Automated Mechanism Design for Classification with Partial Verification

Hanrui Zhang
Duke University
hrzhang@cs.duke.edu

Yu Cheng
University of Illinois at Chicago
yucheng2@uic.edu

Vincent Conitzer
Duke University
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* [14, 26], 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. (Here, the arrival time is the type.) Here, 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 et al. [28] 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 manipulate these samples before they are submitted to the principal, by either withholding or transforming some of them in limited ways [28, 27]. 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* [8, 9], 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 [2, 26, 18, 19]. 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. (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.) 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 [2, 26, 18, 19], paint 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 [8]. 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 [8, 9]). 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 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 [7, 11, 3, 13], mechanism design via deep learning [12, 25], and estimating incentive compatibility [4]. 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).

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 [17, 20, 16] (see also [18, 19]). Strategic aspects of linear regression have also been studied [23, 10, 6]. 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 [21, 22]. These results are incomparable with ours, since they consider rather specific models, while we consider utility and cost functions of essentially any form.

2 The Traditional Setting: 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.[1] However, this is not true in the case of partial verification (see, e.g., [14, 26, 19]). 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

[1] 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.

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 [8]. 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 all types have single-peaked utility functions.*

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.

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. 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. See Appendix B.1 for an example of the graph constructed in 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'*.*

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 algorithm 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

Algorithm 1: Finding an optimal deterministic mechanism.

```
Input: The set of types $\Theta$, 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 $\infty$) ;
    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 $\infty$);
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 $M$;
```

$c_2(o_2) = \infty$. 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 exactly 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 $x \in [o_1, o_m] \setminus \mathcal{O}$,

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

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

```
Input: The set of types $\Theta$, 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
    $M(i) \leftarrow \mathrm{argmin}_{o \in \Delta(\mathcal{O}), \mathbb{E}[o]=\widehat{M}(i)} \mathbb{E}[c_i(o)]$ ;
end
return $M$;
```

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 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.*

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. The problem studied in the previous section can be viewed as a special case (where the principal's cost function is additive over types) of this general problem. 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 $c(M) = \mathbb{E}[c((M(i))_i)]$. We treat M as a distribution or a random variable over $\mathcal{O}^\Theta$ interchangeably. Note that each type's utility is still independent of what other types get. So for type i, the utility from executing mechanism M is still $u_i(M) = \mathbb{E}[M(i)]$. And 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

$$O_1 \wedge O_2 = (\min(O_1^i, O_2^i))_i \quad \text{and} \quad O_1 \vee O_2 = (\max(O_1^i, O_2^i))_i.$$

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. We have shown in Example 1 that the gap between deterministic and randomized mechanisms can be arbitrarily large. 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.*

An efficient algorithm for computing optimal randomized mechanisms. Finally we show there exists 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[2] in time $\mathrm{poly}(n, m, \log(1/\varepsilon))$, even if partial verification is allowed.*

[2] 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.

Broader Impact

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., with a discriminative cost function).

References

[1] AM Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[2] Vincenzo Auletta, Paolo Penna, Giuseppe Persiano, and Carmine Ventre. Alternatives to truthfulness are hard to recognize. *Autonomous Agents and Multi-Agent Systems*, 22(1):200–216, 2011.

[3] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 173–174, 2018.

[4] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Estimating approximate incentive compatibility. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 867–867, 2019.

[5] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[6] Yiling Chen, Chara Podimata, Ariel D Procaccia, and Nisarg Shah. Strategyproof linear regression in high dimensions. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 9–26, 2018.

[7] Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 243–252, 2014.

[8] Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 103–110, 2002.

[9] Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 132–141, 2004.

[10] Ofer Dekel, Felix Fischer, and Ariel D Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76(8):759–777, 2010.

[11] Nikhil R Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. The sample complexity of auctions with side information. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 426–439, 2016.

[12] Paul Duetting, Zhe Feng, Harikrishna Narasimhan, David Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In *International Conference on Machine Learning*, pages 1706–1715, 2019.

[13] Yannai A Gonczarowski and S Matthew Weinberg. The sample complexity of up-to-ε multi-dimensional revenue maximization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 416–426. IEEE, 2018.

[14] Jerry R Green and Jean-Jacques Laffont. Partially verifiable information and mechanism design. *The Review of Economic Studies*, 53(3):447–456, 1986.

[15] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric methods in combinatorial optimization. In *Progress in combinatorial optimization*, pages 167–183. Elsevier, 1984.

[16] Nika Haghtalab, Nicole Immorlica, Brendan Lucier, and Jack Wang. Maximizing welfare with incentive-aware evaluation mechanisms, 2020.

[17] Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM conference on innovations in theoretical computer science*, pages 111–122, 2016.

[18] Andrew Kephart and Vincent Conitzer. Complexity of mechanism design with signaling costs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 357–365, 2015.

[19] Andrew Kephart and Vincent Conitzer. The revelation principle for mechanism design with reporting costs. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 85–102, 2016.

[20] Jon Kleinberg and Manish Raghavan. How do classifiers induce agents to invest effort strategically? In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 825–844, 2019.

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

[22] R Preston McAfee and John McMillan. Competition for agency contracts. *The RAND Journal of Economics*, pages 296–307, 1987.

[23] Javier Perote and Juan Perote-Pena. Strategy-proof estimators for simple regression. *Mathematical Social Sciences*, 47(2):153–176, 2004.

[24] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[25] Weiran Shen, Pingzhong Tang, and Song Zuo. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 215–223. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

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

[27] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Distinguishing distributions when samples are strategically transformed. In *Advances in Neural Information Processing Systems*, pages 3187–3195, 2019.

[28] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. When samples are strategically selected. In *International Conference on Machine Learning*, pages 7345–7353, 2019.

A 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. The problem studied in the previous section can be viewed as a special case (where the principal's cost function is additive over types) of this general problem. Before we proceed to the formal definition of the problem, to better motivate combinatorial cost functions, consider the following example.

Example 2. Suppose in addition to an additive cost function $\{c_i\}_{i\in\Theta}$, the principal has to pay an overhead cost $c_0 > 0$ as long as any type receives a nontrivial outcome, i.e., if there exists $i \in \Theta$, such that $M(i) \in \mathcal{O} \setminus \{o_1\}$. In such cases, the principal's overall cost from executing a truthful deterministic mechanism M can be written as

$$c(M) = \sum_{i\in\Theta} c_i(M(i)) + c_0 \cdot \mathbb{I}[\exists i \in \Theta : M(i) \neq o_1],$$

where $\mathbb{I}[\cdot]$ is the indicator of a statement.

In the above rather natural example, the principal's cost is no longer additive over types. As a result, there is no way to properly formulate Example 2 using our previous definitions. We generalize the principal's cost function, as well as the definition of mechanisms, as follows.

