Automated Mechanism Design for Classification with Partial Verification

Hanrui Zhang,[1] **Yu Cheng,** [2] **Vincent Conitzer** [1]
[1] Duke University
[2] University of Illinois at Chicago
hrzhang@cs.duke.edu, yucheng2@uic.edu, conitzer@cs.duke.edu

Abstract

We study the problem of automated mechanism design with partial verification, where each type can (mis)report only a restricted set of types (rather than any other type), induced by the principal's limited verification power. We prove hardness results when the revelation principle does not necessarily hold, as well as when types have even minimally different preferences. In light of these hardness results, we focus on truthful mechanisms in the setting where all types share the same preference over outcomes, which is motivated by applications in, e.g., strategic classification. We present a number of algorithmic and structural results, including an efficient algorithm for finding optimal deterministic truthful mechanisms, which also implies a faster algorithm for finding optimal randomized truthful mechanisms via a characterization based on convexity. We then consider a more general setting, where the principal's cost is a function of the combination of outcomes assigned to each type. In particular, we focus on the case where the cost function is submodular, and give generalizations of essentially all our results in the classical setting where the cost function is additive. Our results provide a relatively complete picture for automated mechanism design with partial verification.

1 Introduction

Agents are often *classified* into a variety of categories, some more desirable than others. Loan applicants might be classified in various categories of risk, determining the interest they would have to pay. University applicants may be classified into categories such as "rejected," "wait list," "regular accept," and "accept with honors scholarship." Meanwhile universities might themselves be classified into categories such as "most competitive," "highly competitive," etc. In line with the language of *mechanism design* (often considered part of *game theory*), we assume that each agent (i.e., the entity being classified) has a *type*, corresponding to the agent's true financial situation, ability as a student, or competitiveness as a university. This type is information that is private to the agent. In most applications of mechanism design, the type encodes the agent's *preferences*. For example, in an auction, an agent's type is how much he values the outcome where he wins the auction. In contrast, in our setting, the type does not encode the agent's preferences: in the examples above, typically any agent has the same preferences over outcomes, regardless of the agent's true type. Instead, the type is relevant to the objective function of the *principal* (the entity doing the classification), who wants to classify the agents into a class that fits their type.

Often, in mechanism design, it is assumed that an agent of any type can report any other type (e.g., bid any value in an auction), and outcomes are based on these reports. Under this assumption, our problem would be hopeless: every agent would always simply report whatever type gives the most favorable outcome, so we could not at all distinguish agents based on their true type. But in our context this assumption is not sensible: while an agent may be able to take some actions that affect how its financial situation appears, it will generally not be possible for a person in significant debt and without a job to successfully imitate a wealthy person with a secure career. This brings us into the less commonly studied domain of *mechanism design with partial verification* (Green and Laffont 1986; Yu 2011), in which not every type can misreport every other type. That is, each type has certain other types that it can misreport. A standard example in this literature is that it is possible to have arrived later than one really did, but not possible to have arrived earlier. (In that case, the arrival time is the type.) In this paper, however, we are interested in more complex misreporting (in)abilities.

What determines which types can misreport (i.e., successfully imitate) which other types? This is generally specific to the setting at hand. Zhang, Cheng, and Conitzer (2019b) consider settings in which different types produce "samples" (e.g., timely payments, grades, admissions rates, ...) according to different distributions. They characterize which types can distinguish themselves from which other types in the long run, in a model in which agents can either (1) manipulate these samples before they are submitted to the principal, by either withholding transforming some of them in limited ways, or (2) choose the number of costly samples to generate (Zhang, Cheng, and Conitzer 2019b,a, 2021). In this paper, we will take as given which types can misreport which other types; this relation may result from applying the above characterization result, or from some other model.

Our goal is: given the misreporting relation, agents' preferences, and the principal's objective, can we efficiently compute the optimal (single-agent) mechanism/classifier,

which assigns each report to an outcome/class? This is a problem in *automated mechanism design* (Conitzer and Sandholm 2002, 2004), where the goal is to compute the optimal mechanism for the specific setting (outcome space, utility and objective functions, type distribution, ...) at hand. Quite a bit is already known about the complexity of the automated mechanism design problem, and with partial verification, the problem is known to become even harder (Auletta et al. 2011; Yu 2011; Kephart and Conitzer 2015, 2016). The structural advantage that we have here is that, unlike that earlier work, we are considering settings where all types have the same preferences over outcomes. This allows us positive results that would otherwise not be available.

1.1 Our Results and Techniques

Throughout the paper, we assume agents have *utility* functions which they seek to maximize, and the principal has a *cost* function which she seeks to minimize.

General vs. truthful mechanisms. We first set out to investigate the problem of automated mechanism design with partial verification in the most general sense, where there is no restriction on each type's utility function. In light of previously known hardness results, although the most general problem is unlikely to be efficiently solvable, one may still hope to identify maximally nontrivial special cases for which efficient algorithms exist. In order to determine the boundary of tractability, our first finding, Theorem 1, shows that when the revelation principle does not hold, it is NP-hard to find an optimal (randomized or deterministic) mechanism even if (1) there are only 2 outcomes and (2) all types share the same utility function.[1] In other words, without the revelation principle, no efficient algorithm exists even for the minimally nontrivial setting. We therefore focus our attention on cases where the revelation principle holds, or, put in another way, on finding optimal truthful mechanisms.

