Mechanism Design: from Partial to Probabilistic Verification

IOANNIS CARAGIANNIS, University of Patras & CTI, Greece
EDITH ELKIND, Nanyang Technological University, Singapore
MARIO SZEGEDY, Rutgers, the State University of New Jersey
LAN YU, Nanyang Technological University, Singapore

Algorithmic mechanism design is concerned with designing algorithms for settings where inputs are controlled by selfish agents, and the center needs to motivate the agents to report their true values. In this paper, we study scenarios where the center may be able to verify whether the agents report their preferences (types) truthfully. We first consider the standard model of mechanism design with partial verification, where the set of types that an agent can report is a function of his true type. We explore inherent limitations of this model; in particular, we show that the famous Gibbard–Satterthwaite impossibility result holds even if a manipulator can only lie by swapping two adjacent alternatives in his vote. Motivated by these negative results, we then introduce a richer model of verification, which we term *mechanism design with probabilistic verification*. In our model, an agent may report any type, but will be caught with some probability that may depend on his true type, the reported type, or both; if an agent is caught lying, he will not get his payment and may be fined. We characterize the class of social choice functions that can be truthfully implemented in this model. We then proceed to study the complexity of finding an *optimal* individually rational implementation, i.e., one that minimizes the center's expected payment while guaranteeing non-negative utility to the agent, both for truthful and for non-truthful implementation. Our hardness result for non-truthful implementation answers an open question recently posed by Auletta et al. [2011].

Categories and Subject Descriptors: F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; J.4 [**Computer Applications**]: Social and Behavioral Sciences–Economics

General Terms: Economics, Theory, Algorithms

Additional Key Words and Phrases: algorithmic mechanism design, verification

1. INTRODUCTION

Algorithmic mechanism design [Nisan and Ronen 2001; Nisan 2007] is a burgeoning research area that deals with designing algorithms for settings where inputs are controlled by selfish agents. A center wants to implement some function of the inputs, but agents have preferences over possible outcomes and may lie about their inputs if it is profitable for them to do so. Thus, the center needs to create incentives for truthful reporting, usually by offering payments to the agents. Examples of problems in this

Ioannis Caragiannis is supported by EC-funded STREP Project FP7-ICT-258307 EULER. Edith Elkind and Lan Yu are supported by Singapore NRF Research Fellowship 2009-08. Mario Szegedy is supported by NSF grant CCF-0832787.
Author's addresses: I. Caragiannis, Computer Technology Institute & Department of Computer Engineering and Informatics, University of Patras, 26504 Rio, Greece; E. Elkind and L. Yu, Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, 21 Nanyang Link, 637371, Singapore; M. Szegedy, Department of Computer Science, Rutgers, the State University of New Jersey, 110 Frelinghuysen Road, Piscataway, NJ 08854, USA.

area include combinatorial auctions, resource allocation, scheduling, facility location, network creation, and many other problems of practical significance.

Let us describe this problem more formally, focusing on a setting with a single selfish agent[1]. In its most abstract form, a mechanism design problem can be specified by a space $\mathcal{O}$ of possible *outcomes* (say, allocations of items in an auction or job assignments in a scheduling scenario), a set of possible *types* D, where elements of D are functions from $\mathcal{O}$ to $\mathbb{R}$ (the value $v(o)$, $v \in D$, is the benefit that the agent enjoys if outcome o is chosen), and a *social choice function* $f : D \to \mathcal{O}$, which selects an outcome given the agent's reported type. The function f represents the center's preferences, which may differ from those of the agent: that is, $f(v)$ is not necessarily chosen from $\arg\max_{o \in \mathcal{O}} v(o)$. Therefore the agent may lie about his type: instead of submitting his true type v (and obtaining an outcome o), an agent may submit $v' \neq v$ if this leads to an outcome o' that he likes better (i.e., $v(o') > v(o)$). Typically, this issue is handled by designing a *payment rule* $p : D \to \mathbb{R}$ that incentivizes the agent to report truthfully: we want p to be such that agent i maximizes his utility (the sum of the enjoyment he derives from the selected outcome and the payment he gets) by submitting his true type v. If p has this property, the pair (f, p) (such pairs are usually called *mechanisms*) is said to *(truthfully) implement f*.

Not all social choice functions are implementable: for instance, it is impossible to schedule tasks on unrelated machines so as to minimize the makespan if the machines may lie about the time it takes them to execute each task [Nisan and Ronen 2001]. For rich enough domains, only a restricted class of social choice functions admits an implementation [Roberts 1979; Saks and Yu 2005]. In a sense, many of these impossibility results stem from the fact that an agent may declare *any* type in the domain. Motivated by this observation, Nisan and Ronen [2001] proposed a model where agents' misreporting is restricted: the set of types that an agent may report is determined by his true type and may be a strict subset of D. This model can be conveniently encoded as a directed graph M with a vertex set D: an edge from u to v means that an agent of type u can report v. For instance, in the context of scheduling, it makes sense to assume that a machine can understate its speed, but not overstate it. In other settings, such as, e.g., voting, an agent may be able to submit a ranking of the candidates that differs from his true ranking in a few positions, but will be detected if his ranking is the opposite of his true preferences. In the economics literature, this setting is known as *mechanism design with partial verification*; the term "partial" refers to the fact that the center may detect some, but not all of the lies. It has been first formalized by Green and Laffont [1986], and subsequently studied in a number of papers. A number of strong positive results in the partial verification model inspired by [Nisan and Ronen 2001] have been obtained for several important domains, including scheduling and combinatorial auctions [Auletta et al. 2006; Penna and Ventre 2008, 2009; Auletta et al. 2009; Krysta and Ventre 2010].