Notation. 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 (possibly randomized) mechanism $M \in \Delta(\Theta \to \mathcal{O}) = \Delta(\mathcal{O}^\Theta)$, such that it allows correlation across different types. The principal's cost from executing a truthful mechanism M is then

$$c(M) = \mathbb{E}[c((M(i))_i)].$$

We treat M as a distribution or a random variable over $\mathcal{O}^\Theta$ interchangeably. Note that each type's utility is still independent of what other types get. So for type i, the utility from executing mechanism M is still

$$u_i(M) = \mathbb{E}[M(i)].$$

And 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.

A.1 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 only access the cost function only through value queries.

Now suppose we are to design an algorithm to minimize an arbitrary combinatorial cost function, without any additional constraint. That is, given a combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$, we wish to find a point $O \in \mathcal{O}^\Theta$, such that $c(O)$ is minimized over $\mathcal{O}^\Theta$. The example below shows that any algorithm which interacts with c only through value queries needs $\Omega(m^n)$ queries to obtain any nontrivial approximation to the above seemingly basic problem.

Example 3. Let the cost function c be generated in the following random way. A point O^* is drawn from $\mathcal{O}^\Theta$ uniformly at random. c is then constructed such that $c(O^*) = 0$, and $c(O) = 1$ for any $O \neq O^*$. To minimize c, the algorithm has to find O^*. This is equivalent to guessing a uniformly random number among m^n numbers. To guess O^* successfully with constant probability, one has to make $\Omega(m^n)$ guesses.

The above issue has been identified in combinatorial optimization since decades ago. Despite the fact that general combinatorial cost functions are hard to minimize, researchers have developed efficient algorithms for combinatorial minimization for an important subclass of cost functions, namely submodular functions, defined below.

Definition 3 (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

$$O_1 \wedge O_2 = (\min(O_1^i, O_2^i))_i \quad \text{and} \quad O_1 \vee O_2 = (\max(O_1^i, O_2^i))_i.$$

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 the 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.

A.2 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. The algorithm is based on the key observation that the space of truthful deterministic mechanisms is a distributive lattice (defined below in Lemma 2). Given this observation, it is known that the problem can be reduced to minimizing a submodular function without additional constraints, which can be solved efficiently.

Lemma 2. *Fix the set of types Θ, the set of outcomes $\mathcal{O}$, and the reporting structure R. Let $\mathcal{T} \subseteq \mathcal{O}^\Theta$ be the space of all possible ways of assigning outcomes to types, such that no type has the incentive to misreport. That is,*

$$\mathcal{T} = \{O = (O^i)_i \in \mathcal{O}^\Theta \mid \forall (i_1, i_2) \in R,\, O^{i_1} \geq O^{i_2}\}.$$

Then $\mathcal{T}$ is a distributive lattice, i.e., $\mathcal{T}$ satisfies the following conditions.

- *For any $O_1, O_2 \in \mathcal{T}$, $O_1 \wedge O_2 \in \mathcal{T}$, and $O_1 \vee O_2 \in \mathcal{T}$.*
- *For any $O_1, O_2, O_3 \in \mathcal{T}$, $O_1 \vee (O_2 \wedge O_3) = (O_1 \vee O_2) \wedge (O_1 \vee O_3)$.*

Proof. Consider the first property. Fix any $O_1, O_2 \in \mathcal{T}$, and let $O_- = O_1 \wedge O_2$, and $O_+ = O_1 \vee O_2$. For any $(i_1, i_2) \in R$, since $O_1^{i_1} \geq O_1^{i_2}$ and $O_2^{i_1} \geq O_2^{i_2}$, we have

$$O_-^{i_1} = \min(O_1^{i_1}, O_2^{i_1}) \geq \min(O_1^{i_2}, O_2^{i_2}) = O_-^{i_2}.$$

This implies $O_- \in \mathcal{T}$. Similarly we may show $O_+ \in \mathcal{T}$. In other words, the first property holds.

For the second property, simply consider the i-th coordinate for any $i \in \Theta$. Fix $O_1, O_2, O_3 \in \mathcal{T}$, we have

$$\max(O_1^i, \min(O_2^i, O_3^i)) = \min(\max(O_1^i, O_2^i), \max(O_1^i, O_3^i)).$$

Since this is true for any i, the second property follows immediately. □

Given Lemma 2, we can apply the algorithm and the reduction by Schrijver [24] to obtain an efficient algorithm directly.

Corollary 1. *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.*

Proof. Let $\mathcal{T}$ be the family of truthful assignments defined in Lemma 2. The problem of finding an optimal deterministic truthful mechanism can be equivalently formulated as the following optimization problem.

$$\min_{O \in \mathcal{T}} c(O).$$

This can be solved by applying the reduction in Section 6 of [24][3] to any algorithm for minimizing submodular functions (e.g., the one given in [24]). □

We remark that the algorithm by Schrijver [24] can be applied as well in the classical additive setting, but due to its generality, is significantly less efficient than Algorithm 1.

A.3 Sufficient Condition for the Optimality of Deterministic Mechanisms

We have shown in Example 1 that the gap between deterministic and randomized mechanisms can be arbitrarily large. Restricted to additive cost functions, Lemma 1 gives a sufficient condition under which there exists an optimal mechanism that is deterministic. We present in this subsection a combinatorial version of this structural result when the outcome space is binary, i.e., when $m = 2$.

Lemma 3 (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.*

Proof. The overall plan is similar to that of the proof of Lemma 1. We begin with a (possibly randomized) optimal truthful mechanism M, and show that without loss of generality, we may assume its support has some monotone structure. We then round this mechanism, such that the resulting deterministic mechanism is always truthful, and the expected cost of the rounded mechanism is equal to the cost of M.

Without loss of generality, suppose $\mathcal{O} = \{0, 1\}$. Observe that $\mathcal{O}^\Theta$ is isomorphic to 2^Θ, so in the rest of the proof, we interchangeably represent an outcome vector O as a subset of Θ, i.e., the set

$$\{i \in \Theta \mid O^i = 1\}.$$

Let M be any (possibly randomized) optimal truthful mechanism. Below we treat M as a random variable distributed over $\mathcal{O}^\Theta$, or interchangeably, a random subset of Θ. For any $O \in \mathcal{O}^\Theta$, let $p(O) = \Pr[M = O]$ be the probability that M assigns outcomes O to types. We further require M to maximize the following potential function among all optimal truthful mechanisms.

$$\mathbb{E}[|M|^2] = \sum_{O \in \mathcal{O}^\Theta} p(O) \cdot |O|^2.$$

We argue below that for such an M, no two outcome vectors in the support of M "cross."

We say two outcome vectors O_1 and O_2 (represented as sets) cross, if $O_1 \not\subseteq O_2$ and $O_2 \not\subseteq O_1$. Toward a contradiction, suppose O_1 and O_2 cross, where without loss of generality $p(O_1) \geq p(O_2) > 0$. Let $p = p(O_2)$. We show that moving probability mass from O_1 and O_2 to $O_1 \cap O_2$ and $O_1 \cup O_2$ simultaneously preserves truthfulness, does not increase the cost, and strictly increases the potential of M, which contradicts the choice of M.