General vs. structured utility functions. The above result, as well as prior results on mechanism design with partial verification (Auletta et al. 2011; Yu 2011; Kephart and Conitzer 2015, 2016), paints a clear picture of intractability when the revelation principle does not hold. But prior work also often suggests that this is indeed the boundary of tractability. This is in fact true if we consider optimal randomized truthful mechanisms, which can be found by solving a linear program with polynomially many variables and constraints if the number of agents is constant (Conitzer and Sandholm 2002). However, as our second finding (Theorem 2) shows, the case of *deterministic* mechanisms is totally different — even with 3 outcomes and single-peaked preferences over outcomes, it is still NP-hard to find an optimal deterministic truthful mechanism (significantly improving over earlier hardness results for deterministic mechanisms (Conitzer and Sandholm 2002, 2004)). In other words, optimal deterministic truthful mechanisms are almost always hard to find whenever types have different preferences over outcomes. This leads us to what appears to be the only nontrivial case left, i.e., where all types share the same preference over outcomes. But this case is important: as discussed above, it in fact nicely captures a number of real-world scenarios of practical importance, and will be the focus in the rest of our results.

Efficient algorithm for deterministic mechanisms. Our first algorithmic result (Theorem 3) is an efficient algorithm for finding optimal deterministic truthful mechanisms with identical preferences in the presence of partial verification. The algorithm works by building a directed capacitated graph, where each deterministic truthful mechanism corresponds bijectively to a finite-capacity s-t cut. The algorithm then finds an s-t min-cut in polynomial time, which corresponds to a deterministic truthful mechanism with the minimum cost.

Condition for deterministic optimality and faster algorithm for randomized mechanisms. We then consider randomized mechanisms. We aim to answer the following two natural questions.

- In which cases is there a gap between optimal deterministic and randomized mechanisms, and how large can this gap be?
- While LP formulations exist for optimal randomized truthful mechanisms in general, is it possible to design theoretically and/or practically faster algorithms when types share the same utility function?

The answers to these questions turn out to be closely related.

For the first question, we show that the gap in general can be arbitrarily large (Example 1). On the other hand, there always exists an optimal truthful mechanism that is deterministic whenever the principal's cost function is convex with respect to the common utility function (Lemma 1). In order to prove this, we show that without loss of generality, an optimal truthful mechanism randomizes only between two consecutive outcomes (when sorted by utility) for each type, and present a way to round any such mechanism into a deterministic truthful mechanism, preserving the cost in expectation.

For the second question, we give a positive answer, by observing that with randomization, essentially only the convex envelope of the principal's cost function matters. This implies a reduction from finding optimal randomized mechanisms with general costs, to finding optimal randomized mechanisms with convex costs, and – via our answer to the first question (Lemma 1) – to finding optimal deterministic mechanisms with convex costs. As a result, finding optimal randomized truthful mechanisms is never harder than finding optimal deterministic truthful mechanisms with convex costs. Combined with our algorithm for the latter problem (Theorem 3), this reduction implies a theoretically and practically faster algorithm for finding optimal randomized truthful mechanisms when types share the same utility function.

Generalizing to combinatorial costs. With all the intuition developed so far, we then proceed to a significantly more

[1] The *revelation principle* states that if certain conditions hold on the reporting structure, then it is without loss of generality to focus on *truthful* mechanisms, in which agents are always best off revealing their true type. We will discuss below a necessary and sufficient condition for the revelation principle to hold in our setting.

general setting, where the principal's cost is a function of the combination of outcomes for each type, i.e., the principal's cost function is *combinatorial*. This further captures global constraints for the principal, e.g., budget or headcount constraints. We present combinatorial counterparts of essentially all our results for additive costs in Section 3.

1.2 Further Related Work

Some recent research along the line of automated mechanism design includes designing auctions from observed samples (Cole and Roughgarden 2014; Devanur, Huang, and Psomas 2016; Balcan, Sandholm, and Vitercik 2018; Gonczarowski and Weinberg 2018), mechanism design via deep learning (Duetting et al. 2019; Shen, Tang, and Zuo 2019), and estimating incentive compatibility (Balcan, Sandholm, and Vitercik 2019). Most of these results focus on auctions, while in this paper, we consider automated mechanism design in a more general sense (though we focus mostly on the types of setting discussed in the introduction, which have more of a classification focus). More closely related are results on automated mechanism design with partial verification (Auletta et al. 2011; Yu 2011; Kephart and Conitzer 2015, 2016). Those results are about conditions under which the revelation principle holds (including a relevant condition to our setting discussed later), and the computational complexity of deciding whether there exists an implementation of a *specific* mapping from types to outcomes. On the other hand, we focus on algorithms for designing cost-*optimal* truthful mechanisms, which is largely orthogonal to those results.

Another closely related line of research is strategic machine learning. There, a common assumption is that utility-maximizing agents can modify their features in some restricted way, normally at some cost (Hardt et al. 2016; Kleinberg and Raghavan 2019; Haghtalab et al. 2020; Zhang and Conitzer 2021) (see also (Kephart and Conitzer 2015, 2016)). Strategic aspects of linear regression have also been studied (Perote and Perote-Pena 2004; Dekel, Fischer, and Procaccia 2010; Chen et al. 2018). Our results differ from the above in that we study strategic classification from a more general point of view, and do not put restrictions on the class of classifiers or learning algorithms to be used.