An interesting feature of the partial verification model is that, to implement a social choice function f, the center may have to incentivize the agents to misreport their preferences. That is, the center may declare that it is going to execute a mechanism (g, p) with $g \neq f$, but set up the incentives so that an agent with type u prefers to report a type v with $g(v) = f(u)$. Thus, while it may look like the center is not implementing f, once the agents' strategic behavior is taken into account, the outcome on a type u is exactly $f(u)$. A mechanism (g, p) with this property is called a *non-truthful implementation* of f. Now, in the setting without verification there is no need to jump through these hoops: the famous revelation principle [Myerson 1981] says that any so-

[1]It is known that for the type of questions studied in this paper, there is no loss of generality in focusing on single-agent settings; see, e.g., [Archer and Kleinberg 2008; Auletta et al. 2011] for an explanation.

cial choice function f that can be implemented in this roundabout manner can also be implemented truthfully, i.e., by combining the target social choice function with a suitable payment rule. In contrast, in the partial verification model this is not the case: there are social choice functions that can be implemented non-truthfully, but do not have a truthful implementation. However, as shown in a recent paper by Auletta et al. [2011], non-truthful implementations are computationally hard to find; this hardness result holds even when the graph M is acyclic and the outdegree of each vertex is at most 3.

In this paper, we continue the investigation of the partial verification model. In Section 3 we identify two natural classes of partial verification settings: the former can be interpreted as "one-sided" verification, and encompasses the verification models considered in [Singh and Wittman 2001; Auletta et al. 2006, 2009; Krysta and Ventre 2010], while the latter can be viewed as "distance-based" verification. In more detail, in the latter model, the set of types that a player is allowed to report consists of all types that are sufficiently close to his true type, according to some notion of distance. We observe that, in contrast with one-sided verification, distance-based verification does not seem to be useful for designing truthful mechanisms. Specifically, we investigate two settings that admit a natural notion of distance—convex domains and voting—and show that in both of them the set of truthfully implementable social choice functions under distance-based verification is the same as in the model without verification. Our result for the voting domain can be informally summarized as follows: the famous Gibbard–Satterthwaite impossibility result [Gibbard 1973; Satterthwaite 1975] remains true even if the manipulating voter is restricted to swapping a single pair of candidates in his vote.

Motivated by these negative results, we then proceed to study a broader class of verification settings. Our starting point is the observation that in the partial verification model of [Green and Laffont 1986] lie detection is fully deterministic: either an agent of type v can declare a type $v' \neq v$ without any risk of being caught, or he simply cannot declare v' as his type. However, in many real-life scenarios lie detection is probabilistic: an agent can report any type v' that differs from his true type v, and is caught with a certain probability. This probability may depend on both v and v'; in particular, in settings where there is a natural notion of distance between types, such as the ones described in Section 3, the detection probability is likely to grow with the distance from v to v'. If a lie is detected, the lying agent is usually punished: if the center was supposed to pay the agent, the payment may be withheld, and the agent may have to pay a fine; again, the fine may depend on the agent's true type, the declared type, or both.

For instance, consider a member of a decision-making body (let us call him Mr. X) who is supposed to vote by submitting his ranking of several alternatives, such as budget proposals or nominees for an administrative post. Let us denote the available alternatives by A, B, and C, and suppose that Mr. X's true ranking of the alternatives is $A > B > C$. Moreover, he once wrote a private e-mail in which he argued that A is preferable to B, and on another occasion he told a group of supporters that he prefers A to C. Now, if Mr. X votes $B > A > C$ for strategic reasons, his reputation may be damaged if that private e-mail of his is leaked, Thus, when he weighs the cost and the benefits of the strategic vote, he must take into account the leakage probability. Voting $B > C > A$ is even more dangerous, as there is an additional risk that the position he expressed when talking to his supporters becomes publicly known.

Another example is provided by job scheduling. Consider a machine that has speed s, but may declare a speed $s' \neq s$. Suppose that, as a result, this machine is allocated a workload of $L > 0$. The center can observe whether the machine starts working immediately (and withhold the payment if it does not), and, once started, the machine cannot stop until it completes its load. However, the center cannot observe the actual speed of the machine or monitor the entire execution: all it can do is check upon this machine once to see if it is still working. The center aims to schedule its check around the expected completion time, i.e., L/s', but, due to other demands on its resources, it may be unable to time it perfectly. Thus, the timing of the check will be chosen uniformly from the interval $[L/s' - \delta, L/s' + \delta]$ for some $\delta > 0$. If the inspection detects cheating (i.e., it took place before time L/s' and the machine had already stopped working, or it took place after time L/s' and the machine was still working), the center can withhold payment. Clearly, for a fixed value of δ the detection probability depends on both s and s': more specifically, it grows linearly with $|\frac{L}{s} - \frac{L}{s'}|$ as long as $|\frac{L}{s} - \frac{L}{s'}| \leq \delta$, and becomes $1/2$ when $|\frac{L}{s} - \frac{L}{s'}| \geq \delta$.

In Section 4 we provide a formal model for such scenarios by introducing the framework of *mechanism design with probabilistic verification*; this is our main conceptual contribution. Our model allows for probabilistic lie detection and fines, and can be shown to generalize the partial verification model. We characterize the set of social choice functions that can be truthfully implemented in this model; our proof is based on a modification of the graph-theoretic argument given in [Vohra 2011]. Our results indicate that probabilistic verification can be very powerful. In particular, whenever all lie detection probabilities are strictly positive, *any* social choice function can be implemented: intuitively, if payments are large enough, even a small chance of not receiving them makes a player reluctant to lie.

