Complexity of Mechanism Design with Signaling Costs

Andrew Kephart
Dept. of Computer Science
Duke University
kephart@cs.duke.edu

Vincent Conitzer
Dept. of Computer Science
Duke University
conitzer@cs.duke.edu

ABSTRACT

In mechanism design, it is generally assumed that an agent can submit any report at zero cost (with the occasional further restriction that certain types can not submit certain reports). More generally, however, an agent of type θ may be able to report θ', but only at a cost $c(\theta, \theta')$. This cost may reflect the effort the agent would have to expend to be indistinguishable from an agent that truthfully reports θ'. Even more generally, the possible reports (or *signals*) may not directly correspond to types. In this paper, we consider the complexity of determining whether particular social choice functions can be implemented in this context.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences - Economics

General Terms

Algorithms, Economics, Theory

Keywords

automated mechanism design, signaling costs, partial verification, revelation principle

1. INTRODUCTION

In settings with multiple self-interested agents, making a desirable collective decision is complicated by the fact that agents will misrepresent their preferences if they perceive this to be in their interest. In such settings, we must design mechanisms for making decisions that result in good outcomes even when agents act strategically. One topic of particular interest to the AI community has been *automated mechanism design* [5, 6], where instead of analytically characterizing optimal mechanisms, we have a computer intelligently search through the space of possible mechanisms to find an optimal one.

In mechanism design, an agent's private information is represented by his type. Generally the focus is on mechanisms in which each agent's action consists simply of reporting his type, justified by the *revelation principle* which states that, under certain conditions, this does not come at a loss. Usually, an agent can report any type at zero cost. However, in some of the literature, which types an agent can report depends on his type. For example, in the design of online mechanisms [14], it is generally assumed that an agent cannot report an earlier arrival time than his true arrival time. Settings such as these are referred to as *mechanism design with partial verification* [11].

In these settings, it can be computationally hard even to check whether a given social choice function—the function mapping types to outcomes—can be implemented [1, 4]. This is in contrast to regular (automated) mechanism design where this is easy to check thanks to the revelation principle, which no longer holds with partial verification.

In this paper, we consider a more general model, where for every two types θ, θ', an agent with true type θ can misreport θ', but only at a cost $c(\theta, \theta')$. Reporting truthfully is free: $c(\theta, \theta) = 0$. Mechanism design with partial verification is the special case where all costs are 0 or ∞. We also consider the even more general case where the space of *signals* that the agent can send does not correspond naturally to the type space, so that there is a cost $c(\theta, s)$ for an agent of type θ to send signal s, without restrictions on c. We focus on determining, under various conditions, the complexity of deciding whether a particular social choice function can be implemented. Our positive results hold even in this most general case, and our negative results even in the less general case where signals are type reports and reporting truthfully is free (in which case we will also refer to the signals as *reports*).

The motivation for our model is quite different from that of mechanism design with partial verification (even though it is a straightforward mathematical extension). In mechanism design with partial verification, as the name suggests, the designer has access to information (e.g., login time) that allows her to directly detect certain types of misreporting. This motivation has pushed existing extensions of the model in a somewhat different direction from ours, such as mechanism design with probabilistic verification [4], where an agent can misreport but is caught with a probability that depends on the true and reported type, and if caught the agent can be punished.

The inspiration for our model primarily comes from settings where a principal assesses an agent, perhaps based on information provided by that agent, and makes a decision. Examples include university admission, dating, insurance rate-setting, taxation etc. In each case, the agent, who knows he is being evaluated, may be able to present himself more favorably (i.e., send a better signal) at some amount of effort that depends on his true type.

In economics, such *signaling games* are often studied, with perhaps the most famous example being Spence's education model [15]. In this model, education serves as a useful signal of an agent's abilities to a potential employer, but is otherwise useless. In equilib-

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

rium, high types end up more educated than lower types, thereby signaling their type to the employer – and it is not worth it to lower types to "pool" with the higher types by obtaining education, because doing so comes at a much greater cost to the lower types. A mechanism design model with signaling costs, similar to ours, has also been proposed [9]. They show conditions for implementability when the agent can send multiple signals and the cost of misrepresentation increases with the magnitude of it.

We believe that these types of consideration will also be increasingly important in machine learning. For example, consider an e-commerce setting where an algorithm classifies visitors to a website in order to determine whether to make them a special offer on a product. When users get wind of this, they may, at some cost to themselves, attempt to change their classification, for example by erasing their cookies, using a different URL to reach the website, etc. Indeed the problem of *adversarial classification* has already received some attention in machine learning, motivated by detecting spam, intrusions, fraud, etc. [7, 3]. Our contribution here is to make the link to mechanism design and create the right machinery for this on the mechanism design side. Another key difference is our work is motivated by classifying *agents* while adversarial classification emphasizes classifying *actions* taken by agents. This makes it much easier to for us to create this link. We can abstract away the classifier to the cost function, whereas, when classifying actions the specific form of the classifier matters more.

2. MODEL

Throughout, we focus on the case of a single agent. The agent draws a type θ from a set Θ. Subsequently, the agent chooses a single signal s to emit from a set S (possibly but not necessarily, $S = \Theta$). The agent experiences a cost $c(\theta, s)$ (possibly infinite) from doing so. The mechanism maps s to an outcome (e.g., allocation) $o \in O$, as well as, if we allow for transfers, a transfer t to the agent. The agent's resulting utility is $u(\theta, s, o, p) = v(\theta, o) + t - c(\theta, s)$, where $v(\theta, o)$ is the agent's valuation for the outcome.

Let $A : S \to O$ denote the mapping of signals to outcomes, let $T : S \to \mathbb{R}$ denote the mapping of signals to transfers to the agent, and let $M = (A, T)$ be the entire mechanism. We will simply drop t and T in cases where transfers are not possible. As always, the designer commits to M first, after which the agent responds with some $s^* \in \arg\max_{s \in S} u(\theta, s, A(s), T(s))$.

Let $F : \Theta \to O$ be a *social choice function (SCF)* that we seek to implement. We say that M *implements* F if for every $\theta \in \Theta$, there exists some $s^* \in S$ with $c(\theta^*, s) < \infty$ such that (1) $s^* \in \arg\max_{s \in S} u(\theta, s, A(s), T(s))$ and (2) $A(s) = F(\theta)$. In standard mechanism design, where $S = \Theta$ and $c(\cdot, \cdot) = 0$ everywhere, the revelation principle holds, so that one can assume without loss of generality that $A(\theta) = F(\theta)$. However, as is well known, this is already no longer true in the partial verification setting (where $c : \Theta \times \Theta \to \{0, \infty\}$ with $c(\theta, \theta) = 0$), and therefore also not in our more general setting.