Another line of work in economics considers mechanism design with costly misreporting, where the cost is unobservable to the principal (Laffont and Tirole 1986; McAfee and McMillan 1987). These results are incomparable with ours, since they consider rather specific models, while we consider utility and cost functions of essentially any form.

2 Additive Cost over Types

Consider the classical setting of Bayesian (single-agent) mechanism design, which is as follows. The agent can have one of many possible *types*. The agent reports a type to the principal (which may not be his true type), and then the principal chooses an *outcome*. The principal does not know the type of the agent, but she has a prior probability distribution over the agent's possible types. The principal has a different cost for each combination of a type and an outcome. The goal of the principal is to design a mechanism (a mapping from reports to outcomes) to minimize her expected cost assuming the agent best-responds to (i.e., maximizes his utility under) the mechanism. The principal aims to minimize her total cost over this population of agents, which is equal to the sum of her cost over individual agents.

In this section, we focus on the traditional setting where the principal's cost is additive over types. In Section 3, we generalize our results to broader settings where the principal's cost function can be combinatorial (e.g., submodular) over types.

Notation. Let Θ be the agent's type space, and $\mathcal{O}$ the set of outcomes. Let $n = |\Theta|$ and $m = |\mathcal{O}|$ be the numbers of types and outcomes respective. Generally, we use $i \in \Theta$ to index types, and $j \in \mathcal{O}$ to index outcomes. Let $\mathbb{R}_+ = [0, \infty)$. We use $u_i : \mathcal{O} \to \mathbb{R}_+$ to denote the utility of a type i agent, and $c_i : \mathcal{O} \to \mathbb{R}_+$ to denote the cost of the principal of assigning different outcomes to a type i agent.

Let $R \subseteq \Theta \times \Theta$ denote all possible ways of misreporting, that is, a type i agent can report type i' if and only if $(i, i') \in R$. We assume each type i can always report truthfully, i.e., $(i, i) \in R$. The principal specifies a (possibly randomized) mechanism $M : \Theta \to \mathcal{O}$, which maps reported types to (distributions over) outcomes. The agent then responds to maximize his expected utility under M.

Let r_i denote the report of type i when the agent best responds:

$$r_i \in \operatorname{argmax}_{i' \in \Theta, (i,i') \in R} \mathbb{E}[u_i(M(i'))].$$

Without loss of generality, the principal's cost function can be scaled so that the prior distribution over possible types is effectively uniform. The principal's cost under mechanism M is then given by

$$c(M) = \sum_{i \in \Theta} \mathbb{E}\left[c_i\left(M\left(r_i\right)\right)\right]$$

where both expectations are over the randomness in M. Throughout the paper, given a set S, we use $\Delta(S)$ to denote the set of all distributions over S.

2.1 Hardness without the Revelation Principle

The well-known revelation principle states that when any type can report any other type, there always exists a truthful *direct-revelation* mechanism that is optimal for the principal.[2] However, this is not true in the case of partial verification (see, e.g., (Green and Laffont 1986; Yu 2011; Kephart and Conitzer 2016)). In fact, it is known (see Theorem 4.10 of (Kephart and Conitzer 2016)) that in our setting, the revelation principle holds if and only if the reporting structure R is transitive, i.e., for any types $i_1, i_2, i_3 \in \Theta$,

$$(i_1, i_2) \in R \text{ and } (i_2, i_3) \in R \implies (i_1, i_3) \in R.^3$$

[2] A direct-revelation mechanism is a mechanism in which agents can only report their type, rather than sending arbitrary messages. A mechanism is truthful if it is always optimal for agents to report their true types.

[3] To get some intuition for this characterization, suppose that $(i_1, i_2) \in R$, $(i_2, i_3) \in R$, but $(i_1, i_3) \notin R$, and we would like

We begin our investigation by presenting a hardness result, which states that when the revelation principle does not hold, it is NP-hard to find any optimal mechanism (even in the minimal nontrivial setting).

Theorem 1 (NP-hardness without the Revelation Principle)**.** *When partial verification is allowed and the revelation principle does not hold, it is* NP*-hard to find an optimal (randomized or deterministic) mechanism, even if there are only* 2 *outcomes and all types share the same utility function.*

We postpone the proof of Theorem 1, as well as all other proofs in this section, to Appendix C. In light of Theorem 1, in the rest of the paper, we focus on finding optimal truthful direct-revelation mechanisms. That is, we consider only mechanisms M where for any $(i_1, i_2) \in R$,

$$\mathbb{E}[u_{i_1}(M(i_1))] \geq \mathbb{E}[u_{i_1}(M(i_2))].$$

2.2 General vs. Structured Utility Functions

Following the convention in the literature, we assume agents always break ties by reporting truthfully. As a result, for a (possibly randomized) truthful mechanism M, the cost of the principal can be written as

$$c(M) = \sum_{i \in \Theta} \mathbb{E}[c_i(M(i))].$$

Our first finding establishes a dichotomy between deterministic and randomized mechanisms when agents can have arbitrary utility functions. On one hand, it is known that an optimal randomized mechanism can be found in polynomial time by formulating the problem as a linear program (Conitzer and Sandholm 2002). On the other hand, finding an optimal deterministic mechanism is NP-hard even in an extremely simple setting as described below.