Of course, large payments are undesirable from the center's perspective. Therefore, in Section 5 we investigate the natural question of finding an implementation that minimizes the center's expenses, subject to the condition that the agent's expected benefit from participation in the mechanism is non-negative (this constraint is usually referred to as *individual rationality*). For truthful implementation, we show that a unique optimal solution exists and can be found in strongly polynomial time. For non-truthful implementation, this problem is much more difficult: we show that it is NP-hard even for partial verification (and hence, *a forteriori*, for probabilistic verification) and even when the misreport graph M has maximum outdegree 2; our proof also resolves an open problem of [Auletta et al. 2011] regarding implementation without payments. Thus, the best we can hope for is to find an efficient algorithm for computing an optimal non-truthful implementation in settings where each agent can report at most one non-truthful type. We make partial progress towards this goal, by giving a polynomial-time algorithm that computes an optimal non-truthful implementation for the case where types can be arranged on a line, and each agent can either report his true type or declare the type that appears right after his true type in this ordering; any other declaration will be detected with probability 1. These results can be viewed as a contribution to the growing body of work on *automated mechanism design* [Guo and Conitzer 2010].

We believe that the probabilistic verification model raises a number of interesting questions that are waiting to be explored; we formulate some of them in Section 6.

2. PRELIMINARIES AND NOTATION

We start by presenting the general model of mechanism design with partial verification, as proposed by Green and Laffont [1986]; our exposition follows that of Auletta et al. [2011]. Since our focus is on dominant strategy implementation, we will limit our analysis to settings with a single selfish agent.

We assume that there is a set $\mathcal{O}$ of possible *outcomes* and a single agent who has some private information about the outcomes, abstracted as a *valuation function*, or *type* $u : \mathcal{O} \to \mathbb{R}$. The valuation function quantifies how much the agent values each outcome in $\mathcal{O}$. The set of all possible valuation functions is the *domain* D. Throughout the paper, we will assume that $\mathcal{O}$ is a finite set; most (but not all) of our results assume that D is finite, too.

In the usual mechanism design setting, an agent with type u can report any other type $v \in D$. In contrast, in the partial verification model the set of types that the agent can report is limited and may depend on u. This is encoded by a *misreport correspondence* $M : D \to 2^D$, which for each type u specifies the set of types $M(u) \subseteq D$ that the agent can possibly report. This correspondence can be viewed as a directed graph on D where there is an edge from u to v if $v \in M(u)$; we will sometimes refer to M as the *misreport graph* and write $(u, v) \in M$ to indicate that $v \in M(u)$. We only consider correspondences M for which truth-telling is always an option, i.e., $u \in M(u)$ for all $u \in D$; when we describe the graph M, we often omit all self-loops from the description. The standard model (with no verification) corresponds to setting $M(u) = D$ for all u.

A *mechanism* is a pair (g, p) where $g : D \to \mathcal{O}$ is a *social choice function* and $p : D \to \mathbb{R}$ is a *payment function*. Based on the agent's reported type, the social choice function g selects a single outcome from $\mathcal{O}$, and the payment function p assigns to the agent a (positive or negative) payment. The agent's benefit under a mechanism is assumed to be given by a quasi-linear utility function, i.e., it is equal to his valuation of the chosen outcome plus the payment received. Specifically, when he has true type u and reports v, his utility is $u(g(v)) + p(v)$.

We will also sometimes consider the more restricted setting of *mechanism design without payments*, where the function $p(u)$ is required to be identically 0 on D.

We will now present the main definition of this section.

Definition 2.1. A mechanism (g, p) is said to *M-implement*[2] a social choice function $f : D \to \mathcal{O}$ for an agent with quasi-linear utility if for each $u \in D$ there exists a $v \in M(u)$ such that

$$g(v) = f(u) \tag{1}$$

$$u(g(v)) + p(v) \geq u(g(u')) + p(u') \quad \text{for each } u' \in M(u). \tag{2}$$

A function f is called *M-implementable* if there exists a mechanism that M-implements it.

Given a mechanism (g, p) and a social choice function f, we define a function $\phi_{(g,p)} : D \to D$ that given a type u outputs some type v satisfying conditions (1) and (2). Intuitively, under (g, p) an agent with type u weakly prefers to report $\phi_{(g,p)}(u)$, which leads the mechanism to output $g(\phi_{(g,p)}(u)) = f(u)$.

Another notion that will play an important role in our analysis is individual rationality. We say that a mechanism (g, p) that M-implements a social choice function f is *individually rational* if for every $v \in D$ it holds that $u(g(\phi_{g,p}(u))) + p(\phi_{(g,p)}(u)) \geq 0$. In words, under an individually rational mechanism an agent acts so that his utility is non-negative.

A mechanism (g, p) that M-implements f is said to be *truthful* if for all $u \in D$ we have $g(u) = f(u)$ and $u(f(u)) + p(u) \geq u(g(u')) + p(u')$ for each $u' \in M(u)$. In the absence of verification, the revelation principle [Myerson 1981] implies that, without

[2]This definition is somewhat different from the one given in [Auletta et al. 2011]: we assume that the agent breaks ties in a way that suits the mechanism designer's goals (rather than in favor of reporting his true type, as in [Auletta et al. 2011]). This distinction becomes important when we discuss non-truthful mechanisms that minimize the center's expenses: under our definition, if the center wants an agent of type a to report type b, he can achieve this by making b as attractive as a, while under the definition of [Auletta et al. 2011], he will have to make b infinitesimally more attractive, which means that no mechanism would be optimal for the center.

loss of generality, we may limit our attention to truthful mechanisms. However, when partial verification is available, this is no longer the case: our next example shows that the center may prefer a non-truthful implementation over a truthful one.