2.1 Example: Car Insurance

We include an example to illustrate the model. Brad would like to buy car insurance from Sally. Brad is either a *safe* driver or a *risky* driver. Sally would like to sell to Brad if he is *safe* and not sell to him if he is *risky*. In the language of our model this gives:

$\Theta = \{safe, risky\}$
$O = \{sale, no\ sale\}$
$F = \{safe \to sale, risky \to no\ sale\}$

Sally requires that Brad take a driving test, the result of which can be *perfect*, *good*, or *bad*. How much effort he needs to put into the test depends on both his true type, and which result he wants to achieve. This effort could take many forms (getting a good night's sleep before the test, bribing the test proctor, etc). Specifically:

$c =$

	perfect	*good*	*bad*
safe	10	0	0
risky	30	15	0

Brad's value for insurance also depends on his type: the worse of a driver he really is, the more valuable insurance is to him. We have:

$v =$

	sale	*no sale*
safe	12	0
risky	20	0

Now, Sally must choose a mechanism that maps the result of the test to an outcome. In the case without transfers, the naïve mechanism, $A = \{perfect \to sale, good \to sale, bad \to no\ sale\}$ wouldn't implement F. This is because if Brad has the *risky* type, it would then be optimal for him to put in the 15 units of effort cost to a achieve a score of *good* and get the 20 units of value for being insured.

A mechanism (still without transfers) that would succeed at implementing F is $A = \{perfect \to sale, good \to no\ sale, bad \to no\ sale\}$. Under this mechanism, if Brad has type *safe* it would be worth it to him to put in the effort to achieve a score of *perfect* on the test and obtain insurance; but if he has type *risky*, it is not worth the 30 units of effort cost to do so.[1]

With transfers, we could of course use the above mechanism with transfers set to zero, but other possibilities emerge as well. For example, the mechanism $M(A, T)$ with $A = \{perfect \to sale, good \to sale, bad \to no\ sale\}$ and transfers $T = \{perfect \to 0, good \to -6, bad \to 0\}$ would implement F, for the following reasons. If Brad has the *risky* type then it is no longer worth it for him to test as *good* (the effort cost of 15 plus the required transfer of 6 would outweigh the 20 benefit from insurance), but if he has the *safe* type he is happy to pay 6 for being insured (with no effort cost).[2]

In the example above, there is a clear distinction between the type space and the signal space; in fact, they do not even have the same size. However, the example can be modified to make the type space and the signal space the same, by introducing a new type *supersafe* that can obtain a score of *perfect* (or any other score) at no cost, which naturally Sally would like to insure. Then, we can relabel *perfect* as a report of *supersafe*, *good* as *safe*, and *bad* as *risky*. Hence, Brad would always be able to report his true type at zero cost.

Because these modifications do not change the behavior of the *safe* and *risky* types at all, the analysis above remains unchanged. It follows that the revelation principle does not hold in this context (as is the case in the partial verification model): without transfers, F is not truthfully implementable but it is (nontruthfully) implementable. It also shows that we cannot solve the problem by reducing it to mechanism design without costly signaling by somehow embedding the reporting costs into the valuations or transfers.

In the traditional setting without costs, it is straightforward to

[1]Note that if the effort that Brad had to expend as a *safe* driver to achieve the *perfect* result were increased to anything greater than 12, then F would be no longer implementable (without transfers) at all.

[2]Note that if we set $c(safe, perfect) > 22$ and $c(safe, good) > 7$, then even with transfers, no mechanism could implement F. This is because under these conditions, whenever it is worth it for a *safe* type to obtain insurance, then it is also worth it for a *risky* type, no matter the transfers.

		Transfers (T)		No Transfers (NT)	
		Two Outcomes (TO)	*Injective SCF (FI)*	*Two Outcomes (TO)*	*Injective SCF (FI)*
Free Utilities (FU)	*Unrestricted Costs (U)*	NP-c	**NP-c**	NP-c	**NP-c**
	$\{0, \infty\}$ *Costs (ZI)*	NP-c	**NP-c**	NP-c	**P**
Targeted Utilities (TU)	*Unrestricted Costs (U)*	**NP-c**	**P**	NP-c	**P**
	$\{0, \infty\}$ *Costs (ZI)*	**NP-c**	**P**	NP-c	**P**

Figure 1: Complexity of implementation results with non-constant $|\Theta|$ and non-constant $|S|$. Bold font represents new results proven in this paper; others were proven in, or follow immediately from, work by Auletta et al.

check whether a particular social choice function can be implemented, precisely due to the revelation principle; but as we will see, with general costs it is (in some cases) NP-hard.

3. COMPLEXITY OF IMPLEMENTATION

In this paper, we are interested in the complexity of the following problem.

DEFINITION 1 (IMPLEMENTATION). *We are given an instance I consisting of a type set Θ, a signal set S, an outcome set O, a cost function $c : \Theta \times S \to \mathbb{R}$, a valuation function $v : \Theta \times O \to \mathbb{R}$, an SCF $F : \Theta \to O$, and potentially some additional restrictions on the mechanism. We are asked whether F can be implemented.*

The restrictions correspond to (1) the case where transfers are not allowed and (2) the case where for every type, we have a "target" utility $g(\theta) \in \mathbb{R}$ that it should obtain.

The IMPLEMENTATION problem is in NP: given the outcome function $A : S \to O$ and a mapping $E : \Theta \to S$ with $A(E(\theta)) = F(\theta)$, we can check in polynomial time whether there is a transfer function (possibly required to be 0) such that (1) the agent is best off following E and (2, if necessary) each type gets its target utility. (One can write a linear program with the transfers as variables for this problem.[3]) As we will see, in general, it is NP-complete. We will be interested in special cases where the problem is tractable.

If $|\Theta|$ is constant, IMPLEMENTATION is in P, by the following argument. We can do a brute-force search over all E as $|E| = |S|^{|\Theta|}$. This will restrict A on the signals that are in the image of E. For the remaining signals, if transfers are allowed, we can impose a sufficiently negative transfer on such signals that the agent will never choose them. If transfers are not allowed, we can check for each such signal s separately whether we can set $A(s)$ so that no type will choose s. At this point, we have both A and E and can check whether they can be made to work as above.