Theorem 2 (NP-hardness with General Utility Functions)**.** *When partial verification is allowed, even when the revelation principle holds, it is* NP*-hard to find an optimal deterministic mechanism, even if there are only* 3 *outcomes and the utility functions are single-peaked (see Appendix B.1 for a definition).*

Although Theorem 2 establishes hardness for finding optimal deterministic mechanisms in most nontrivial cases, it leaves the possibility of efficient algorithms when all types have the same utility function — which, as discussed in the introduction, is the setting we focus on in this paper.

2.3 Finding Optimal Deterministic Mechanisms

In light of the previously mentioned hardness results, for the rest of this section, we focus on the setting where the revelation principle holds and all types have the same utility function.

to accept i_2 and i_3 but not i_1. That is, higher types are better, and each type (except for the top one) can make itself look a bit, but not much, better than it is. There is no truthful mechanism that achieves what we want: if we accept a report of i_2, we will end up accepting i_1 as well because it can misreport i_2. On the other hand, if we accept only i_3, then we get what we want, by relying on i_2 to non-truthfully report i_3 (whereas i_1 cannot). Hence, our goal can be achieved in a non-truthful implementation while it cannot be achieved in a truthful implementation, showing that the revelation principle does not hold in this case.

We recall and simplify some notations before we state the main result of this section (Theorem 3). Let $u : \mathcal{O} \to \mathbb{R}_+$ be the common utility function of all types. Recall that $n = |\Theta|$ is the number of types and $m = |\mathcal{O}|$ is the number of outcomes. Let $\Theta = [n] = \{1, \ldots, n\}$. For brevity, we use $\mathcal{O} = \{o_1, \ldots, o_m\} \subseteq \mathbb{R}_+$ to encode the utility function u. That is, for all $j \in [m]$, $o_j \in R_+$ is the utility of the agent under the j-th outcome. Without loss of generality, assume $o_1 = 0$, and $o_j < o_{j+1}$ for all $j \in [m-1]$.

We give an efficient algorithm (Algorithm 1) for finding an optimal deterministic mechanism when partial verification is allowed.[4] Our algorithm first builds a (capacitated) directed graph based on the principal's cost function and the reporting structure, then finds an s-t min-cut in the graph, and then constructs a mechanism based on the found min-cut. The idea is finite-capacity cuts in the graph constructed correspond bijectively to truthful mechanisms, where the capacity is precisely the cost of the principal. In particular, we use edges with ∞ capacity to ensure that if one type gets an outcome, any type that can misreport the former must get at least as good an outcome. See Figure 1 for an illustration of Algorithm 1. The following theorem establishes the correctness and time complexity of Algorithm 1.

Theorem 3 (Fast Algorithm for Finding Optimal Deterministic Mechanisms)**.** *Suppose for any $i \in [n]$ and $j \in [m]$, $c_i(o_j) \in \mathbb{N}$. Let $W = \max_{i,j} c_i(o_j)$. Algorithm 1 outputs an optimal deterministic truthful mechanism in time $O(T_{\mathsf{MinCut}}(mn, mn^2, W))$, where $T_{\mathsf{MinCut}}(n', m', W')$ is the time it takes to find an s-t min-cut in a graph with n' vertices, m' edges, and maximum capacity W'.*

We note that Algorithm 1 only finds an optimal deterministic mechanism *subject to truthfulness* — when the revelation principle does not hold, Algorithm 1 may not find an unconditionally optimal mechanism (and indeed finding that is NP-hard given Theorem 1). The same applies for all our algorithmic results.

2.4 Optimality of Deterministic Mechanisms with Convex Costs

In the previous subsection, we showed that when the revelation principle holds and all types have the same utility function, there is a min-cut-based algorithm (Algorithm 1) that finds an optimal deterministic truthful mechanism.

In this subsection, we identify an important special case where there exists an optimal truthful mechanism that is deterministic (even when randomized mechanisms are allowed). Consequently, we have an algorithm (Algorithm 1) for finding the optimal truthful mechanism that runs faster than solving a linear program. More importantly, as we will show in Section 2.5, we can essentially reduce the general case to this special case, and consequently obtain an algo-

[4] In a more empirically focused companion paper (Krishnaswamy et al. 2021), we apply a simplified version of Algorithm 1 to a special case of the problem studied in this paper. There, the goal is to find a nearly optimal *binary* classifier (i.e., $m = 2$), given only *sample access* to the population distribution over the type space.