To be precise, we decrease $p(O_1)$ and $p(O_2)$ simultaneously by p, and increase $p(O_1 \cap O_2)$ and $p(O_1 \cup O_2)$ simultaneously by p. To see why truthfulness is preserved, observe that the expected utility of any type does not change after the modification. The change of the cost can be written as

$$\begin{aligned} \Delta c(M) &= p \cdot (c(O_1 \cap O_2) + c(O_1 \cup O_2) - c(O_1) - c(O_2)) \\ &\leq p \cdot (c(O_1) + c(O_2) - c(O_1) - c(O_2)) && \text{(submodularity of } c\text{)} \\ &= 0, \end{aligned}$$

so the cost does not increase. Finally, the change of the potential is

$$p \cdot (|O_1 \cap O_2|^2 + |O_1 \cup O_2|^2 - |O_1|^2 - |O_2|^2).$$

[3] Although the reduction presented therein is for ring families, one may check it also works for distributive lattices.

It is easy to check the above is strictly positive as long as O_1 and O_2 cross.

From now on we assume no two outcomes in the support of M cross, or equivalently, the support of M is a family of nested subsets of Θ. For any $r \in [0,1]$, let $M_r \subseteq \Theta$ be such that

$$M_r = \{i \in \Theta \mid u_i \geq r\}.$$

Observe that M_r is truthful for any $r \in [0,1]$. In fact, for any $(i_1, i_2) \in R$, we always have

$$i_2 \in M_r \implies i_1 \in R.$$

Truthfulness then follows.

Consider the random (but not randomized) mechanism M_r when r is uniformly distributed over $[0,1]$. We argue below that the expected cost of M_r is precisely that of M. In fact, M_r and M are even identically distributed. For any $i \in \Theta$, let the expected utility of type i be $u_i = \Pr[i \in M]$. Without loss of generality, suppose M satisfies $u_i \geq u_{i+1}$ for any $i \in [n-1]$. To show M_r and M are identically distributed, we only need to show that

$$p(O) > 0 \implies \exists i \in \Theta, O = [i].$$

In other words, M only assigns outcomes that are prefixes of Θ. Given this, M_r is the only distribution that gives each type i expected utility u_i simultaneously.

To see why the above claim is true, suppose there exists some $O \subseteq \Theta$ where $1 \notin O$ and $p(O) > 0$. Then since the support of M is a nested family of subsets of Θ, for any $i \in O$, regardless of the realization of M, we always have

$$1 \in M \implies i \in O.$$

As a result,

$$u_i = \sum_{O' \subseteq \Theta : i \in O'} p(O') \geq p(O) + \sum_{O' \subseteq \Theta : 1 \in O'} p(O') = p(O) + u_1 > u_1,$$

a contradiction. This concludes the proof. □

We make the following remarks regarding Lemma 3.

- The binary outcomes assumption, despite being more restrictive than the general model, still captures many real-life applications. In particular, it models binary classification problems where one label is more desirable than the other for all agents. Common examples include hiring decisions, university admissions, etc. Moreover, such decisions are generally correlated over types (e.g., universities cannot admit too many students) — this is captured by the principal's submodular cost function.
- The proof of Lemma 3 can be alternatively interpreted in the following way. Without loss of generality, any optimal mechanism corresponds to a point on the convex envelope of the principal's cost function. And for submodular cost functions particularly, this convex envelope happens to coincide with the Lovász extension (see [15]), which can be derandomized into deterministic mechanisms preserving truthfulness. We will further develop this intuition in the next result, which is an efficient algorithm for finding optimal truthful mechanisms for any submodular function.

A.4 An Efficient Algorithm for Computing Optimal Randomized Mechanisms

In this subsection, we present an algorithm for finding an optimal mechanism with arbitrary submodular cost functions.

Our algorithm, Algorithm 3, again builds on the intuition that for optimal mechanisms, only the convex envelope of the cost function matters. The problem of finding optimal mechanisms can therefore be formulated as a convex program. However, unlike in Algorithm 2, with submodular cost functions, it is not clear how one can efficiently evaluate the convex envelope of the cost function.

To get around this issue, instead of parametrizing by the target utilities, we parametrize the convex envelope by the marginal probabilities $\{p_{i,j}\}_{i \in \Theta, j \in [m]}$, where $p_{i,j}$ is the probability that type i gets outcome o_j. One of the key ingredients of Algorithm 3 is a subroutine (Algorithm 4) which efficiently interprets each point on the convex envelope as a convex combination of integral points, corresponding to a distribution over combinations of outcomes. In other words, given the desired marginal probabilities, Algorithm 4 finds a randomized truthful mechanism realizing these marginal probabilities which minimizes the principal's expected cost.

Algorithm 3: Finding an optimal mechanism with submodular cost functions.

```
Input: The set of types $\Theta$, the principal's submodular cost function $c$, the set of outcomes $O$
       (which encodes the common utility function), the reporting structure $R$, and a precision
       parameter $\varepsilon$.
Output: An optimal truthful mechanism $M \in \Delta(\Theta \to \mathcal{O})$.
Compute an $\varepsilon$-approximately optimal solution $p^*_{i,j}$ to the following convex program using the
  ellipsoid method (see, e.g., [5]), and call Algorithm 4 with parameters $\{p_{i,j}\}$ for evaluating
  $\mathrm{prob}(\cdot \mid \{p_{i,j}\})$
      $$\begin{aligned} \min \quad & \sum_{O \in \mathcal{O}^\Theta} \mathrm{prob}(O \mid \{p_{i,j}\}_{i \in \Theta, j \in [m]}) \cdot c(O) & \\ \text{s.t.} \quad & \sum_{j \in [m]} p_{i_1,j} \cdot o_j \ge \sum_{j \in [m]} p_{i_2,j} \cdot o_j & \forall (i_1, i_2) \in R \\ & \sum_{j \in [m]} p_{i,j} = 1 & \forall i \in \Theta \\ & p_{i,j} \ge 0 & \forall i \in \Theta, j \in [m]; \end{aligned}$$
for each $O \in \mathcal{O}^\Theta$ do
    $\Pr[M = O] \leftarrow \mathrm{prob}(O \mid \{p^*_{i,j}\})$;
end
return $M$;
```

Algorithm 4: Algorithm for interpreting the convex envelope.

```
Input: Marginal probabilities $\{p_{i,j}\}_{i,j}$, where $p_{i,j}$ is the desired probability that type $i$ gets
       outcome $o_j$.
Output: A distribution $\{\mathrm{prob}(O \mid \{p_{i,j}\})\}_{O \in \mathcal{O}^\Theta}$ over combinations of outcomes.
Let $\mathrm{prob}(O) \leftarrow 0$ for all $O \in \mathcal{O}^\Theta$;
while there is some $p_{i,j} > 0$ do
    for each $i \in \Theta$ do
        $t_i \leftarrow \max\{j \in [m] \mid p_{i,j} > 0\}$;
    end
    $\delta \leftarrow \min_i p_{i,t_i}$;    $\mathrm{prob}((t_i)_{i \in [n]}) \leftarrow \delta$;
    for each $i \in \Theta$ do
        $p_{i,t_i} \leftarrow p_{i,t_i} - \delta$;
    end
    return $\{\mathrm{prob}(O)\}_{O \in \mathcal{O}^\Theta}$;
end
```

Theorem 8. *When the cost function c is submodular and bounded, for any desired additive error $\varepsilon > 0$, Algorithm 3 finds an ε-approximately optimal (possibly randomized) truthful mechanism*[4] *in time* $\mathrm{poly}(n, m, \log(1/\varepsilon))$*, even if partial verification is allowed.*

Proof. Consider the program in Algorithm 3. Observe there are mn variables, namely $\{p_{i,j}\}$, and $O(n^2 + mn)$ linear constraints in the program. In order for the program to be efficiently solvable, we only need to show the following three claims hold.