If $|S|$ constant and we are in the setting without transfers, IMPLEMENTATION is in P, by the following argument. We can do a brute-force search over all $M = A$ as $|A| = |O|^{|S|}$ (a mechanism consists in assigning an outcome to each signal). It is easy to then check whether each type can maximize utility by choosing a signal that results in its assigned outcome.

We currently do not know the complexity of the case where transfers are allowed, and $|S|$ is constant but $|\Theta|$ is not. Of course, this cannot happen in mechanisms where $S = \Theta$.

In the remainder of the paper, we consider the following four different dimensions of the problem. For each dimension, we only consider its two extremes, leading to $2^4 = 16$ cases. Figure 1 shows our results.

Utilities We consider implementation with free utilities (FU) and with targeted utilities (TU). In the latter case we are given a function $g : \Theta \to \mathbb{R}$, and restrict attention to implementations where type θ gets utility $g(\theta)$.

Transfers We consider implementation with transfers (T) and without (NT).

Signaling Costs We consider instances where the signaling costs are unrestricted (U) and instances where signaling costs lie in $\{0, \infty\}$ (ZI).

Outcomes We consider SCFs with only two outcomes (TO) and SCFs that are injective (FI) – i.e., for every outcome there is at most one type that gets it.[4]

We refer to each of the 16 combinations with an acronym. For instance, FU/NT/U/FI is the combination of free utilities, no transfers, unrestricted signaling costs, and an SCF that is injective. Clearly, ZI is a special case of U, but for the other dimensions, it is not immediately clear which extreme point is easier. For example, one might think that TO cases are easier than FI cases because the former have only two outcomes—but, perhaps surprisingly, this is not the case: for example, we show that FU/NT/ZI/FI can be solved in polynomial time, even though FU/NT/ZI/TO is NP-complete as shown in [1].

3.1 Cases Implied by Earlier Work

For six of the two-outcome cases, NP-completeness results have either been proven in or follow immediately from [1]. (As in our paper, their hardness results hold in the setting where $S = \Theta$ and $c(\theta, \theta) = 0$ for all $\theta \in \Theta$.)

THEOREM 1 ([1]). *FU/T/ZI/TO is NP-complete.*

The hardness of FU/T/U/TO immediately follows.

THEOREM 2 ([1]). *FU/NT/ZI/TO is NP-complete.*

The hardness of FU/NT/U/TO immediately follows.

THEOREM 3. *TU/NT/ZI/TO is NP-complete.*

The hardness of TU/NT/U/TO immediately follows.

[3]Note that this generalizes the observation that this problem is in P when $S = \Theta$ and $c(\cdot, \cdot) = 0$ everywhere, because by the revelation principle in that case it is sufficient to consider the case where E is the identity function and $A = F$.

[4]These two cases are the extremes, in the sense that two is the smallest nontrivial number of outcomes and $|\Theta|$, which corresponds to the injective case, is the largest. Moreover, they both correspond to natural settings, as our examples illustrate.

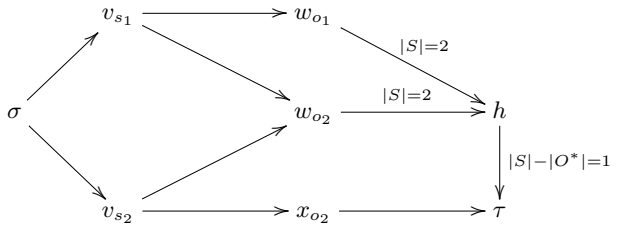

Figure 2: MAX-FLOW corresponding to instance where $O^+_{s_1} = \{o_1, o_2\}$, $O^=_{s_1} = \varnothing$, $O^+_{s_2} = \{o_2\}$, $O^=_{s_2} = \{o_2\}$, and $O^* = \{o_2\}$. Edges without a label have capacity 1.

PROOF. If a FU/NT/ZI/TO instance has a solution, then in any successful implementation, the agent's final utility when it draws type θ is $v(\theta, F(\theta))$. This is because in FU/NT/ZI/TO, there are no transfers or signaling costs. As a result, the FU/NT/ZI/TO instance is equivalent to the TU/NT/ZI/TO instance that has $g(\theta) = v(\theta, F(\theta))$ and is otherwise identical. Hence, TU/NT/ZI/TO is NP-complete. □

In the remainder of the paper, we prove our new results.

3.2 Easy Cases

We now turn to our cases where it is easy to find whether an implementation exists. As it turns out, if we set things up correctly, we can show all of them to be easy using the same algorithm. Say that a version of our problem *predetermines utilities* if given the instance, one can compute in polynomial time a utility for each type such that, *if* the instance has a solution, then that is the utility that that type has in all solutions. Any version of the problem with targeted utilities predetermines utilities. Also, any version of the problem with zero/infinity costs and no transfers predetermines utilities, because in this case, $u(\theta, s, f(\theta), p) = v(\theta, f(\theta)) + p - c(\theta, s) = v(\theta, f(\theta))$ in any solution. The main theorem we prove in this section is the following.

THEOREM 4. *If a variant of IMPLEMENTATION (1) predetermines utilities and (2) has injective SCFs (FI) then we can solve it in polynomial time by reduction to MAX-FLOW.*[5]

Theorem 4 allows us to solve TU/T/U/FI (and thus TU/T/ZI/FI), TU/NT/U/FI (and thus TU/NT/ZI/FI), and FU/NT/ZI/FI in polynomial time. To prove it, we first need the following lemma.

LEMMA 1. *Let $u^*(\theta)$ be the predetermined utility of θ. Then, we can compute in polynomial time a function $T^* : S \times O \to \mathbb{R}$ such that if I has a solution $M = (A, T)$, and $A(s) = o$, then $T(s) = T^*(s, o)$.*

PROOF. In variants of the problem without transfers, the lemma is trivially true by setting $T^* = 0$. In the remainder, we focus on variants where transfers are allowed. Set

$$T^*(s, o) = \max\{p : (\forall\theta) v(\theta, o) - c(\theta, s) + p \leq u^*(\theta)\}$$

This is easy to compute by computing for every θ separately $u^*(\theta) - v(\theta, o) + c(\theta, s)$, and taking the minimum of these.