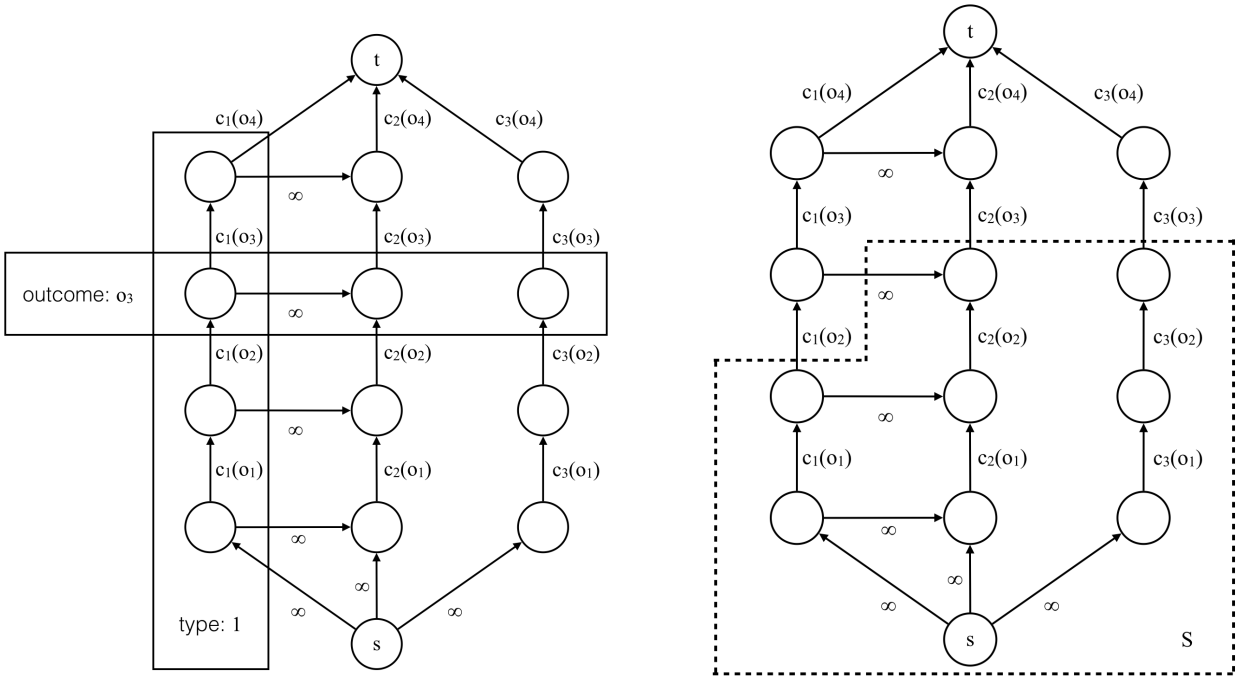

Figure 1: An example of the graph constructed in Algorithm 1. As highlighted in the left graph, each row corresponds to an outcome and each column corresponds to a type. The horizontal edges with infinite capacity correspond to the fact that type 2 can misreport as type 1. The right graph gives a possible s-t min-cut, which corresponds to a mechanism where $M(1) = o_2$, $M(2) = (o_3)$, and $M(3) = o_3$. The horizontal edges make sure that type 1 never gets a more desirable outcome than type 2, so type 2 never misreports. The cost of the mechanism M is equal to the value of the min-cut, which is $c_1(o_2) + c_2(o_3) + c_3(o_3)$.

rithm for computing the optimal truthful mechanism whose runtime is asymptotically the same as Algorithm 1.

We first show (in Example 1) that, in general, there can be an arbitrarily large gap between the cost of the optimal deterministic mechanism and that of the optimal randomized mechanism, even when restricted to truthful mechanisms and when all types share the same utility function.

Example 1 (Gap between Deterministic and Randomized Mechanisms)**.** There are 2 types $\Theta = \{1, 2\}$ and 3 outcomes $\mathcal{O} = \{o_1 = 1, o_2 = 2, o_3 = 3\}$, which encode the common utility function. The principal's cost is given by $c_1(o_1) = c_1(o_3) = \infty$, $c_1(o_2) = 0$, $c_2(o_1) = c_2(o_3) = 0$, and $c_2(o_2) = \infty$. The reporting structure R allows any type to report any other type, i.e., $R = \{(1, 1), (2, 2), (1, 2), (2, 1)\}$. Consider first the optimal truthful randomized mechanism, which as we argue below has cost 0. To make the principal's cost finite, the optimal truthful mechanism must assign outcome o_2 to type 1 with probability 1, which gives type 1 utility 2. To prevent misreporting, the mechanism must give type 2 the same expected utility. And again, to make the cost finite, it must never assign outcome o_2 to type 2. The unique way to satisfy the above is to assign to type 2 outcome o_1 with probability $1/2$, and o_3 with probability $1/2$.

Now consider any deterministic truthful mechanism. Any truthful mechanism must give both types the same utility to prevent misreporting. The only way to achieve this deterministically is to assign the same outcome to both types. However, all 3 possibilities result in infinite total cost, so all deterministic truthful mechanisms have cost infinity.

Example 1 shows that Algorithm 1 in general does not find an (approximately) optimal truthful mechanism when randomized mechanisms are allowed. In such cases, one has to fall back to significantly slower algorithms, e.g., solving the straightforward LP formulation of the problem with mn variables and n^2 constraints. It is worth noting that the LP formulation does not utilize the fact that types share an identical utility function. To address this issue, we identify an important special case where there does exist an optimal truthful mechanism that is deterministic: when the principal's cost is convex in the common utility function. More importantly, as we will show in Section 2.5, we can reduce the problem of finding the optimal randomized mechanism under general costs to the problem of finding the optimal mechanism with convex costs. First we formally define the notion of convex costs we use.

Definition 1 (Convex Costs)**.** For any $i \in \Theta$, let the piecewise linear extension $c_i^\ell : [o_1, o_m] \to \mathbb{R}_+$ of c_i be such that (1) for any $x \in \mathcal{O}$, $c_i^\ell(x) = c_i(x)$, and (2) for any $x \in [o_1, o_m] \setminus \mathcal{O}$,

$$c_i^\ell(x) = \frac{o_{j+1} - x}{o_{j+1} - o_j} \cdot c_i(o_j) + \frac{x - o_j}{o_{j+1} - o_j} c_i(o_{j+1}),$$

where $j = \max\{j' \in [m] \mid o_{j'} \le x\}$. The principal's cost function $\{c_i\}_{i \in \Theta}$ is convex if for every $i \in \Theta$, the piecewise

Algorithm 1: Finding an optimal deterministic mechanism.