Example 2.2 (*adapted from [Green and Laffont 1986; Auletta et al. 2011]*).
Consider a setting with $\mathcal{O} = \{T, F\}$, $D = \{u, v, w\}$, and $x(T) = 1$, $x(F) = 0$ for every $x \in D$. Suppose that the correspondence M is given by $M(u) = \{u, v\}$, $M(v) = \{v, w\}$, $M(w) = \{w\}$, and we would like to implement the social choice function $f(u) = F$, $f(v) = f(w) = T$. This social choice function can be truthfully M-implemented by setting $p(u) = 1$, $p(v) = p(w) = 0$; it is not hard to see that this M-implementation minimizes the sum of payments over all individually rational truthful mechanisms. However, f admits a (non-truthful) individually rational M-implementation *without* payments. Indeed, we can set $g(u) = g(v) = F$, $g(w) = T$: under this mechanism $g(u') = F$ for any $u' \in M(u)$, and v and w can both report w to obtain their preferred outcome $g(w) = T$.

Example 2.2 shows that in settings without money not all M-implementable social choice functions are truthfully M-implementable; an alternative interpretation is that allowing non-truthful implementation may reduce the center's expenses. For settings where money is available, an example of a correspondence M and a social choice function f that is M-implementable, but not truthfully M-implementable can be found in [Auletta et al. 2011; Yu 2011].

3. LIMITATIONS OF PARTIAL VERIFICATION

The partial verification model has proved to be quite powerful in settings where the verification is "one-sided", i.e., each outcome is associated with a cost incurred by the agent, and an agent can lie by overstating his cost, but not by understating it, or, alternatively, each outcome is associated with a profit, and the agent can understate his profit, but not overstate it. Under this type of verification, a number of strong results have been obtained for scheduling [Auletta et al. 2006, 2009], combinatorial auctions [Krysta and Ventre 2010], and several other domains.

We will now argue that the "asymmetry" of the correspondence function M plays an important role in these results. To this end, we give two examples of settings where under a very restricted, yet "symmetric" correspondence M the set of social choice functions that can be truthfully implemented is the same as in the setting without verification. In other words, in these settings partial verification does not help at all, as long as we limit ourselves to truthful implementation.

Our first example of a setting where "symmetric" partial verification does not help is that of convex domains, where $M(v)$ is restricted to the ε-neighborhood of v for an arbitrarily small ε. The second scenario is strategic voting, where each voter is only allowed to modify his preference list by swapping two adjacent candidates. As we will see, in either case, the correspondence M is indeed very restrictive, yet these restrictions do not help to achieve truthfulness.

Now, if one is willing to consider non-truthful implementation, our negative results no longer hold. However, this approach suffers from another problem: non-truthful implementations can be hard to find. We will discuss this issue in more detail in Section 5.2.

3.1. Convex Domains

We may view a valuation v as a point in $\mathbb{R}^{\mathcal{O}}$, and domain D as a subset of $\mathbb{R}^{\mathcal{O}}$. This allows us to use the standard notions of distance and convexity in $\mathbb{R}^{\mathcal{O}}$. In particular, for any $\varepsilon > 0$, the *ε-neighborhood* of a valuation v, denoted by $N_\varepsilon(v)$, is defined as the set of all points in D that are at ℓ_2-distance of at most ε from v, and D is said to be *convex* if it is a convex subset of $\mathbb{R}^{\mathcal{O}}$. We remark that a convex domain contains infinitely many points unless it is a singleton.

In this very general setting, one natural restriction on the agent's ability to lie is to require that he reports a type that is ε-close to his true type for some small ε, i.e., to define the misreport correspondence as $M^{\varepsilon}(v) = N_{\varepsilon}(v)$. However, it turns out that no matter how small ε is, this restriction does not enable us to implement any social choice function that is not implementable without verification.

THEOREM 3.1. *For any $\varepsilon > 0$ a social choice function f on a convex domain D is truthfully M^{ε}-implementable if and only if it is truthfully implementable.*

Theorem 3.1 is a direct corollary of Theorem 4.1 in [Archer and Kleinberg 2008]. In fact, Archer and Kleinberg's result extends to infinite outcome spaces. However, it is phrased in terms of local properties of the function f rather than in terms of verification. We remark that Theorem 3.1 also admits a simple direct proof that is based on "stitching" together a sequence of ε-neighborhoods; we omit it due to space constraints.

3.2. Strategic Voting

We will now show that the classic Gibbard–Satterthwaite impossibility result [Gibbard 1973; Satterthwaite 1975] holds even under a very powerful partial verification model.

An *election* is given by a set of m *alternatives* A (the *candidates*), and a set of n *voters* V. Let $\mathcal{L}(A)$ denote the space of all linear orders over A. Each voter i is described by an order $R_i \in \mathcal{L}(A)$, also denoted by $\prec_i$; this order is called i's *preference order*. When $a \prec_i b$ for some $a, b \in A$, we say that voter i *prefers* b to a. The vector $\mathcal{R} = (R_1, \dots, R_n)$, where $R_i \in \mathcal{L}(A)$ for all $i = 1, \dots, n$, is called a *preference profile*. Given a preference profile $\mathcal{R}$ over a set of candidates A, for any preference order $L \in \mathcal{L}(A)$ we denote by $(\mathcal{R}_{-i}, L)$ the preference profile obtained from $\mathcal{R}$ by replacing R_i with L.

A *voting rule* $\mathcal{F}$ is a mapping that, given a preference profile $\mathcal{R}$ over A outputs an element a of A; we write $a = \mathcal{F}(\mathcal{R})$.