We must prove this function has the desired property. To do so, let $M = (A, T)$ be a mechanism that solves I. If for some s, $T(s) > T^*(s, A(s))$, then by the definition of T^* there exists some type θ with $T^*(s, A(s)) = u^*(\theta) - v(\theta, A(s)) + c(\theta, s)$, or equivalently, $v(\theta, A(s)) - c(\theta, s) + T^*(s, A(s)) = u^*(\theta)$. The utility that θ would get from sending s under M is therefore $v(\theta, A(s)) - c(\theta, s) + T(s) > v(\theta, A(s)) - c(\theta, s) + T^*(s, A(s)) = u^*(\theta)$, so it is able to (and therefore will) obtain a larger utility than it is supposed to.

Now, define $M^* = (A, T^*)$ (weakly increasing M's transfer function to T^*). If under M, it was optimal for θ to report s, then this is also the case for M^*, for the following reason. Reporting s will still give θ a utility of exactly $u^*(\theta)$ (it is at least this much because transfers have only increased, and at most this much by the definition of T^*). On the other hand, reporting another s' will give θ a utility of at most $u^*(\theta)$ by the definition of T^*. Because A is unchanged, it follows that M^* is also a solution to I. □

We are now ready to prove Theorem 4.

PROOF OF THEOREM 4. We will use u^* (which is available due to utilities being predetermined) and T^* (which is available by Lemma 1) to define, for every s and o, whether letting $A(s) = o$ is *safe* and whether it is *satisfying*. "Safe" means that no type will obtain a strictly higher utility from sending s than it is supposed to, and "satisfying" means that it is safe and allows the type that is supposed to get o to send s to get o at the right utility. Let $O^* = \{o | (\exists\theta) F(\theta) = o\}$. Then, a mechanism is a solution if and only if (1) all outcome assignments are safe and (2) for each $o \in O^*$ there is at least one satisfying assignment of it to a signal. This is so because every type is able to get the correct outcome at the correct utility, and is not able to do better by deviating.

Safe $O^+_s \triangleq \{o | (\forall\theta)\ u^*(\theta) \geq v(\theta, o) - c(\theta, s) + T^*(s, o)\}$.

Satisfying $O^=_s \triangleq \{o \in O^+_s \cap O^* \mid U^*(F^{-1}(o)) = v(F^{-1}(o), o) - c(F^{-1}(o), s) + T^*(s, o)\}$.

We use these definitions to create a MAX-FLOW instance $G = (V, E)$ as follows (an illustration is given in Figure 2). V consists of: A start node σ; a signal node v_s for each $s \in S$, an outcome node w_o; for each $o \in O$, another outcome node x_o for each o in O^*; a single helper node h; and a final node τ. E consists of: an edge between σ and each v_s; for each s, edges between v_s and each w_o with $o \in O^+_s$ and each x_o with $o \in O^=_s$; edges between each w_o and h (with capacity $|S|$); an edge between h and τ (with capacity $|S| - |O^*|$); and edges between each x_o and τ. Unless otherwise specified, edges have capacity 1. Our goal is to get a flow of $|S|$.

At a high level, solving MAX-FLOW on G tests whether a bipartite matching can be found between some subset of the signals and O^*, while simultaneously assigning to each unmatched signal an outcome that will not incentivise any type to deviate to it. We now prove the equivalence formally.

I has a solution $\implies$ G has a flow of $|S|$. Let M be a mechanism which solves I with transfers T^*; we will create a flow of size $|S|$. Send one unit of flow from σ to each signal node v_s. Then, for each θ, there is a distinct s such that $F(\theta) \in O^=_s$; send one unit of flow from the corresponding v_s to $x_{F(\theta)}$ and then on to τ. For each remaining s, $A(s)$ must be a safe assignment; send one unit of flow from v_s to $w_{A(s)}$, and then via h on to τ. (Note there is at most $|S| - |O^*|$ such flow so that it does not exceed the capacity on the edge (h, τ).) It is straightforward to check that this is a valid flow of size $|S|$.

G has a flow of $|S|$ $\implies$ I has a solution. Given a flow of size $|S|$, we will create a mechanism $M = (A, T^*)$ that is a solution to I. Due to the capacity on (h, τ), each edge (x_o, τ) must have

[5]If, in addition, $|S| = |\Theta|$, then we can reduce to the problem of perfect bipartite matching. We omit the proof due to space constraint.

one unit of flow on it. Thus, for each x_o, there exists some v_s such that (v_s, x_o) has a unit of flow on it (by flow integrality); moreover, these v_s are all distinct, because exactly one unit of flow passes through each v_s. If (v_s, x_o) has a unit of flow on it, set $A(s) = o$. Thereby, for each $o \in O^*$, there is a satisfying assignment of o to a signal s (Condition (2) above). For any remaining s, there must be some w_o such that there is a unit of flow on (v_s, w_o) (by flow integrality); in this case, set $A(s) = o$. This is a safe assignment (Condition (1) above). Thus, the mechanism meets the two conditions for it to solve I. □

3.2.1 Example: Identification

This easiness result shows that we can solve (at scale) several special cases of mechanism design with signaling costs. In our opinion, the most interesting of these is free utilities, no transfers, $\{0, \infty\}$ signaling costs, and an injective SCF (FU/NT/ZI/FI).[6] Conceptually, the goal here is *identification*: the mechanism needs to learn the type of the agent exactly. The following example illustrates this.

It is early Christmas Morning and Santa Claus has almost finished delivering gifts to the small children of the world. There are only three left who have not received anything: Johnny, Molly, and Timmy. In his sack, Santa Claus has a specific gift for each, as well as a lump of coal. Each child should get its own gift and not anyone else's.[7] Thus:

$\Theta = \{Molly,\ Johnny,\ Timmy\}$
$O = \{G_{Johnny},\ G_{Molly},\ G_{Timmy},\ coal\}$
$F = \{* \to G_*\}$

Upon arriving inside a house Santa Claus must decide which gift to leave behind. Unfortunately, he does not know in which house each child lives. All he can observe is what food has been left out for him by whoever lives there. Luckily, Santa Claus knows which foods each child is capable of preparing and setting out. We refer to the set of foods that *child* can put out as S_{child}. We have:

$S = \{coffee,\ cookies,\ eggnog,\ gingerbread\}$

S_{Johnny} = {*cookies, gingerbread*}
S_{Molly} = {*coffee, cookies*}
S_{Timmy} = {*coffee, eggnog, gingerbread*}

Each child has preferences over the gifts. These only need to be specified up to order, as we do not have transfers or complex costs.