- The subroutine for evaluating the convex envelope, Algorithm 4, runs in polynomial time.
- The objective is convex in $\{p_{i,j}\}$.

[4] 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.

- A first-order oracle, which computes a (sub)gradient of the objective function at any point, can be efficiently implemented. (The ellipsoid method also requires an efficient separation oracle, which for our program exists straightforwardly, since there are only $\mathrm{poly}(n, m)$ linear constraints.)

The second claim and the third claim also imply the correctness of the Algorithm. Below we prove the three claims.

Consider the first claim. We only need to show that the while-loop at line 2 repeats only polynomially many times. Consider the following potential function ϕ.

$$\phi(\{p_{i,j}\}) = \sum_{i \in \Theta} \max\{j \in [m] \mid p_{i,j} > 0\}.$$

Before line 2, the value of ϕ is at most mn. Observe that ϕ is monotone in $\{p_{i,j}\}$, and the latter never increase during the execution of the loop. Moreover, in each repetition of the loop, ϕ decreases at least by 1. This is because after the update in line 9, for some $i \in \Theta$, p_{i,t_i} becomes 0, and as a result, $\max\{j \in [m] \mid p_{i,j} > 0\}$ decreases at least by 1. When ϕ becomes 0, it must be the case that $p_{i,j} = 0$ for any $i \in \Theta, j \in [m]$, so the loop terminates. Therefore the while-loop repeats at most mn times, which implies the first claim.

Now consider the second claim. We show that in Algorithm 4, the output distribution $\{p(O)\}$ minimizes the expected cost

$$\sum_{O \in \mathcal{O}^\Theta} p(O) \cdot c(O)$$

among all distributions whose marginals are $\{p_{i,j}\}$ — this is equivalent to the second claim. We first prove the following characterization of the output distribution $\{p(O)\}$.

Lemma 4. *The output distribution $\{p(O)\}$ of Algorithm 4 is the only distribution over O^Θ satisfying the following properties.*

- *$\{p(O)\}$ induce the input marginal probabilities $\{p_{i,j}\}$ over type-outcome pairs.*
- *For any $O_1, O_2 \in \mathcal{O}^\Theta$, if $O_1 \wedge O_2 \notin \{O_1, O_2\}$, then either $p(O_1) = 0$ or $p(O_2) = 0$. In other words, no two combinations of outcomes in the support of $\{p(O)\}$ cross.*

Proof. The first bullet point is clear from the construction of $\{p(O)\}$. We therefore focus on the second bullet point. We first show that for any marginal probabilities $\{p_{i,j}\}$ and distribution $\{p'(O)\}$, if (1) $\{p'(O)\}$ induce $\{p_{i,j}\}$ and no two combinations of outcomes in the support of $\{p'(O)\}$ cross, then the topmost combination of outcomes O_t, where

$$O_t^i = \max\{o_j \mid j \in [m], p_{i,j} > 0\},$$

must have probability exactly

$$p'(O_t) = \min_{i \in \Theta} p_{i,t_i},$$

where $t_i = \max\{j \in [m] \mid p_{i,j} > 0\}$.

Let $\delta = \min_{i \in \Theta} p_{i,t_i}$. Observe that $p'(O_t) \le \delta$. Suppose toward a contradiction that $p'(O_t) < \delta$. Since no two combinations of outcomes in the support of $\{p'(O)\}$ cross, we can order the support of $\{p'(O)\}$ as $O_1, \dots, O_\ell$, where ℓ is the size of the support, such that for any $i \in [\ell - 1]$,

$$O_i \wedge O_{i+1} = O_{i+1}.$$

Clearly we have $O_t \wedge O_1 = O_1$. Consider the following two cases.

- $O_t \ne O_1$. In other words, there is some $i^* \in \Theta$, such that $O_t^{i^*} > O_1^{i^*}$. As a result, for any $k \in [\ell]$,

  $$O_k^{i^*} \le O_1^{i^*} < O_t^{i^*} \le o_{t_{i^*}}.$$

  Then we have

  $$0 < \delta \le p_{i^*,t_{i^*}} = \sum_{k \in [\ell]} p'(O_k) \cdot \mathbb{I}[O_k^{i^*} = o_{t_{i^*}}] = 0,$$

  a contradiction.

- $O_t = O_1$. There is some $i^* \in \Theta$, such that $O_1^{i^*} > O_2^{i^*}$. As a result, for any $2 \leq k \leq \ell$,
$$O_k^{i^*} \leq O_2^{i^*} < O_1^{i^*} = o_{t_{i^*}}.$$
So we have
$$\delta \leq p_{i^*,t_{i^*}} = \sum_{k \in [\ell]} p'(O_k) \cdot \mathbb{I}[O_k^{i^*} = o_{t_{i^*}}] \leq p'(O_1) < \delta,$$
a contradiction.

So in any case, we must have $p'(O_t) = \delta$.

Now observe that the above argument does not depend on the fact that for any $i \in \Theta$, $\sum_{j \in [m]} p_{i,j} = 1$. Therefore, we can repeatedly apply the characterization of the probability of the topmost combination. That is, we first compute the probability of the topmost combination, and subtract the marginal probabilities contributed by this topmost combination from $\{p_{i,j}\}$. For the new marginal probabilities, the characterization still applies to the new topmost combination, by which we can determine the probability of that combination. This is precisely the procedure implemented in Algorithm 4. By repeatedly applying the characterization, we obtain the unique distribution over $\mathcal{O}^\Theta$ satisfying the conditions of the lemma, which is the output distribution $\{p(O)\}$ of Algorithm 4. □

Given Lemma 4, we then consider any distribution $\{p'(O)\}$ which (1) induces marginal probabilities $\{p_{i,j}\}$, (2) minimizes the expected cost, and (3) among all distributions satisfying (1) and (2), maximizes the potential function w, defined as
$$w(p') = \sum_{O \in \mathcal{O}^\Theta} p'(O) \cdot \left(\sum_{i \in \Theta} O^i\right)^2.$$
The goal is to show such a distribution $\{p'(O)\}$ satisfies the conditions of Lemma 4, and therefore coincides with $\{p(O)\}$, the output of Algorithm 4. Moreover, such a distribution has the additional property, that it minimizes the expected cost. In other words, the output distribution of Algorithm 4 minimizes the expected cost, which is equivalent to the second claim at the beginning of the proof.