We say that a voter i can *manipulate* an election (A, V) with a preference profile $(R_1, \dots, R_n)$ with respect to a voting rule $\mathcal{F}$ if there exists an L such that $\mathcal{F}(\mathcal{R}) \prec_i \mathcal{F}(\mathcal{R}_{-i}, L)$. A voting rule $\mathcal{F}$ is called *incentive compatible* if it cannot be manipulated by any voter on any preference profile. It is not hard to see that the strategic voting problem can be viewed in the mechanism design framework of Section 2, as long as we fix voter i as well as $\mathcal{R}_{-i}$. With this interpretation $\mathcal{F}'(R) = \mathcal{F}(\mathcal{R}_{-i}, R)$ will play the role of a social choice function and incentive compatibility is an analogue of truthful implementability; thus we can use the terminology defined in Section 2 to discuss incentive compatibility under partial verification.

A voting rule is said to be *onto* if for every $a \in A$ there exists a preference profile $\mathcal{R}$ such that $\mathcal{F}(\mathcal{R}) = a$. Being onto is viewed as a minimal requirement for a reasonable voting rule: a voting rule that is not onto will fail to elect some candidate even if he is ranked first by all voters.

Voter i is a *dictator* under a voting rule $\mathcal{F}$ if for all preference profiles $\mathcal{R}$ over A, the winner is simply the top alternative in R_i. $\mathcal{F}$ is called a *dictatorship* if some i is a dictator for it; usually, dictatorships are not viewed as reasonable voting rules.

The Gibbard–Satterthwaite theorem states that for $|A| \geq 3$ there exists no reasonable incentive compatible voting rule.

THEOREM 3.2 ([GIBBARD 1973; SATTERTHWAITE 1975]). *Let $\mathcal{F}$ be an incentive compatible voting rule onto A, where $|A| \geq 3$. Then $\mathcal{F}$ is a dictatorship.*

Now, let us suppose the voters' ability to manipulate the elections is limited: each voter is only allowed to misreport by swapping two adjacent alternatives in his preference list. This can be encoded by the partial verification model as follows: if R is given by $a_1 \prec a_2 \prec \cdots \prec a_m$, we set

$$M_{\text{swap}}(R) = \{R(a_{j-1} \leftrightarrow a_j) : j = 2, 3, ..., m\},$$

where $R(a_{j-1} \leftrightarrow a_j)$ denotes the linear order obtained from R by swapping the adjacent alternatives a_{j-1} and a_j. For $L \in M_{\text{swap}}(R)$, we use L/R to denote the pair of alternatives that is ordered differently in R and L. Clearly, $L/R = \{a, b\}$ implies $L = R(a \leftrightarrow b)$.

The main result of this section is that even under this very restrictive model of manipulation, if $|A| \geq 3$, no reasonable voting rule is incentive compatible. Thus, partial verification cannot be used to circumvent Gibbard–Satterthwaite.

THEOREM 3.3. *If $\mathcal{F}$ is an M_{swap}-incentive compatible voting rule onto A, where $|A| \geq 3$, then $\mathcal{F}$ is a dictatorship.*

A voting rule $\mathcal{F}$ is called *monotone* if for all $i \in V$, all preference profiles $\mathcal{R}_{-i}$ of the other voters and all $R_i, R'_i \in \mathcal{L}(A)$, if it holds that $f(R_i, \mathcal{R}_{-i}) = a$, $f(R'_i, \mathcal{R}_{-i}) = a'$ and $a \neq a'$, then $a' \prec_i a$ and $a \prec'_i a'$. That is, if the winner changed from a to a' when voter i changed his preference from R_i to R'_i, then i must have swapped a and a' in his vote. As shown in [Nisan 2007], this notion is equivalent to incentive compatibility:

PROPOSITION 3.4. *A voting rule is incentive compatible if and only if it is monotone.*

To prove Theorem 3.3 we will show that M_{swap}-incentive compatibility implies monotonicity. Theorem 3.3 then follows by combining this result with Proposition 3.4 and Theorem 3.2.

PROPOSITION 3.5. *If a voting rule $\mathcal{F}$ is M_{swap}-incentive compatible, then it is monotone.*

We start by stating a simple, but useful lemma.

LEMMA 3.6. *Suppose that $\mathcal{F}$ is an M_{swap}-incentive compatible voting rule. For any $i \in V$, any preference profile $\mathcal{R}_{-i}$ of the other voters, any $R_i \in \mathcal{L}(A)$ and any $R'_i \in M_{\text{swap}}(R_i)$, if $\mathcal{F}(R_i, \mathcal{R}_{-i}) = a$, $\mathcal{F}(R'_i, \mathcal{R}_{-i}) = a'$ and $a' \neq a$ then $a' \prec_i a$ and $a \prec'_i a'$.*

PROOF. Immediate from the definition of manipulation. □

We will also need the following fact, which is a well-known property of the so-called *swap distance*, i.e., the distance between two preference profiles that is defined as the number of swaps of adjacent candidates needed to transform one profile into the other.

LEMMA 3.7. *For any two preference orders $R \neq L$, there is a sequence of preferences $R = R^0, R^1, \ldots, R^t = L$ where $R^j \in M_{\text{swap}}(R^{j-1})$, $j = 1, 2, ..., t$. Moreover, this sequence can be chosen so that the pair of alternatives swapped never repeats, i.e., $R^{j-1}/R^j \neq R^{k-1}/R^k$ for any $j \neq k$.*

Now we are ready to prove Proposition 3.5.

PROOF. Let $\mathcal{F}$ be an M_{swap}-incentive compatible voting rule. Fix a voter i with a preference ordering R_i, a preference profile $\mathcal{R}_{-i}$ of the other voters, and a preference ordering R'_i. Let $a = \mathcal{F}(R_i, \mathcal{R}_{-i})$, $a' = \mathcal{F}(R'_i, \mathcal{R}_{-i})$. Out goal is to show that $a' \prec_i a$, $a \prec'_i a'$.