Input: The set of types Θ, the principal's cost function $\{c_i\}_{i\in\Theta}$ for each type, the set of outcomes $\mathcal{O}$ (which encodes the agents' common utility function), and the reporting structure R.

Output: A deterministic truthful mechanism $M : \Theta \to \mathcal{O}$ minimizing the principal's cost.

Let $V \leftarrow (\Theta \times \mathcal{O}) \cup \{s, t\}$, $E \leftarrow \emptyset$;
Replace R with its transitive closure (using the Floyd–Warshall algorithm):
for $i_2, i_1, i_3 \in \Theta$ *where* $(i_1, i_2) \in R$ *and* $(i_2, i_3) \in R$ **do**
    $R \leftarrow R \cup \{(i_1, i_3)\}$;
end
for *each type* $i \in \Theta$ **do**
    $E \leftarrow E \cup \{(s, (i, o_1), \infty)\}$ (add an edge from s to (i, o_1) with capacity ∞) ;
    for *each outcome* $j \in [m-1]$ **do**
        $E \leftarrow E \cup \{((i, o_j), (i, o_{j+1}), c_i(o_j))\}$ (add an edge from (i, o_j) to (i, o_{j+1}) with capacity $c_i(o_j)$);
    end
    $E \leftarrow E \cup \{((i, o_m), t, c_i(o_m))\}$ (add an edge from (i, o_m) to t with capacity $c_i(o_m)$);
end
for *each pair of types* (i_1, i_2) *where* $i_1 \neq i_2$ *and* $(i_1, i_2) \in R$*, and each outcome* $o_j \in \mathcal{O}$ **do**
    $E \leftarrow E \cup \{((i_2, o_j), (i_1, o_j), \infty)\}$ (add an edge from (i_2, o_j) to (i_1, o_j) with capacity ∞);
end
Compute an s-t min-cut $(S, \overline{S})$ on graph $G = (V, E)$;
for *each type* $i \in \Theta$ **do**
    Let $M(i) = o_j$ where $j = \max\{j' \in [m] \mid (i, o_{j'}) \in S\}$;
end
return M;

linear extension c_i^ℓ of c_i is convex.

Lemma 1 (Optimality of Deterministic Mechanisms with Convex Costs)**.** *When all types share the same utility function, and the principal's cost function is convex, there is an optimal truthful mechanism that is deterministic even with partial verification allowed.*

2.5 Reducing General Costs to Convex Costs

Lemma 1 together with Algorithm 1 provides an efficient way for finding optimal truthful mechanisms with convex costs (even when randomized mechanisms are allowed). One may still wonder if it is possible to design faster algorithms *in general* than solving the standard LP formulation, presumably by exploiting the additional structure that the agents share the same utility function. To this end, we observe that for computing optimal mechanisms, only the convex envelope of the principal's cost function matters. Given this observation, we show that finding optimal truthful mechanisms can be reduced very efficiently to finding optimal deterministic mechanisms.

We present Algorithm 2, which computes the optimal truthful mechanism and has the same asymptotic runtime as Algorithm 1. Algorithm 2 first computes the convex envelope of the principal's cost function, and then finds an optimal "deterministic" mechanism by calling Algorithm 1 with the same types and outcomes, but replacing the principal's cost function with its convex envelope. Algorithm 2 then recovers an optimal randomized mechanism from the "deterministic" one, by interpreting each "deterministic" outcome as a convex combination of outcomes in an optimal way. The following theorem establishes the correctness and time complexity of Algorithm 2.

Theorem 4. *Algorithm 2 finds an optimal (possibly randomized) truthful mechanism, in asymptotically the same time as Algorithm 1.*

Below we give a comparison between the time complexity of our algorithm, Algorithm 2, and that of the LP-based approach.[5] The current best algorithm for LP (Cohen, Lee, and Song 2019) takes time that translates to $\widetilde{O}(n^{2.37}m^{2.37} + n^{4.74})$[6] in our setting (this is, for example, at least $\widetilde{O}(n^{3.24}m^{1.5})$). The current best algorithm for s-t min-cut (Lee and Sidford 2014) takes time that translates to $\widetilde{O}(n^{2.5}m^{1.5})$ in our setting. Moreover, in a typical classification setting, it is the number of outcomes (corresponding to "accept", etc.) m that is small, and the number of types (e.g., "(CS major, highly competitive, female, international, ...)", "(math major, acceptable, male, domestic, ...)") n is much larger. In such cases, the improvement becomes even more significant. Our results are theoretical, but practically, while there are highly optimized packages for LP, there are also highly optimized packages for max-flow / min-cut that are still much faster. Last but not least, in many practical settings, the principal has to implement a deterministic policy (it is hard to imagine college admissions explicitly made random), in which case our Algorithm 1 can be applied while LP generally does not give a solution.

[5] We note that a conclusive comparison is unrealistic since algorithms for both LP and min-cut keep being improved.

[6] $\widetilde{O}$ hides a poly-logarithmic factor.