$v_{Johnny} = G_{Timmy} > G_{Molly} > G_{Johnny} > coal$
$v_{Molly} = G_{Timmy} > G_{Molly} > G_{Johnny} > coal$
$v_{Timmy} = G_{Johnny} > G_{Timmy} > G_{Molly} > coal$

When Santa arrives inside a house and sees what food has been put out for him, what gift should he leave? We can translate this problem instance into the appropriate MAX-FLOW instance (or work it out in our head) and solve for a mechanism that Santa can use. As it turns out, the only way that works is:

$M = \{coffee \to G_{Molly},\ cookies \to G_{Johnny},\ eggnog \to G_{Timmy},\ gingerbread \to coal\}$

Note the usefulness to Santa Claus of being able to give *coal*. Without *coal*, there is no mechanism that implements F: no matter which gift he gave for a signal of *gingerbread*, either *Johnny* or *Timmy* would want to set out *gingerbread* and receive a different gift than he was supposed to.

[6]This setting restricts that of mechanism design with partial verification by requiring an injective SCF, and generalizes it by allowing $S \neq \Theta$ and $c(\theta, \theta) \neq 0$.

[7]Note that the fact that coal is a possible outcome does not pose a problem for the injectivity of the SCF, which specifies that no child should ever receive coal.

3.3 Free Utilities / Transfers / Injective Social Choice Function

We now move on to our cases where it is hard to find whether an implementation exists.

THEOREM 5. *FU/T/ZI/FI is NP-complete.*

This immediately implies FU/T/U/FI is NP-complete.

PROOF. We reduce from the MINSAT problem, in which we are given a Boolean formula in conjunctive normal form with m clauses and a number n, and are asked whether there is an assignment that satisfies at most n clauses. MINSAT is NP-complete [12]. Given an instance I of MINSAT, we construct the following instance I^* of FU/T/ZI/FI.

The types are:

- n "rescue" types $r_1, \ldots, r_n$.
- m clause types $k_1, \ldots, k_m$. In a slight abuse of notation, we will also refer to the corresponding clauses by $k_1, \ldots, k_m$.
- One variable gadget for each variable v in I. The gadget consists of three types: two literal types l, one for each of v^+ and v^-, and one "switch" type w_v.

We let $-l$ be the opposite literal of l (i.e., $-l = v^-$ if $l = v^+$, and $-l = v^+$ if $l = v^-$). We assume without loss of generality that no clause contains a pair of opposite literals. We will use $w_l = w_{-l}$ to indicate the switch type in the same clause as l.

The reporting cost structure c is:

All costs are symmetrical, i.e. $c(\theta, \theta') = c(\theta', \theta)$.
$\forall \theta,\ c(\theta, \theta) = 0$.
$\forall r \forall k,\ c(r, k) = 0$.
$\forall k \forall l \in k,\ c(k, l) = 0$.
$\forall l,\ c(w_l, l) = 0$. (And $c(w_l = w_{-l}, -l) = 0$).
All costs not defined above are infinite.

Figure 3 illustrates the reporting cost structure. The SCF is injective, so for each type θ there is a unique outcome $o_\theta = F(\theta)$.

The valuation function v is:

$v(\theta, o_\theta) = 0$.
$\forall k \forall l \in k,\ v(k, o_l) = 1$ and $v(l, o_k) = 0$.
$\forall l \forall k,\ v(w_l, o_k) = 24|\Theta| + 12$.
All valuations not defined above are -10.

I has a solution $\Rightarrow$ I^* has a solution. Given a MINSAT solution, we construct a mechanism $M = (A, T)$ that is a solution to I^* as follows.

- No more than n clauses are satisfied. Thus, for each clause k that is satisfied we can pick a different rescue node r and set $A(r) = o_k$, $A(k) = o_r$, and $T(r) = 2$.
- For each $l, -l$ pair one literal will be true and the other will be false. Assume without loss of generality that l is the true literal. Let $A(w_l) = o_l$, $A(l) = o_{w_l}$, and set $T(w_l) = 2$.
- For all $A(\theta)$ and $T(\theta)$ not defined above, let $A(\theta) = o_\theta$ and $T(\theta) = 0$.

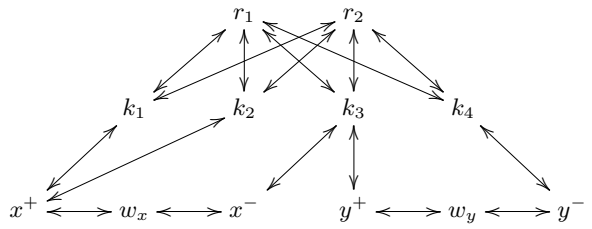

Figure 3: Reporting cost structure of the FU/T/ZI/FI instance corresponding to the MINSAT instance: $(x^+) \wedge (x^+) \wedge (x^- \vee y^+) \wedge (y^-)$ **with n = 2. A (θ, θ') edge indicates that type θ can report type θ' at zero cost.**

We will now show that $M = (A, T)$ solves I^*. Note that $c(\theta, A^{-1}(o_\theta)) = 0$ for all θ. Thus, each θ can report at zero cost the type that would lead to its intended outcome o_θ.

Because the largest transfer by the mechanism is 2, a switch or rescue type θ will obtain a utility of no higher than -8 from any outcome they could obtain other than o_θ , and thus will prefer to report $A^{-1}(o_\theta)$ for a utility of 0.

We now show that each literal type l will be best off reporting $A^{-1}(o_l)$. If l is false, then it will get a utility of 2 for reporting w_v, which leads to o_l. This is the maximum possible utility for a literal type, so this is an optimal course of action. On the other hand, if l is true, then each clause k that it is in must be satisfied. Thus, for each such k, $A(k) = o_r$ for some r. Thus l would get a utility of -10 for reporting k, -8 for reporting w_l (which would lead to o_{-l}), and 0 for reporting truthfully (which leads to o_l). So l will report truthfully and get the correct outcome.

Finally, we show that each clause type k is best off reporting $A^{-1}(o_k)$. If k is satisfied, k can get a utility of 2 by reporting the r such that $r = A^{-1}(o_k)$. Because 2 is the maximum transfer, k could only hope to achieve a larger utility by reporting in a way that results in some o_l with $l \in k$ and a transfer of 2; however, o_l only comes with a transfer when it results from reporting a switch type, which k cannot do. So k will report $A^{-1}(o_k)$. If k is not satisfied, all the literals in it must be false. Thus, if k were to report a literal, it would instead achieve some o_{w_v}, leading to a utility of -10. Reporting a rescue type would give a utility of at most -8. Hence, k is best off reporting truthfully and achieving a utility of 0.

