algorithm that works with ties by intuitively mimicking a depth-first search, backtracking if it reaches a critical set. We are then able to bound its running time in terms of $|\mathcal{C}|$.

Theorem 5. *The implementation of a choice function for substitutable preferences with ties (condition 2) can be done with $O(\text{poly}(|D|, |\mathcal{C}|))$ queries to the verifier and the preference relations.*

From these results, we infer that in preference relations with ties and satisfying condition 2 of substitutability, given efficient verification, the computational complexity is captured by the number of critical sets. When it is large, we have examples that are hard for all algorithms; when it is polynomial, our algorithm is efficient.

Conclusion

This paper takes a step toward understanding the complexity of stable matchings in many-to-one settings. It was previously known that substitutability was the "right" condition for existence of stable matchings, but we showed it does not suffice for efficiently computing them. However, it is enough to have, in addition, a *verifier* that can tell whether a set is preferred to all of its subsets. If there are ties, the situation is more complicated: there are two natural interpretations of substitutability, and they result in qualitatively different behavior in terms of computational complexity of the choice function.

The key question was the complexity of computing a choice function: when can we efficiently compute the most preferred subset for an agent, given access to her (exponentially large) preference order over subsets? While substitutability was a natural assumption given the application

to stable matchings, this work also fits in a broader research program of discovering the computational complexity of choice functions under other assumptions. Submodular maximization is a well-studied example that fits in this framework. Identifying other natural conditions under which a choice function can be implemented (exactly or approximately) is an interesting direction for future work.

Acknowledgment

Debmalya Panigrahi was supported in part by NSF grants CCF-1527084 and CCF-1535972. Yuan Deng is supported by ARO grants W911NF-12-1-0550 and W911NF-11-1-0332. Bo Waggoner is supported by a postdoctoral fellowship from the Warren Center for Network and Data Sciences at the University of Pennsylvania.

References

Abdulkadiroglu, A., and Sönmez, T. 2003. School choice: A mechanism design approach. *The American Economic Review* 93(3):729–747.

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

Abdulkadiroğlu, A.; Pathak, P. A.; and Roth, A. E. 2005. The new york city high school match. *American Economic Review* 364–367.

Abraham, D. J.; Blum, A.; and Sandholm, T. 2007. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM conference on Electronic commerce*, 295–304. ACM.

Aziz, H.; Brill, M.; and Harrenstein, P. 2013. Testing substitutability of weak preferences. *Mathematical Social Sciences* 66(1):91–94.

Edelman, B.; Ostrovsky, M.; and Schwarz, M. 2007. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *The American economic review* 97(1):242–259.

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

Gul, F., and Stacchetti, E. 1999. Walrasian equilibrium with gross substitutes. *Journal of Economic theory* 87(1):95–124.

Hanemann, W. M. 1984. Discrete/continuous models of consumer demand. *Econometrica: Journal of the Econometric Society* 541–561.

Hatfield, J. W., and Kojima, F. 2008. Matching with contracts: Comment. *The American Economic Review* 98(3):1189–1194.

Hatfield, J. W., and Kominers, S. D. 2011. Contract design and stability in matching markets. *Harvard University and Stanford University working paper*.

Hatfield, J. W., and Milgrom, P. R. 2005. Matching with contracts. *The American Economic Review* 95(4):913–935.

Hatfield, J. W.; Immorlica, N.; and Kominers, S. D. 2012. Testing substitutability. *Games and Economic Behavior* 75(2):639–645.

Irving, R. W., and Manlove, D. F. 2009. Finding large stable matchings. *Journal of Experimental Algorithmics (JEA)* 14:2.

Irving, R. W.; Manlove, D. F.; and Scott, S. 2000. The hospitals/residents problem with ties. In *Scandinavian Workshop on Algorithm Theory*, 259–271. Springer.

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

Kelso Jr, A. S., and Crawford, V. P. 1982. Job matching, coalition formation, and gross substitutes. *Econometrica: Journal of the Econometric Society* 1483–1504.

Leme, R. P. 2014. Gross substitutability: An algorithmic survey. *preprint*.

Milgrom, P., and Strulovici, B. 2009. Substitute goods, auctions, and equilibrium. *Journal of Economic theory* 144(1):212–247.

Niederle, M., and Roth, A. E. 2005. The gastroenterology fellowship market: Should there be a match? *The American economic review* 95(2):372–375.

Ostrovsky, M. 2008. Stability in supply chain networks. *The American Economic Review* 98(3):897–923.

Roth, A. E., and Peranson, E. 1997. The effects of the change in the nrmp matching algorithm. *JAMA* 278(9):729–732.

Roth, A. E., and Xing, X. 1997. Turnaround time and bottlenecks in market clearing: Decentralized matching in the market for clinical psychologists. *Journal of political Economy* 105(2):284–329.

Roth, A. E.; Sonmez, T.; and Unver, M. U. 2003. Kidney exchange. Technical report, National Bureau of Economic Research.

Roth, A. E.; Sönmez, T.; and Ünver, M. U. 2005. A kidney exchange clearinghouse in new england. *American Economic Review* 376–380.

Roth, A. E. 1984. Stability and polarization of interests in job matching. *Econometrica: Journal of the Econometric Society* 47–57.

Roth, A. E. 1996. The national residency matching program as a labor market. *Journal of the American Medical Association* 275(13):1054–1056.

Sotomayor, M. 1999. Three remarks on the many-to-many stable matching problem. *Mathematical social sciences* 38(1):55–70.