Algorithm 2: Finding an optimal (possibly randomized) truthful mechanism.

Input: The set of types Θ, the principal's cost function $\{c_i\}_{i\in\Theta}$ for each type, the set of outcomes $\mathcal{O}$ (which encodes the common utility function), and the reporting structure R.

Output: A truthful mechanism $M : \Theta \to \mathcal{O}$ minimizing the principal's cost.

for *each type* i **do**

  Compute the convex envelope $c_i^- : [o_1, o_m] \to \mathbb{R}_+$ of c_i, defined such that for any $x \in [o_1, o_m]$,

$$c_i^-(x) = \min_{o\in\Delta(\mathcal{O}),\mathbb{E}[o]=x} \mathbb{E}[c_i(o)].$$

  Let $\widehat{c_i}$ be c_i^- restricted to $\mathcal{O}$;

end

Run Algorithm 1 on input $(\Theta, \{\widehat{c_i}\}_{i\in\Theta}, \mathcal{O}, R)$. Let $\widehat{M}$ be the resulting deterministic mechanism;

for *each type* i **do**

  $M(i) \leftarrow \operatorname{argmin}_{o\in\Delta(\mathcal{O}),\mathbb{E}[o]=\widehat{M}(i)} \mathbb{E}[c_i(o)]$;

end

return M;

3 Generalizing to Combinatorial Costs

In this section, we generalize the problem considered in the previous section, allowing the principal to have a combinatorial cost function over outcomes for each type. See Appendix A for a more detailed exposition.

The combinatorial setting. As before, let $\Theta = [n]$ be the set of types, $\mathcal{O} = \{o_j\}_{j\in[m]} \subseteq \mathbb{R}_+$ be the set of outcomes encoding the common utility function, and $R \subseteq \Theta\times\Theta$ be the reporting structure. The principal's cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ now maps a vector $O = (O^i)_i$ of outcomes for all types to the principal's cost $c(O)$. This subsumes the additive case, since one can set the cost function c to be

$$c((O^i)_i) = \sum_{i\in\Theta} c_i(O^i).$$

Because the cost function is now combinatorial, it matters how the mechanism combines outcomes for different types. We therefore modify the definition of a randomized mechanism $M \in \Delta(\Theta \to \mathcal{O}) = \Delta(\mathcal{O}^\Theta)$, so that it allows correlation across different types. The principal's cost from using a truthful mechanism M is then $c(M) = \mathbb{E}[c((M(i))_i)]$. For type i, the utility from executing mechanism M is still $u_i(M) = \mathbb{E}[M(i)]$. M is truthful iff for any $(i_1, i_2) \in R$, $u_{i_1}(M) \geq u_{i_2}(M)$. In the rest of the section, we present combinatorial generalizations of all our algorithmic and structural results given in the previous section.

General vs. submodular cost functions. Combinatorial functions in general are notoriously hard to optimize, even ignoring incentive issues. To see the difficulty, observe that a combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ over $\mathcal{O}^\Theta$ generally does not even admit a succinct representation (e.g., one whose size is polynomial in m and n). It is therefore infeasible to take the entire cost function as input to an algorithm. To address this issue, the standard assumption in combinatorial optimization is that algorithms can access the combinatorial function through *value queries*. That is, we are given an oracle that can evaluate the combinatorial function c at any point $O \in \mathcal{O}^\Theta$, obtaining the value $c(O)$ in constant time. For the rest of the paper, we assume that our algorithm can access the cost function only through value queries.

Still, in order to minimize an *arbitrary* combinatorial function, in general one needs $\Omega(m^n)$ queries to obtain any nontrivial approximation. Despite that, there exist efficient algorithms for combinatorial minimization for an important subclass of cost functions, namely submodular functions.

Definition 2 (Submodular Functions)**.** For any $O_1 = (O_1^i)_i \in \mathcal{O}^\Theta$ and $O_2 = (O_2^i)_i \in \mathcal{O}^\Theta$, let $O_1 \wedge O_2 = (\min(O_1^i, O_2^i))_i$ and $O_1 \vee O_2 = (\max(O_1^i, O_2^i))_i$. A combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ is submodular if for any $O_1, O_2 \in \mathcal{O}^\Theta$,

$$c(O_1) + c(O_2) \geq c(O_1 \wedge O_2) + c(O_1 \vee O_2).$$

In the rest of this section, we focus on submodular cost functions. For this important special case, we give efficient algorithms for finding optimal truthful deterministic / randomized mechanisms, as well as a sufficient condition for the existence of an optimal mechanism that is deterministic.

Finding optimal deterministic mechanisms. First we present a polynomial-time combinatorial algorithm for finding optimal truthful deterministic mechanisms with partial verification, when the cost function is submodular.

Theorem 5. *There exists a polynomial-time algorithm which accesses the cost function via value queries only, and computes an optimal deterministic truthful mechanism when partial verification is allowed and the cost function is submodular.*

Sufficient condition for the optimality of deterministic mechanisms. Restricted to additive cost functions, Lemma 1 gives a sufficient condition under which there exists an optimal mechanism that is deterministic. We present below a combinatorial version of this structural result when the outcome space is binary, i.e., when $m = 2$.