To achieve the above goal, we only need to show that the distribution $\{p'(O)\}$ chosen above has the property, that no two combinations of outcomes in the support of $\{p'(O)\}$ cross. Suppose otherwise, i.e., there exist $O_1, O_2 \in \mathcal{O}^\Theta$, such that $O_1 \wedge O_2 \notin \{O_1, O_2\}$ and $p'(O_1) > p'(O_2) > 0$. Let $q = p'(O_2)$. We show that subtracting q from $p'(O_1)$ and $p'(O_2)$ simultaneously and adding q to $p'(O_1 \wedge O_2)$ and $p'(O_1 \vee O_2)$ simultaneously (1) preserves the marginal probabilities, (2) does not increase the expected cost of $\{p'(O)\}$, and (3) strictly increases the potential function w, thus leading to a contradiction. (1) clearly holds. (2) follows from the submodularity of c, i.e.,
$$q \cdot (c(O_1) + c(O_2)) \geq q \cdot (c(O_1 \wedge O_2) + c(O_1 \vee O_2)).$$
And finally, (3) follows from elementary calculation, i.e., whenever O_1 and O_2 cross,
$$\left(\sum_{i \in \Theta} O_1^i\right)^2 + \left(\sum_{i \in \Theta} O_2^i\right)^2 < \left(\sum_{i \in \Theta} \min(O_1^i, O_2^i)\right)^2 + \left(\sum_{i \in \Theta} \max(O_1^i, O_2^i)\right)^2.$$
This establishes second claim.

As for the third claim, i.e., the existence of an efficient first-order oracle, observe that the distribution $\{\mathrm{prob}(O \mid \{p_{i,j}\})\}$ output by Algorithm 4 is piecewise linear in $\{p_{i,j}\}$. On the other hand, the objective function is linear in the output distribution, and is therefore piecewise linear in $\{p_{i,j}\}$. This implies that (sub)gradients of the objective function can be easily computed, and concludes the proof of the theorem. □

We make a few remarks regarding Algorithm 3.

- In addition to the ellipsoid method, one may also apply gradient-based methods, e.g., projected gradient descent, to solve the convex program in Algorithm 3. Gradient-based methods generally perform better in practice, and they usually have better dependence on m and n but worse (polynomial) dependence on $1/\varepsilon$.

- Observe that Algorithm 3 outputs a randomized mechanism such that, in its support, no two combinations of outcomes cross. Therefore, when restricted to binary outcomes ($m = 2$), Lemma 3 gives a way to round the randomized mechanism output by Algorithm 3 into a deterministic one. This gives an alternative way (in addition to Corollary 1) of computing optimal deterministic mechanisms restricted to binary outcomes.

B Omitted Figures and Remarks in Section 2

B.1 Illustration of Algorithm 1

See Figure 1 for an illustration of Algorithm 1.

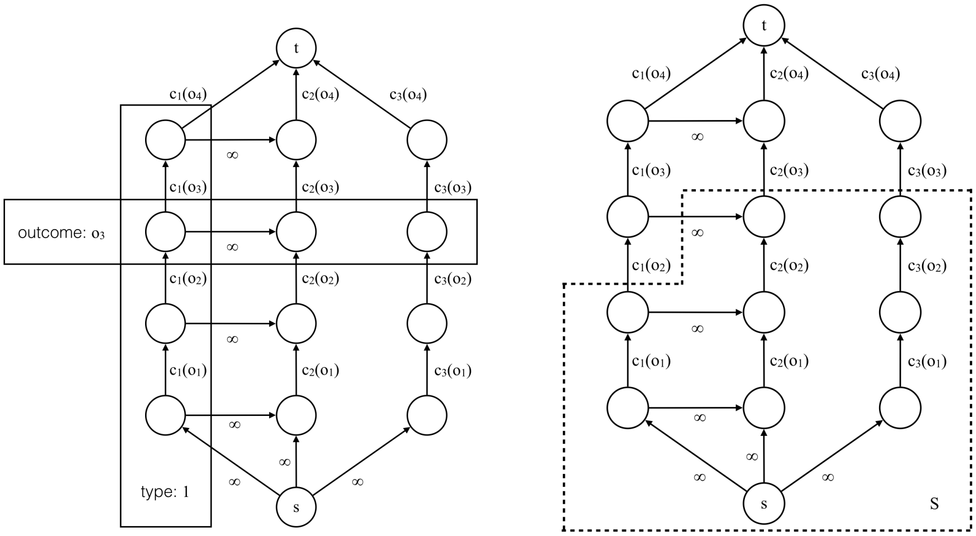

Figure 1: An example of the graph constructed in Algorithm 1. As highlighted in the left graph, each row corresponds to an outcome and each column corresponds to a type. The horizontal edges with infinite capacity correspond to the fact that type 2 can misreport as type 1. The right graph gives a possible 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 $c(M)$ of the mechanism M is equal to the value of the min-cut, which is $c_1(o_2) + c_2(o_3) + c_3(o_3)$.

B.2 Remarks on Algorithm 1

We make two remarks regarding Algorithm 1.

- For finding an optimal deterministic mechanism, the precise values of the agents' utility functions do not matter. Consequently, Algorithm 1 works as long as all types *order* the outcomes in the same way.
- With minor modifications, Algorithm 1 can handle *costly* misreporting, in which there is a fixed (non-negative) cost for type i to report as type i'. Partial verification is a special case of costly misreporting: reporting either costs the agent 0 or ∞, and the reporting structure R is the set of all reporting actions which cost 0. The key modification which allows Algorithm 1 to handle costly misreporting is that the edges used to model the reporting structure can be diagonal (as opposed to horizontal), where the slope of the edge depends on each type's utility function and the cost of misreporting. We will not expand on this in the current paper.

B.3 Remarks on Lemma 1

We make a few remarks regarding Lemma 1.

- The proof we present is a combination of several concrete arguments. There is an alternative relatively high-level, and sometimes more useful, interpretation of the lemma, which is based on a convex program formulation of the problem. We will make heavy use of this alternative interpretation in the rest of the paper, especially when dealing with randomized mechanisms.
- Throughout the paper we assume payments are not allowed. One may show that with payments, there always exists an optimal truthful mechanism that is deterministic, as long as both agents and the principal value payments linearly. Moreover, there exist relatively simple algorithms for computing an optimal mechanism with payments. We will not expand on this in the current paper.

B.4 Remarks on Algorithm 2

We make a few remarks regarding Algorithm 2.

- Algorithm 2 gives a constructive proof that finding an optimal truthful mechanism is always no harder than finding an optimal truthful deterministic mechanism with convex costs. As a result, a faster algorithm for the latter problem would imply a faster algorithm for the former.
- As a byproduct, Algorithm 2 shows that in general, to achieve the minimum cost, it suffices to randomize only between two outcomes for each type, .