I^* has a solution $\Rightarrow$ I has a solution. Let $M = (A, T)$ be the mechanism that constitutes a solution to I^*. We first prove a lemma, (a),: for any $\theta_1 \neq \theta_2$, if $A(\theta_2) = o_{\theta_1}$, then $T(\theta_2) - T(\theta_1) < 12|\Theta|$. This holds because of the following. First, because F is injective, A must decompose the type space into cycles of the form $o_{\theta_1} = A(\theta_2), o_{\theta_2} = A(\theta_3), \ldots, o_{\theta_j} = A(\theta_1)$, where $1 \leq j \leq |\Theta|$. Now suppose, for the sake of contradiction, that for some such cycle, $T(\theta_2) - T(\theta_1) \geq 12|\Theta|$.We know that $T(\theta_3) - T(\theta_2) > -12$, because for all o (including $o = o_{\theta_1}$), $v(\theta_2, o_{\theta_2}) - v(\theta_2, o) \leq 11 < 12$. Adding this inequality to the previous one, we obtain, $T(\theta_3) - T(\theta_1) > 12(|\Theta| - 1)$. Similarly, $T(\theta_4) - T(\theta_3) > -12$, and adding this, we obtain $T(\theta_4) - T(\theta_1) > 12(|\Theta| - 2)$. Continuing this, we eventually find $T(\theta_j) - T(\theta_1) > 12(|\Theta| - j + 2)$, and subsequently $T(\theta_1) - T(\theta_1) > 12(|\Theta| - j + 1)$—but this is a contradiction, because the left-hand side is 0 and the right-hand side is at least 12.

We use lemma (a) to prove another lemma, (b), which is that there can be no literal l and clause k such that $A(l) = o_k$. For the sake of contradiction, suppose this were the case. There are two possibilities. One is that $A(w_l) = o_l$. In this case, set $\theta_1 = l, \theta_2 = w_l, \theta_3 = A^{-1}(o_{w_l})$; these are part of a cycle. By two applications of lemma (a), we obtain $T(\theta_3) - T(\theta_1) < 24|\Theta|$. But $v(w_l, A(\theta_1)) - v(w_l, A(\theta_3)) = v(w_l, o_k) - v(w_l, o_{w_l}) = 24|\Theta| + 12$, so w_l would prefer reporting θ_1, and we have a contradiction. The other possibility is that $A(w_l) \neq o_l$. In this case, w_l and l belong to different cycles. For w_l not to misreport l (to obtain o_k), it must be the case that $T(A^{-1}(o_{w_l})) - T(l) \geq 24|\Theta| + 12$. By lemma (a), $T(A^{-1}(o_{w_l})) - T(w_l) < 12|\Theta|$, and also $T(A^{-1}(o_l) - T(l) < 12|\Theta|$. If we subtract the latter two inequalities from the one before it, we obtain $T(w_l) - T(A^{-1}(o_l) > 12$. But then l will prefer to report w_l instead of $A^{-1}(o_l)$, because $v(l, o_l) - v(l, A(w_l)) \leq 11$, and we again have a contradiction.

Next, we show that, for any literal l, either $A(l) = o_l$ or $A(-l) = o_{-l}$. This is because at least one of o_l and o_{-l} is not equal to $A(w_l)$; without loss of generality, suppose it is o_l. Two possibilities remain: $A(l) = o_l$ or $A(k) = o_l$ for some k. We must rule out the latter. If $A(k) = o_l$, then somewhere in the corresponding cycle of misreports there must be some l' and k' such that $A(l') = o_{k'}$—but this contradicts lemma (b).

Then, create a truth value assignment such that $l =$ true implies that $A(l) = o_l$. This assignment must solve the MINSAT instance. For consider any clause k satisfied by this assignment. We show that $A^{-1}(o_k)$ must be a rescue node. Since k is satisfied, there must be some $l \in k$ such that k can obtain o_l by reporting l. Then, it cannot be the case that $A(k) = o_k$. For the sake of contradiction, suppose this were so; then on the one hand, $T(k) - T(l) \leq 0$ to keep l from misreporting k, and on the other hand, $T(k) - T(l) \geq 1$ to keep k from misreporting l—but this gives a contradiction. So $A^{-1}(o_k)$ cannot be k. It cannot be another clause node or a switch node, because these cannot be reported by k. It cannot be a literal by lemma (b). Hence, it must be a rescue node. Because there are only n rescue nodes, it follows that there are at most n satisfied clauses. □

3.4 Free Utilities / No Transfers / Unrestricted Edge Costs / Injective Social Choice Function

THEOREM 6. *FU/NT/U/FI is NP-complete.*

PROOF. We reduce from satisfiability. Given an instance I of SAT, we construct the following instance I^* of FU/NT/U/FI.

The types are:

- For each clause k in I, a *clause gadget*, which consists of one clause type, k, and one "displacer" type, d_k.
- For each variable v in I, a *variable gadget* which contains a type (l, k) for every $l \in \{v^+, v^-\}$ and every $k \ni l$. (From here on, when we refer to (l, k), we implicitly assume $l \in k$.)

Let $-l$ be the opposite literal of l (i.e., $-l = v^-$ if $l = v^+$, and $-l = v^+$ if $l = v^-$). We assume without loss of generality that no clause contains a pair of opposite literals. The SCF is injective, so for each type θ there is a unique outcome $o_\theta = F(\theta)$.

The reporting cost structure c is:

$c(d_k, k) = 0.$

$\forall (l, k),\ c(k, (l, k)) = 0.$

$\forall (l, k) \forall (-l, k'),\ c((l, k), (-l, k')) = 2.$

$\forall (l, k),\ c((l, k), d_k) = 2.$

All other costs are infinite.

Figure 4 illustrates the reporting cost structure.
The valuation function v is:

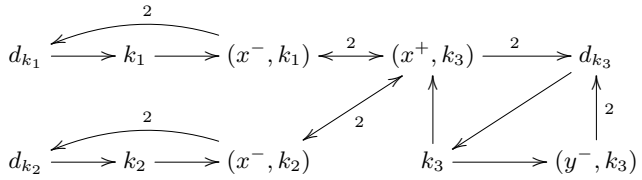

Figure 4: Reporting cost structure of the FU/NT/U/FI instance corresponding to the SAT instance: $(x^-) \wedge (x^-) \wedge (x^+ \vee y^-)$. Edges without a label have cost 0.