Theorem 6 (Optimality of Deterministic Mechanisms with Binary Outcomes)**.** *When the outcome space is binary, i.e., $|\mathcal{O}| = 2$, and the principal's cost function is submodular, there is an optimal truthful mechanism that is deterministic, even when partial verification is allowed.*

Computing optimal randomized mechanisms. Finally we give an algorithm for finding an optimal mechanism with arbitrary submodular cost functions.

Theorem 7. *When the cost function c is submodular and bounded, for any desired additive error $\varepsilon > 0$, there is an algorithm which finds an ε-approximately optimal (possibly randomized) truthful mechanism[7] in time $\mathrm{poly}(n, m, \log(1/\varepsilon))$, even if partial verification is allowed.*

[7] An ε-approximately optimal truthful mechanism is a truthful mechanism whose expected cost is at most ε larger than the minimum possible cost of any truthful mechanism.

Ethics Statement

Our results can be used to encourage truthful reporting, and thereby improve the efficiency of mechanisms for, e.g., allocating public resources. In particular, this helps the principal to allocate resources to agents in actual need. By presenting more efficient algorithms, we make large-scale applications of automated mechanism design possible, which can be applied to problems related to social good that were previously beyond reach. Of course, our algorithms, as well as any other algorithm, could potentially cause harm if implemented in an irresponsible way (e.g., by using a cost function that discriminates between applicants in an unfair way).

Acknowledgements

Part of this work was done while Yu Cheng was visiting the Institute of Advanced Study. Hanrui Zhang and Vincent Conitzer are thankful for support from NSF under award IIS-1814056.

References

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

Balcan, M.-F.; Sandholm, T.; and Vitercik, E. 2018. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, 173–174.

Balcan, M.-F.; Sandholm, T.; and Vitercik, E. 2019. Estimating Approximate Incentive Compatibility. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, 867–867.

Chen, Y.; Podimata, C.; Procaccia, A. D.; and Shah, N. 2018. Strategyproof linear regression in high dimensions. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, 9–26.

Cohen, M. B.; Lee, Y. T.; and Song, Z. 2019. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, 938–942.

Cole, R.; and Roughgarden, T. 2014. The sample complexity of revenue maximization. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 243–252.

Conitzer, V.; and Sandholm, T. 2002. Complexity of mechanism design. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, 103–110.

Conitzer, V.; and Sandholm, T. 2004. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, 132–141.

Dekel, O.; Fischer, F.; and Procaccia, A. D. 2010. Incentive compatible regression learning. *Journal of Computer and System Sciences* 76(8): 759–777.

Devanur, N. R.; Huang, Z.; and Psomas, C.-A. 2016. The sample complexity of auctions with side information. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 426–439.

Duetting, P.; Feng, Z.; Narasimhan, H.; Parkes, D.; and Ravindranath, S. S. 2019. Optimal Auctions through Deep Learning. In *International Conference on Machine Learning*, 1706–1715.

Gonczarowski, Y. A.; and Weinberg, S. M. 2018. The Sample Complexity of Up-to-ε Multi-Dimensional Revenue Maximization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 416–426. IEEE.

Green, J. R.; and Laffont, J.-J. 1986. Partially verifiable information and mechanism design. *The Review of Economic Studies* 53(3): 447–456.

Haghtalab, N.; Immorlica, N.; Lucier, B.; and Wang, J. 2020. Maximizing Welfare with Incentive-Aware Evaluation Mechanisms.

Hardt, M.; Megiddo, N.; Papadimitriou, C.; and Wootters, M. 2016. Strategic classification. In *Proceedings of the 2016 ACM conference on innovations in theoretical computer science*, 111–122.

Kephart, A.; and Conitzer, V. 2015. Complexity of mechanism design with signaling costs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 357–365.

Kephart, A.; and Conitzer, V. 2016. The revelation principle for mechanism design with reporting costs. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, 85–102.

Kleinberg, J.; and Raghavan, M. 2019. How Do Classifiers Induce Agents to Invest Effort Strategically? In *Proceedings of the 2019 ACM Conference on Economics and Computation*, 825–844.

Krishnaswamy, A.; Li, H.; Rein, D.; Zhang, H.; and Conitzer, V. 2021. Classification with Strategically Withheld Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Laffont, J.-J.; and Tirole, J. 1986. Using cost observation to regulate firms. *Journal of political Economy* 94(3, Part 1): 614–641.

Lee, Y. T.; and Sidford, A. 2014. Path finding methods for linear programming: Solving linear programs in $\widetilde{O}(\sqrt{\mathrm{rank}})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, 424–433. IEEE.

McAfee, R. P.; and McMillan, J. 1987. Competition for agency contracts. *The RAND Journal of Economics* 296–307.

Perote, J.; and Perote-Pena, J. 2004. Strategy-proof estimators for simple regression. *Mathematical Social Sciences* 47(2): 153–176.

Shen, W.; Tang, P.; and Zuo, S. 2019. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, 215–223.

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

Zhang, H.; Cheng, Y.; and Conitzer, V. 2019a. Distinguishing Distributions When Samples Are Strategically Transformed. In *Advances in Neural Information Processing Systems*, 3187–3195.

Zhang, H.; Cheng, Y.; and Conitzer, V. 2019b. When samples are strategically selected. In *International Conference on Machine Learning*, 7345–7353.

Zhang, H.; Cheng, Y.; and Conitzer, V. 2021. Classification with Few Tests through Self-Selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, H.; and Conitzer, V. 2021. Incentive-Aware PAC Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.