C Omitted Proofs in Section 2

Proof of Theorem 1. We give a reduction from MinSAT. Fix a MinSAT instance with n variables, $\{x_i\}_{i\in[n]}$, and m clauses, $\{C_j\}_{j\in[m]}$, and let $\ell_{j,k} \in C_j$ be the k-th literal in clause C_j. We construct an AMD instance as follows.

- Create a type for each variable, each literal, and each clause, i.e., $\Theta = \{x_i, x_i^+, x_i^-\}_{i\in[n]} \cup \{C_j\}_{j\in[m]}$.
- There are two possible outcomes, $\mathcal{O} = \{o^+, o^-\}$. Moreover, for any type $\theta \in \Theta$, $u_\theta(o^+) = 1$ and $u_\theta(o^-) = 0$.
- The principal's cost is as follows.
  - For each literal ℓ, $c_\ell(o^+) = c_\ell(o^-) = 0$.
  - For each variable x_i, $c_{x_i}(o^+) = 0$ and $c_{x_i}(o^-) = m + 1$, so any optimal mechanism never assigns o^- to a variable.
  - For each clause C_j, $c_{C_j}(o^+) = 1$ and $c_{C_j}(o^-) = 0$, so any optimal mechanism minimizes the number of clauses which get outcome o^+.
- The reporting structure R is as follows.
  - Each literal ℓ can only report itself.
  - Each variable x_i can report itself and its two literals x_i^+ and x_i^-.
  - Each clause C_j can report itself, all variables, and all literals $\ell_{j,k} \in C_j$ contained in C_j.

Now consider the structure of optimal solutions for the above AMD instance. First observe that without loss of generality, any optimal solution assigns o^+ only to types which report literals. Moreover, for each variable x_i, any optimal solution assigns o^+ to exactly one of x_i^+ and x_i^-. So the problem boils down to choosing between the two literals for each variable.

On the other hand, each clause C_j will report any literal that is contained in C_j and assigned outcome o^+, as long as possible. Whenever this happens, the principal incurs cost 1 from this clause. In other words, the principal incurs cost 1 from a clause iff one of the literals contained in the clause is assigned outcome o^+, i.e., iff the clause is satisfied. The total cost of the mechanism is $(m+1)n + k$, where k is the number of clauses satisfied. This encodes precisely the MinSAT instance. □

Proof of Theorem 2. Consider the following reduction from MinSAT. Fix a MinSAT instance with n variables, $\{x_i\}_{i\in[n]}$, and m clauses, $\{C_j\}_{j\in[m]}$, and let $\ell_{j,k} \in C_j$ be the k-th literal in clause C_j. We construct an AMD instance as follows.

- Create a type for each variable, each literal, and each clause, i.e., $\Theta = \{x_i, x_i^+, x_i^-\}_{i \in [n]} \cup \{C_j\}_{j \in [m]}$.
- There are three possible outcomes, $\mathcal{O} = \{o^+, o^-, o^0\}$.
- Let $N_1 >> N_2 > m$ be large (but polynomial in n and m) numbers. The principal's cost is as follows.
  - For each variable x_i, $c_{x_i}(o^+) = c_{x_i}(o^-) = 0$, and $c_{x_i}(o^0) = N_1$. As a result, an optimal mechanism never assigns o^0 to a variable.
  - For each positive literal ℓ^+, $c_{\ell^+}(o^+) = N_2$, $c_{\ell^+}(o^0) = 0$, and $c_{\ell^+}(o^-) = N_1$. For each negative literal ℓ^-, $c_{\ell^-}(o^+) = N_1$, $c_{\ell^-}(o^0) = 0$, and $c_{\ell^-}(o^-) = N_2$. We will see later that for any variable x_i, an optimal mechanism assigns precisely one of its literals the outcome with cost N_2, and the other outcome o^0 with cost 0.
  - For each clause C_j, $c_{C_j}(o^+) = c_{C_j}(o^-) = 0$, and $c_{C_j}(o^0) = 1$.
- The types' utility functions are as follows.
  - For each variable x_i, $u_{x_i}(o^+) = u_{x_i}(o^-) = u_{x_i}(o^0)$. Note that the numerical values of the utility functions do not matter for deterministic mechanisms.
  - For each positive literal ℓ^+, $u_{\ell^+}(o^+) > u_{\ell^+}(o^0) > u_{\ell^+}(o^-)$. For each negative literal ℓ^-, $u_{\ell^-}(o^+) < u_{\ell^-}(o^0) < u_{\ell^-}(o^-)$.
  - For each clause C_j, $u_{C_j}(o^0) > u_{C_j}(o^+) = u_{C_j}(o^-)$.
- The reporting structure R is as follows.
  - Each variable x_i can only report itself.
  - Each literal ℓ can report itself or the variable it corresponds to.
  - Each clause C_j can report itself, any literal $\ell \in C_j$ contained in the clause, or the variable ℓ corresponds to.

Now consider the structure of optimal deterministic mechanisms. For each variable x_i, an optimal mechanism assigns either o^+ or o^-. Moreover, for x_i's two literals, if x_i is assigned o^+ (resp. o^-), then the mechanism always assigns x_i^+ o^+ (resp. o^0) and x_i^- o^0 (resp. o^-). One may check this is the only way to minimize cost subject to incentive compatibility. So conceptually, the mechanism chooses exactly one value for each variable, where assigning o^0 to x_i^+ (resp. x_i^-) corresponds to choosing value 1 (resp. 0) for x_i.

For each clause C_j, if any of the literals contained in C_j is chosen (i.e., is assigned outcome o^0), then to prevent C_j from misreporting that literal, the mechanism must assign C_j outcome o^0, at a cost of 1. This corresponds to the case where the clause is satisfied. Otherwise, if none of the literals in C_j is chosen, the mechanism assigns either o^+ or o^- to C_j, at a cost of 0. The total cost of the mechanism is then $nN_2 + k$, where k is the number of clauses satisfied. This encodes precisely the MinSAT instance. □

Proof of Theorem 3. First consider the runtime of Algorithm 1. The bottleneck is finding an s-t min-cut on the graph G, which has $mn + 2$ vertices and at most $mn^2 + (m + 1)n$ edges. Therefore, it is sufficient to show that one can replace the infinite capacities with capacity $W + 1$.

We first prove that any horizontal edge with capacity $W + 1$ does not belong to any min-cut. Suppose in some min-cut, for some $j \in [m]$, a horizontal edge from $(i_1, o_j) \in S$ to $(i_2, o_j) \notin S$ is cut. We argue that including all out-neighbors of (i_1, o_j) through horizontal edges into S strictly decreases the capacity of the cut. For each of these horizontal out-neighbors, by including it in S, we decrease the cut value by $W + 1$ (from one horizontal edge), and possibly incur an additional cost from the edge between that neighbor and its vertical out-neighbor, whose capacity is at most W. Because we take the transitive closure of R, the newly included vertices do not have any horizontal out-neighbor out of S, so the total cost decreases at least by 1. A similar argument shows that edges leaving S can be replaced to have capacity $W + 1$ as well.