By Lemma 3.7, there is a sequence of preferences $R_i = R^0, R^1, \ldots, R^t = R'_i$ where $R^j \in M_{\text{swap}}(R^{j-1})$, $j = 1, 2, ..., t$. Let $m_1 < m_2 < \cdots < m_r$ be the indices where

$\mathcal{F}(R^{m_k-1}, \mathcal{R}_{-i}) \neq \mathcal{F}(R^{m_k}, \mathcal{R}_{-i})$, $k = 1, 2, ..., r$. Let $a_k = \mathcal{F}(R^{m_k}, \mathcal{R}_{-i})$. For convenience of notation, set $m_0 = 0$, $m_{r+1} = t + 1$, and $a_0 = \mathcal{F}(R^0, \mathcal{R}_{-i}) = a$. Now the sequence is divided by the m_k's into $r + 1$ blocks, and within the k-th block the winner is always a_{k-1}: for $m_{k-1} \leq j \leq m_k - 1$ it holds that $\mathcal{F}(R^j, \mathcal{R}_{-i}) = a_{k-1}$.

For each $k = 1, \ldots, r$, we look at the border of the block: as we move from R^{m_k-1} to R^{m_k}, the winner switches from a_{k-1} to a_k. Since $R^{m_k} \in M_{\text{swap}}(R^{m_k-1})$, we have $\mathcal{F}(R^{m_k-1}, \mathcal{R}_{-i}) = a_{k-1} \neq a_k = \mathcal{F}(R^{m_k}, \mathcal{R}_{-i})$, so by Lemma 3.6 we conclude that R^{m_k} is obtained from R^{m_k-1} by swapping a_{k-1} and a_k.

Moreover, by Lemma 3.7 alternatives a_{k-1} and a_k never get swapped anywhere else in the sequence of preferences, i.e., $R^{j-1}/R^j \neq \{a_{k-1}, a_k\}$ for all $j \neq m_k$. Since for all $j \neq m_k$ the preference rankings R^{j-1} and R^j order a_{k-1} and a_k in the same way, for the preference orders that precede R^{m_k-1}, the alternative a_{k-1} is preferred to a_k, just as in R^{m_k-1}; for the preference orders that follow R^{m_k}, a_k is preferred to a_{k-1}, just as in R^{m_k}. In particular, $a_k \prec_i a_{k-1}$ and $a_{k-1} \prec'_i a_k$.

Since $a_k \prec_i a_{k-1}$ holds for all $k = 1, 2, \ldots, r$, by transitivity we get $a' = a_r \prec_i a_0 = a$. Similarly, we have $a \prec'_i a'$. This completes the proof. □

4. PROBABILISTIC VERIFICATION MODEL

Motivated by the negative results of the previous section, we will now define the probabilistic verification model that was informally discussed in Section 1. As before, since we are interested in dominant strategy implementation, we focus on the single agent case.

We assume that for each pair of types $u, v \in D$ we are given a pair of numbers $\lambda[u, v] \in [0, 1]$ and $\psi[u, v] \in \mathbb{R}^+ \cup \{0, +\infty\}$: $\lambda[u, v]$ is the probability that a player with type u can report type v and not get caught, and $\psi[u, v]$ is the fine[3] that a player of type u has to pay when he is caught reporting v. We require $\lambda[u, u] = 1$ for all $u \in D$, and write $\Lambda = \{\lambda[u, v]\}_{u,v\in D}$, $\Psi = \{\psi[u, v]\}_{u,v\in D}$. We refer to Λ as the *verification probability matrix* and to Ψ as the *fine matrix*.

We assume that the outcome is chosen according to the declared type and the agent enjoys the utility associated with this outcome, but he only gets paid/does not get fined if the lie is not detected. This is motivated by applications such as scheduling, where lie detection typically takes place *after* an assignment of jobs to machines has been determined, but *before* the payments are distributed. That is, under a mechanism (g, p) the expected utility of an agent with type u who reports v is

$$U_{(g,p)}(u, v) = u(g(v)) + \lambda[u, v]p(v) - (1 - \lambda[u, v])\psi[u, v].$$

Definition 4.1. Given an outcome space $\mathcal{O}$, a domain D, a verification probability matrix $\Lambda = \{\lambda[u, v]\}_{u,v\in D}$, a fine matrix $\Psi = \{\psi[u, v]\}_{u,v\in D}$, and a social choice function $f : D \to \mathcal{O}$, a mechanism (g, p) is said to (Λ, Ψ)-*implement* f if for every $u \in D$ there exists a $v \in D$ such that

$$g(v) = f(u) \quad (3)$$
$$U_{(g,p)}(u, v) \geq U_{(g,p)}(u, u') \quad \text{for each } u' \in D. \quad (4)$$

If Ψ is the all-zero matrix, we omit it from the notation, and say that (g, p) Λ-*implements* f.

[3]One may feel that fines are not a necessary component of a probabilistic verification model; indeed, a simpler model with no fines may be adequate for many scenarios. However, there are settings where fines are externally available, and if our goal is to design a mechanism that minimizes the center's expenses (we consider this problem in Section 5), ignoring the availability of fines would lead to suboptimal solutions.

In words, the expected utility of the agent when declaring a type v with $g(v) = f(u)$ must be at least as high as for any other declaration.

A (Λ, Ψ)-implementation is said to be *truthful* if $f = g$ and condition (4) holds with $v = u$, i.e.,

$$u(f(u)) + p(u) \geq u(f(u')) + \lambda[u,u']p(u') - (1 - \lambda[u,u'])\psi[u,u'] \quad \text{for all } u, u' \in D.$$

This condition can be rewritten as

$$\lambda[u,v] \cdot p(v) - p(u) \leq \hat{c}[u,v] \quad \text{for all } u, v \in D, \tag{5}$$

where $\hat{c}[u,v] = u(f(u)) - u(f(v)) + (1 - \lambda[u,v])\psi[u,v]$.