$v(d_k, o_k) = 10.$
$v(d_k, o_{d_k}) = 5.$
$v(k, o_{d_k}) = -10.$
$\forall (l,k),\ v(k, o_{(l,k)}) = -1.$
$\forall (l,k) \forall k' \neq k,\ v((l,k), o_{k'}) = 1.$
$\forall (l,k), v((l,k), o_k) = -10.$
$\forall (l,k) \forall (l',k') \neq (l,k),\ v((l,k), o_{(l',k')}) = -10.$
All other valuations are 0.

I has a solution $\implies$ I^* has a solution. Given an assignment that solves I, we construct a mechanism $M = (A, T)$ (where necessarily $T = 0$) which solves I^*. For each clause k, choose some literal l that satisfies it, and let $A((l,k)) = o_k, A(k) = o_{d_k}$, and $A(d_k) = o_{(l,k)}$. For all remaining types θ that do not yet have $A(\theta)$ set by this (which are all of the form (l,k)) let $A(\theta) = o_\theta$. Note that if $A((l,k)) = o_k$, then $A((-l,k')) = o_{(-l,k')}$ for all $k' \ni -l$, because $-l$ must be set to *false*.

We now show that M solves I^*. If $A((l,k)) = o_{(l,k)}$, then the reporting options for (l,k) are (i) itself, for a utility of 0; (ii) d_k, for a utility of $-10 - 2 = -12$; and (iii) some $(-l,k')$ for a utility of at most $1 - 2 = -1$. Thus it will report itself, resulting in the appropriate outcome. If $A((l,k)) = o_k$, then the reporting options for (l,k) are (i) itself, for a utility of -10; (ii) d_k, for a utility of $0 - 2 = -2$; and (iii) some $(-l,k')$, for a utility of $-10 - 8$ (because we know $A((-l,k')) = o_{(-l,k')}$). Thus it will report d_k, resulting in the appropriate outcome. For each clause k, it can report (i) itself, for a utility of -10; (ii) the (l,k) such that $A((l,k)) = o_k$, for a utility of $0 - 0 = 0$; or (iii) some (l,k) such that $A((l,k)) = o_{(l,k)}$, for a utility of $-1 - 0 = -1$. Thus, it will report the (l,k) that results in the outcome o_k. Finally, a displacer type d_k can report (i) itself, for a utility of 0; or (ii) k, for a utility of $5 - 0 = 5$. Thus it will report k, resulting in the correct outcome. Hence, M solves I^*.

I^* has a solution $\implies$ I has a solution. Let $M = (A, T)$ (where necessarily $T = 0$) be a mechanism that solves I^*. For each clause k, its displacer type, d_k, values o_k more than o_{d_k}. Thus, we must have $M(k) \neq o_k$. It follows that to obtain o_k, k must report some (l,k) instead, so that $A((l,k)) = o_k$. For this l, we may then conclude for all $(-l,k')$ that $A((-l,k')) \neq o_{k'}$ This is because if it were the case that $A((-l,k')) = o_{k'}$, then (l,k) could report $(-l,k')$ to obtain a utility of $1 - 2 = -1$, and this would be strictly higher than reporting the type $A^{-1}(o_{(l,k)}) \neq (l,k)$ (because $A((l,k)) = o_k$), which would give a utility of $0 - 2 = -2$.

It follows that the following assignment is well defined: set l to *true* if and only if there exists some k such that $A((l,k)) = o_k$ (unless this results in a pair of opposite literals both being set to *false*, in which case arbitrarily choose one to set to *true*). Because for every k, there exists some $l \in k$ such that $A((l,k)) = o_k$, this is a satisfying assignment.[8] $\square$

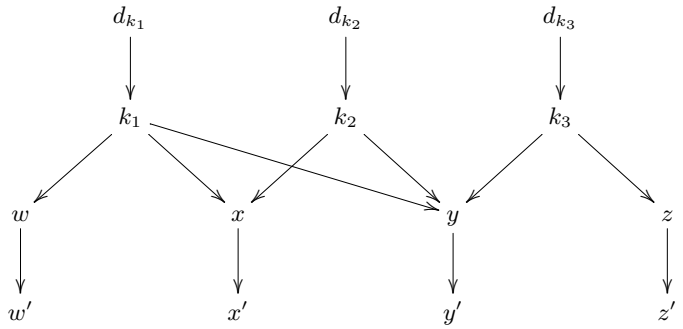

Figure 5: Reporting cost structure of the TU/T/ZI/TO instance corresponding to the Monotone SAT instance: $(w^+ \vee x^+ \vee y^+) \wedge (x^- \vee y^-) \wedge (y^+ \vee z^+)$. Edges indicate a zero cost report.

3.5 Target Utilities / Transfers / Two Outcomes

THEOREM 7. *TU/T/ZI/TO is NP-complete.*

This immediately implies that TU/T/U/TO is NP-complete as well.

PROOF. We reduce from the Monotone SAT problem (SAT where within each clause all the literals have the same sign, which is NP-complete [10]). Given an instance I of Monotone SAT we construct the following instance I^* of TU/T/ZI/TO.

The types are:

- For each clause k in I, a *clause gadget* which consists of one clause type, k, and one "displacer" type, d_k.
- For each variable w in I, a variable type w and slot type w'

The reporting cost structure c is:

$\forall \theta, c(\theta, \theta) = 0$
$\forall k,\ c(d_k, k) = 0.$
$\forall k \forall w \in k,\ c(k, w) = 0.$
$\forall w\ c(w, w') = 0$
All other costs are infinite.

Figure 5 illustrates the reporting cost structure.
The two outcomes are o^+ and o^-. Let $\neg o^+ = o^-$ and $\neg o^- = o^+$.

The social choice function F is:

$\forall k\ s.t.\ k$ only contains positive literals, $F(k) = o^+$.
$\forall k\ s.t.\ k$ only contains negative literals, $F(k) = o^-$.
$\forall d_k,\ (F(k) = o^+) \Rightarrow F(d_k) = o^-$ and $(F(k) = o^-) \Rightarrow F(d_k) = o^+$.
$\forall w,\ F(w) = F(w') = o^+.$

The valuation function v is:

[8]Note that the mechanism could be quite different from the mechanism used in the first part of this proof; even so, our argument that it produces a satisfying assignment still holds.