Now we move on to proving the correctness of Algorithm 1. We assume the infinite-capacity edges still have capacity ∞ (rather than $W + 1$), which simplifies our argument. Observe that with infinite capacities, taking the transitive closure of R in Line 3-5 of Algorithm 1 makes no difference. We prove the correctness for the algorithm without this step.

The argument consists of two parts. First we show there is a one-to-one correspondence between all finite-capacity *downward-closed* s-t cuts and all deterministic truthful mechanisms, where the capacity of the cut is the same as the cost of the mechanism. We then show that taking the *downward closure* of any cut does not increase its capacity, and as a result, we only need to consider downward-closed cuts. These two claims together imply the correctness of Algorithm 1.

Formally, a cut $(S, \overline{S})$ is downward closed, if for any $i \in \Theta$ and $1 \leq j_1 < j_2 \leq m$,

$$(i, o_{j_2}) \in S \implies (i, o_{j_1}) \in S.$$

Fix a downward closed cut $(S, \overline{S})$, we construct a mechanism $M : \Theta \to \mathcal{O}$ in the same way as in Line 17-19 of Algorithm 1. That is, for all $i \in \Theta$,

$$M(i) = \max\{o_{j'} \in \mathcal{O} \mid (i, o_{j'}) \in S\}.$$

The one-to-one correspondence follows immediately from the definition of M. We now argue $(S, \overline{S})$ has finite capacity iff M is truthful. Notice that $(S, \overline{S})$ has finite capacity iff no horizontal edge is cut, i.e., iff $M(i_1) \geq M(i_2)$ for all $(i_1, i_2) \in R$, which is precisely the condition for the truthfulness of M. Moreover, whenever $(S, \overline{S})$ has finite capacity, the capacity is equal to the cost of the truthful mechanism M.

Now we prove the second claim, i.e., taking the downward closure does not increase the capacity of the cut. We first define the downward closure. Given any s-t cut $(S, \overline{S})$, the downward closure $(C(S), \overline{C(S)})$ is defined such that for all $i \in \Theta$,

$$(i, o_j) \in C(S) \iff j \leq \max\{j' \in [m] \mid (i, o_{j'}) \in S\}.$$

We show below that the capacity of $C(S)$ is no larger than that of S.

If some horizontal edge is cut in S, then the cut has capacity ∞ and the claim is trivial. Suppose no horizontal edge is cut in S. Because the set of vertical edges cut in $C(S)$ is a subset of those cut in S, we only need to show that no horizontal edge is cut in $C(S)$. Suppose for contradiction that some horizontal edges are cut in $C(S)$. Let $i_1, i_2 \in \Theta$ and $j \in [m]$ be such that $e = ((i_1, o_j), (i_2, o_j))$ is one of the highest horizontal edges being cut in $C(S)$. By the choice of e, it must be the case that $(i_1, o_j) \in S$, and since $S \subseteq C(S)$, we have $(i_2, o_j) \notin S$. Therefore, the same edge e is also cut in S, which leads to a contradiction. □

Proof of Lemma 1. We prove the lemma by construction. Let M be any (possibly randomized) optimal truthful mechanism. We construct a deterministic truthful mechanism from M whose cost is no larger than that of M.

First we show it suffices for M to randomize between only two consecutive outcomes for each type i. Let $p_i(o_j)$ be the probability that type i receives outcome o_j. Suppose for some type i, there exist $j_1, j_2 \in [m]$, where $j_2 - j_1 > 1$, $p_i(o_{j_1}) > 0$, and $p_i(o_{j_2}) > 0$. We argue that one can move probability mass from o_{j_1} and o_{j_2} to o_{j_3}, where $j_3 = j_1 + 1$ lies between j_1 and j_2, without violating truthfulness or increasing the total cost.

Let $0 < \alpha < 1$ be such that $o_{j_3} = \alpha o_{j_1} + (1 - \alpha) o_{j_2}$. Without loss of generality suppose $p_i(o_{j_1})/\alpha \leq p_i(o_{j_2})/(1 - \alpha)$. For brevity let $p = p_i(o_{j_1})$. We show that the following operation achieves the above goal.

- Decrease $p_i(o_{j_1})$ by p.
- Decrease $p_i(o_{j_2})$ by $(1 - \alpha) \cdot p/\alpha \leq p_i(o_{j_2})$.
- Increase $p_i(o_{j_3})$ by p/α.

Observe that (1) after the operation, the probabilities of each outcome still sum to 1, and (2) type i receives exactly the same expected utility. The principal's cost changes by

$$\begin{aligned}
&\frac{p}{\alpha} \cdot c_i(o_{j_3}) - p \cdot c_i(o_{j_1}) - \frac{(1-\alpha) \cdot p}{\alpha} \cdot c_i(o_{j_2}) \\
&= \frac{p}{\alpha} \cdot (c_i(o_{j_3}) - \alpha \cdot c_i(o_{j_1}) - (1-\alpha) \cdot c_i(o_{j_2})) \\
&= \frac{p}{\alpha} \cdot \left(c_i^\ell(o_{j_3}) - \alpha \cdot c_i^\ell(o_{j_1}) - (1-\alpha) \cdot c_i^\ell(o_{j_2})\right) && (c_i^\ell \text{ extends } c_i) \\
&\le \frac{p}{\alpha} \cdot \left(c_i^\ell(o_{j_3}) - c_i^\ell(\alpha \cdot o_{j_1} + (1-\alpha) \cdot o_{j_2})\right) && (\text{convexity of } c_i^\ell) \\
&= \frac{p}{\alpha} \cdot \left(c_i^\ell(o_{j_3}) - c_i^\ell(o_{j_3})\right) = 0. && (o_{j_3} = \alpha \cdot o_{j_1} + (1-\alpha) \cdot o_{j_2})
\end{aligned}$$

In other words, the total cost does not increase.

We then apply the above operation in the following way. Fix i, and let $j_- = \min\{j \mid p_i(o_j) > 0\}$, and $j_+ = \max\{j \mid p_i(o_j) > 0\}$. As long as $j_+ - j_- > 1$, apply the operation to i, j_- and j_+. Observe that each time we apply the operation, $j_+ - j_-$ decreases by at least 1, so eventually we must stop and $j_+ - j_- \le 1$. Performing this for each i yields a mechanism which randomizes only between two consecutive outcomes for each type, without increasing the total cost. Without loss of generality, from now on, we assume M has this property.

Now we show there is a way to round M, producing a distribution over deterministic truthful mechanisms, such that the expected cost of this distribution is precisely the cost of M. As a result, there exists one mechanism in the support of the distribution, whose cost is upper bounded by that of M, which is our desired deterministic truthful mechanism.