The definition of the reporting function $\phi_{(g,p)} : D \to D$ extends to mechanisms with probabilistic verification in a natural way: $\phi_{(g,p)}(u)$ outputs some type v satisfying conditions (3) and (4). The notion of individual rationality can be extended similarly: we say that a mechanism (g,p) that (Λ, Ψ)-implements a social choice function f is *individually rational* if for every $u \in D$ it holds that $U_{(g,p)}(u, \phi_{g,p}(u)) \geq 0$.

Our probabilistic model generalizes the partial verification model (and, hence, also the classic model): a misreport graph M can be simulated by setting $\lambda[u,v] = 1$, $\psi[u,v] = 0$ if $(u,v) \in M$ and $\lambda[u,v] = 0$, $\psi[u,v] = +\infty$ otherwise. We denote the resulting verification probability matrix by Λ^M and the fine matrix by Ψ^M. Note that in this construction the fine matrix Ψ^M cannot be replaced by the all-zero matrix, i.e., a mechanism that M-implements f does not necessarily Λ^M-implement it. Indeed, in the absence of fines, if $u(g(v)) > u(g(u')) + p(u')$ for some $v \notin M(u)$ and all $u' \in M(u)$, an agent of type u would prefer to report v, even if he knows for sure that he will be denied payment. However, there is an interesting special case of the probabilistic verification model where fines are not necessary. Namely, suppose that all outcomes are associated with tasks, so that the agent incurs a cost for each outcome, i.e., $v(o) \leq 0$ for all $o \in \mathcal{O}$ and all $v \in D$. Let M be a misreport correspondence, and let (g,p) be an individually rational mechanism that M-implements some function f; we can assume that $p(v) \geq 0$ for all $v \in D$. Then (g,p) is an individually rational Λ^M-implementation of f. Indeed, if an agent of type u reports a type $v \notin M(u)$, he will be detected and his utility will be $u(g(v)) \leq 0$, whereas if he reports $\phi_{(g,p)}(u)$, the individual rationality of (g,p) guarantees him a non-negative utility. We will discuss this setting in more detail in Section 5.2.

4.1. Characterization of Truthfully Implementable Rules

It is natural to ask which social choice functions are truthfully implementable in our model. For the setting without verification, the answer is given by the classic result of Rochet [1987]. It turns out that a natural adaptation of that result works in our case as well.

Recall that Rochet's result is presented in terms of a complete directed graph G_f whose vertex set is the domain D, and the weight of the directed edge (u,v) is given by $c[u,v] = u(f(u)) - u(f(v))$. Rochet [1987] shows (see also [Vohra 2011]) that in the absence of verification, truthful implementability of f reduces to a combinatorial property of the graph G_f.

THEOREM 4.2 ([ROCHET 1987; VOHRA 2011]). *A social choice function f is truthfully implementable if and only if G_f has no negative cycle.*

We will now show that a similar characterization can be obtained for probabilistic verification under some mild conditions on the domain D and the matrix Λ. Namely, we will assume that (1) D is *bounded*, i.e., $-C < v(o) < C$ for some sufficiently large constant $C > 0$ and all $v \in D$, $o \in \mathcal{O}$, and (2) all elements of Λ are either equal to 1 or are bounded away from 1, i.e., there exists an $\varepsilon > 0$ such that for all $u, v \in D$ the inequality $\lambda[u,v] < 1$ implies $\lambda[u,v] \leq 1-\varepsilon$. Note that both of these restrictions trivially hold if the domain D is finite.

We will modify Rochet's construction as follows. Given a social choice function f and a probability verification matrix $\Lambda = \{\lambda[u,v]\}_{u,v\in D}$, we construct a graph G_f^Λ from G_f by setting edge weight $c^\Lambda[u,v] = +\infty$ for all edges (u,v) with $\lambda[u,v] < 1$ and $c^\Lambda[u,v] = c[u,v]$ otherwise. Now, the set of social choice functions that are (Λ,Ψ)-implementable can be characterized as follows.

THEOREM 4.3. *Suppose that D is bounded and all entries of Λ are either equal to 1 or are bounded away from 1. Then a social choice function f is truthfully (Λ,Ψ)-implementable if and only if G_f^Λ has no negative cycle.*

PROOF. Suppose that G_f^Λ has a negative cycle, and assume for the sake of contradiction that f admits a truthful (Λ,Ψ)-implementation. Then there is a payment vector $\{p(u)\}_{u\in D}$ that satisfies the inequalities (5) for all $u,v \in D$. All edges (u,v) in a negative cycle of G_f^Λ have $\lambda[u,v] = 1$, $c^\Lambda[u,v] = \hat{c}[u,v]$. Therefore, when we add the inequalities (5) along the negative cycle, the resulting inequality is of the form $0 \leq W$, where W is the (negative) weight of the cycle, a contradiction.

For the "if" direction, since G_f^Λ has no negative cycle, Theorem 4.2 implies that there are payments $\{p'(u)\}_{u\in D}$ satisfying all inequalities without probabilistic verification or fines: $p'(v) - p'(u) \leq c^\Lambda[u,v]$ for all $u,v \in D$ such that $\lambda[u,v] = 1$. Further, since D is bounded and the outcome space $\mathcal{O}$ is finite, we can assume that all payments are bounded, i.e., $-C' < p'(u) < C'$ for some $C' > 0$ and all $u \in D$; indeed, the shortest path-based construction of [Vohra 2011] would produce a payment vector with this property. We will now modify p' to construct a truthful payment vector.