$\forall d_k,\ v(d_k, F(d_k)) = 0$ and $v(d_k, \neg F(d_k)) = 10$.

$\forall k,\ v(k, F(k)) = 0$ and $v(k, \neg F(k)) = -10$.

$\forall w\ v(w, o^+) = v(w', o^+) = 0$.

$\forall w\ v(w, o^-) = v(w', o^-) = -1$.

The target utility function g is simply: $\forall \theta,\ g(\theta) = 0$.

I has a solution $\Rightarrow$ I^* has a solution. We construct the mechanism $M = (A, T)$ as follows. Set all transfers to zero. For each d_k let $A(d_k) = F(d_k)$. For each k, let $A(k) = \neg F(k) = F(d_k)$. For each variable w that is positively instantiated in I, let $A(w) = o^+$, for each variable that is negatively instantiated, let $A(w) = o^-$. For each w' let $A(w') = o^+$.

We argue that M is a solution to I^*. Each w' recieves o^+. Each w will report w' to achieve an outcome of o^+. For each clause k which includes only negative literals there will be at least one variable w contained in k that is negatively instantiated in I. Thus, k can report w and receive a utility maximizing outcome of o^-. Similarly, each clause which contains only positive literals will be able to receive o^+. Finally, for each d_k, $A(d_k) = A(k) = F(d_k)$, so d_k has no choice but to give a report which leads to the appropriate outcome.

I^* has a solution $\Rightarrow$ I has a solution. Let $M = A$ be the mechanism that solves I^*. For each k, since d_k values $\neg F(d_k)$ more than $F(d_k)$, it must be the case that: $A(k) = F(d_k)$, or, there is large negative transfer for reporting k. Since $F(d_k) = \neg F(k)$ and $g(k) = 0$, either way each k cannot report itself. Thus each k must report a variable node to achieve the correct outcome—hence $\forall k \exists w\ s.t.\ A(w) = F(k)$.

Then, for each w in I, set w to *true* if $A(w) = o^+$ and to *false* if $A(w) = o^-$. This represents a solution to I, because $\forall k \exists w\ s.t.\ A(w) = F(k)$ implies that at least one of the variables in each clause will be set in a way that satisfies the clause. □

4. CONCLUSION

In this paper, we generalized the model of mechanism design with partial verification—where not every type can report every other type—to mechanism design with costly signaling, where there is a cost $c(\theta, \theta')$ for type θ to report another type θ' (or, more generally, a cost $c(\theta, s)$ for sending signal s). Building on earlier results for the partial-verification case, we characterized the complexity of determining whether a specific social choice function can be implemented under various conditions.

Future research could be devoted to expanding our complexity results, for example obtaining fixed-parameter tractability results. It could also be devoted to the case where the social choice function is not given, but rather has to be optimized as well with respect to some objective function, as is often done in automated mechanism design [5, 6]. Of course, insofar as any SCF may maximize the objective value, this problem can be no easier than the problem studied here, because the answer to the optimization version would tell us whether the SCF that maximizes the objective value is feasible or not. (In fact, in automated mechanism design with costless and unlimited misreporting, checking whether a given SCF can be implemented is easy, though the optimization problem is sometimes hard.) However, the optimization version may still allow efficient approximations. Other future research can be devoted to expanding the connection to machine learning, as discussed in the introduction. The techniques discussed in this paper may allow us to tailor classifiers to strategic agents based on our knowledge of the cost structure.[9] Also, in a companion (working) paper, we characterize when the revelation principle holds in the costly-signaling model (citation suppressed for anonymity). This generalizes earlier results for partial verification [1, 16].

[9]Note that the resulting setup is quite different from previous work on *incentive compatible machine learning* [8, 13] where an agent is not attempting to be misclassified himself, but rather is aiming to get a global classifier, constructed based on the data reported by multiple agents, to be as accurate as possible on his own data. Both this direction and ours are even more different from mechanism design *via* machine learning [2], where the approach is to use one set of bids to set prices for another set of bidders.

Acknowledgments

We thank NSF and ARO for support under grants CCF-1101659, IIS-0953756, CCF-1337215, W911NF-12-1-0550, and W911NF-11-1-0332.

REFERENCES

[1] V. Auletta, P. Penna, G. Persiano, and C. Ventre. Alternatives to truthfulness are hard to recognize. *Autonomous Agents and Multi-Agent Systems*, 22(1):200–216, 2011.

[2] M. Balcan, A. Blum, J. D. Hartline, and Y. Mansour. Reducing mechanism design to algorithm design via machine learning. *Journal of Computer and System Sciences*, 74(8):1245–1270, 2008.

[3] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[4] I. Caragiannis, E. Elkind, M. Szegedy, and L. Yu. Mechanism design: from partial to probabilistic verification. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 266–283, Valencia, Spain, 2012.

[5] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 103–110, Edmonton, Canada, 2002.

[6] V. Conitzer and T. Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 132–141, New York, NY, USA, 2004.

[7] N. N. Dalvi, P. Domingos, Mausam, S. K. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Seattle, WA, USA, 2004.

[8] O. Dekel, F. Fischer, and A. D. Procaccia. Incentive compatible regression learning. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 884–893, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[9] R. Deneckere and S. Severinov. Optimal screening with costly misrepresentation, 2007. Working paper available at http://www.severinov.com/working_papers/screening_costly_misrepresentation_jul07.pdf.

[10] M. Garey and D. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[11] J. Green and J.-J. Laffont. Partially verifiable information and mechanism design. *Review of Economic Studies*, 53:447–456, 1986.

[12] R. Kohli, R. Krishnamurti, and P. Mirchandani. The minimum satisfiability problem. *SIAM Journal on Discrete*

Mathematics, 7(2):275–283, 1994.

[13] R. Meir, A. D. Procaccia, and J. S. Rosenschein. Algorithms for strategyproof classification. *Artificial Intelligence*, 186:123–156, 2012.

[14] D. Parkes. Online mechanisms. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 16. Cambridge University Press, 2007.

[15] M. Spence. Job market signaling. *Quarterly Journal of Economics*, 87(3):355–374, 1973.

[16] L. Yu. Mechanism design with partial verification and revelation principle. *Autonomous Agents and Multi-Agent Systems*, 22(1):217–223, 2011.