For each type i, let $j_i = \min\{j \mid p_i(j) > 0\}$ and $\alpha_i = p_i(j_i + 1)$. Note that $0 \le \alpha_i < 1$ for all i. For any $r \in [0, 1]$, let M_r be the deterministic mechanism defined such that for each type i,

$$M_r(i) = \begin{cases} o_{j_i+1}, & r \le \alpha_i, \\ o_{j_i}, & \text{otherwise.} \end{cases}.$$

We first argue that M_r is truthful for any $r \in [0, 1]$. Fix any pair $(i_1, i_2) \in R$. Given that M itself is truthful, we proceed by considering the following two cases.

- $j_{i_1} > j_{i_2}$. In such cases, we always have $M_r(i_1) \ge M_r(i_2)$, so i_1 has no incentive to report i_2.
- $j_{i_1} = j_{i_2}$ and $\alpha_{i_1} \ge \alpha_{i_2}$. For any $r \in [0, 1]$, we have

  $$r \le \alpha_{i_1} \impliedby r \le \alpha_{i_2}.$$

  So again, regardless of r, $M_r(i_1) \ge M_r(i_2)$.

Applying the above argument to each pair $(i_1, i_2) \in R$ establishes the truthfulness of M_r for any $r \in [0, 1]$.

Now consider the distribution over deterministic mechanisms M_r when r is uniformly distributed over $[0, 1]$. We show that the expected cost of M_r is equal to the cost of M:

$$\begin{aligned}
\mathbb{E}_r[c(M_r)] &= \sum_{i \in \Theta} (\Pr[r \le \alpha_i] \cdot c_i(o_{j_i+1}) + \Pr[r > \alpha_i] \cdot c_i(o_{j_i})) \\
&= \sum_{i \in \Theta} (\alpha_i \cdot c_i(o_{j_i+1}) + (1-\alpha_i) \cdot c_i(o_{j_i})) \\
&= \sum_{i \in \Theta} (p_i(j_i+1) \cdot c_i(o_{j_i+1}) + p_i(j_i) \cdot c_i(o_{j_i})) \\
&= \sum_{i \in \Theta} \mathbb{E}_M[c_i(M(i))] \\
&= c(M),
\end{aligned}$$

which concludes the proof. □

Proof of Theorem 4. We prove the correctness first. Observe that the problem of finding a (randomized) optimal mechanism can be written as the following linear program.

$$\begin{aligned}
\min \quad & \sum_{i\in\Theta=[n],j\in[m]} p_{i,j}\cdot c_i(o_j) & \\
\text{subject to} \quad & u_i = \sum_j p_{i,j}\cdot o_j & \forall i\in\Theta \\
& u_{i_1} \ge u_{i_2} & \forall (i_1,i_2)\in R \\
& \sum_j p_{i,j} = 1 & \forall i\in\Theta \\
& p_{i,j}\ge 0 & \forall i\in\Theta, j\in[m].
\end{aligned}$$

Here, u_i is the expected utility of type i, and $p_{i,j}$ is the probability of assigning type i outcome o_j. This is not the most succinct LP formulation of the problem, but it capture the structure of the problem in a way that is more useful for our analysis.

Now fix $\{u_i\}_i$ and consider the optimal choice of $\{p_{i,j}\}_{i,j}$. This can be solved separately for each type i, by considering the following linear program (with the additional constraints that $\{p_{i,j}\}_j$ are nonnegative and sum up to 1 for all i).

$$\begin{aligned}
\min \quad & \sum_j p_{i,j}\cdot c_i(o_j) \\
\text{s.t.} \quad & \sum_j p_{i,j}\cdot o_j = u_i.
\end{aligned}$$

This is precisely evaluating the convex envelope c_i^- of c_i at $u_i\in[o_1,o_m]$. Consequently, the problem of finding a (randomized) optimal mechanism can be rewritten as the following convex program.

$$\begin{aligned}
\min \quad & \sum_{i\in\Theta} c_i^-(u_i) & \\
\text{s.t.} \quad & u_{i_1}\le u_{i_2}, & \forall (i_1,i_2)\in R, \\
& u_i\in[o_1,o_m], & \forall i\in\Theta.
\end{aligned}$$

Now observe that the reformulated program cannot distinguish between $\{c_i\}_{i\in\Theta}$ and $\{\widehat{c_i}\}_{i\in\Theta}$, where $\widehat{c_i}$ is c_i^- restricted to $\mathcal{O}$ as in Algorithm 2 — the two cost functions simply induce exactly the same program. Moreover, observe that the newly constructed cost function $\{\widehat{c_i}\}_{i\in\Theta}$ is convex, according to Definition 1. Given this convexity, Lemma 1 implies that there exists a deterministic mechanism for $\{\widehat{c_i}\}_i$ which is optimal. In other words, there exists an optimal solution $\{u_i\}_i$ to the reformulated program in which $u_i\in\mathcal{O}$ for each $i\in\Theta$. Algorithm 2 finds such a solution $\{u_i\}_i$ by calling Algorithm 1.

Now the only problem left is to recover $\{p_{i,j}\}_j$ from u_i for each type i. This is done in Line 6 of Algorithm 2. Since the output mechanism M implements $\{u_i\}_i$ in an optimal way, it is a truthful mechanism that minimizes the principal's cost. This establishes the correctness of Algorithm 2.

Now we consider the time complexity. Compared to Algorithm 1, the additional steps in Algorithm 2 include (1) computing $\{\widehat{c_i}\}_i$ in Line 2, and (2) interpreting $\widehat{M}(i)$ as an optimal convex combination of outcomes in Line 6. We will show that both operations can be done in time $O(mn)$, i.e., linear in the size of the input. The time complexity of Algorithm 2 then follows immediately.

For computing the convex envelope of c_i, one may use the classical algorithm by Andrew [1], which scans $\mathcal{O}$ from left to right, and maintains a stack containing the partial convex envelope of c_i on $\{o_1,\dots,o_j\}$ for every $j\in[m]$. The algorithm runs in time $O(m)$.

Once we know $\{\widehat{c_i}\}$, to find an optimal convex combination for a target utility u_i, we first find in time $O(m)$ the largest integer $\ell\in[m]$ such that $o_\ell\le u_i$ and $c_i(o_\ell)=\widehat{c_i}(o_\ell)$, and the smallest integer $r\in[m]$ such that $o_r\ge u_i$ and $c_i(o_r)=\widehat{c_i}(o_r)$. If $\ell=r$, then we output $o_\ell=o_r$. Otherwise, there is a unique $\alpha\in(0,1)$ such that randomizing between o_ℓ and o_r gives expectation $\alpha\cdot o_\ell+(1-\alpha)\cdot o_r=u_i$. The convex envelope is linear between o_l and o_r, and hence $\alpha\cdot c_i(o_\ell)+(1-\alpha)\cdot c_i(o_r)=\widehat{c_i}(u_i)$. Then, we can set $M(i)$ to o_ℓ with probability α and to o_r with probability $(1-\alpha)$. Performing the above for every type i takes $O(mn)$ time. □