Set

$$\begin{aligned} L &= \sup\{1/(1-\lambda[u,v]) \mid \lambda[u,v] < 1\}, \\ \Delta &= \sup\{\lambda[u,v]p'(v) - c[u,v] - p'(u) \mid \lambda[u,v] < 1\}; \end{aligned}$$

our assumptions on D and Λ ensure that both L and Δ are finite. We claim that the mechanism (f,p), where $p(u) = p'(u) + \Delta L$ for each $u \in D$, is Λ-truthful (and, hence, (Λ,Ψ)-truthful for any fine matrix Ψ, since all fines are non-negative).

Indeed, by construction, for each edge (u,v) with $\lambda[u,v] < 1$, we have $\lambda[u,v]p'(v) - c[u,v] - p'(u) \leq \Delta$ and $(1-\lambda[u,v])L \geq 1$. Therefore,

$$\begin{aligned} \lambda[u,v]p(v) - p(u) &= \lambda[u,v](p'(v) + \Delta L) - (p'(u) + \Delta L) \\ &= (\lambda[u,v]p'(v) - p'(u)) + (\lambda[u,v]-1)\Delta L \leq (c[u,v] + \Delta) - \Delta \leq \hat{c}[u,v]. \end{aligned}$$

On the other hand, for edges with $\lambda[u,v] = 1$ we get

$$p(v) - p(u) = (p'(v) + \Delta L) - (p'(v) + \Delta L) = p'(v) - p'(u) \leq c[u,v] = \hat{c}[u,v].$$

Thus, $\{p(u)\}_{u\in D}$ satisfies all inequalities (5), i.e., it truthfully (Λ,Ψ)-implements f. □

An interesting corollary of Theorem 4.3 is that if all probabilities are less than 1, *any* social choice function becomes implementable. Intuitively, this is not very surprising: if the payments are large and the agent has a non-zero risk of not receiving them whenever he lies, he prefers not to lie. Note also that the class of (Λ,Ψ)-implementable social choice functions does not depend on Ψ. In particular, any social choice function that can be implemented for some Λ and Ψ can be Λ-implemented. Intuitively, this is because the mechanism can simulate the fines through payments.

5. OPTIMAL PAYMENT COMPUTATION

In mechanism design, payments provide a useful tool to incentivize desirable behavior. However, they are costly to the center, and thus we are often interested in finding mechanisms that implement a given social choice function in the cheapest possible way. Of course, this question is only meaningful if we have to provide some welfare guarantees to the agent, as otherwise the center can always improve its profits by adding a large negative constant to all payments, i.e., charging the agents for participation. The standard way of handling this issue is to require individual rationality, as defined in Section 2. Another issue is that two payment vectors may be incomparable: if (f, p) and (f, q) are two truthful mechanisms with $p(u) = 3$, $p(v) = 1$ and $q(u) = 0$, $q(v) = 7$, which one should we select? Thus, we need to define our optimization objective more carefully.

The standard approach to this problem is to assume that, in addition to the outcome space $\mathcal{O}$, the domain D, the matrices Λ and Ψ, and the function f that we want to implement, we are given a probability distribution $\Pi = \{\pi_u\}_{u \in D}$ over D, and the agent draws his type from D according to Π; we can assume without loss of generality that Π assigns non-zero probability to all elements of D, i.e., $\pi_u > 0$ for all $u \in D$. Then, our aim is to minimize the expected payment subject to the individual rationality constraint. In other words, we aim to pick a mechanism (g, p) so as to minimize $\sum_{u \in D} \pi_u \lambda[u, \phi_{(g,p)}(u)] p(\phi_{(g,p)}(u))$. We will refer to this problem as $\textsc{Exp-Ver}(\mathcal{O}, D, \Lambda, \Psi, \Pi, f)$. A careful reader will notice that we do not treat the fines paid by the agent as the center's profits, but rather we assume that the fines are paid to a third party; this is consistent with our interpretation that fines are external to the mechanism. Alternatively, we may want to minimize the amount we have to pay in the worst-case scenario, i.e., $\max_{u \in D} p(\phi_{(g,p)}(u))$ (note that by the definition of $\phi_{(g,p)}$ the agent reports $\phi_{(g,p)}(u)$ with probability at least π_u, i.e., we will actually pay $p(\phi_{(g,p)}(u))$ with non-zero probability). This problem will be referred to as $\textsc{Max-Ver}(\mathcal{O}, D, \Lambda, \Psi, f)$; note that there is no dependence on Π in this definition. Both of our optimization problems can also be formulated in the partial verification model; these variants of our problems will be denoted by $\textsc{Exp-Ver}(\mathcal{O}, D, M, \Pi, f)$ and $\textsc{Max-Ver}(\mathcal{O}, D, M, f)$, respectively.

In the rest of this section, we investigate these optimization problems for the probabilistic verification model, first for truthful implementations and then for the general case. From now on, we will assume that the domain D is finite, and write $n = |D|$.

5.1. Truthful Implementation

Interestingly, it turns out that for truthful implementation the choice of the optimization objective is immaterial: for any truthfully (Λ, Ψ)-implementable social choice function f, the set $P_f(\Lambda, \Psi)$ of all payment vectors such that (f, p) is an individually rational (Λ, Ψ)-implementation of f has a minimum element. That is, there exists a unique payment vector $p^* \in P_f(\Lambda, \Psi)$ such that $p^*(v) \leq p(v)$ for any $v \in D$ and any $p \in P_f(\Lambda, \Psi)$. For the setting without verification, this result follows easily from the construction in [Vohra 2011]. We will now show how to extend it to our setting.

Recall that the truthfulness constraints (5) contain one inequality for each pair $(u, v) \in D \times D$, $u \neq v$, so altogether $n(n-1)$ inequalities. The individual rationality requirement can be encoded by n additional constraints: we have

$$p(u) + u(f(u)) \geq 0 \quad \text{for all } u \in D. \tag{6}$$

The inequalities in (5) and (6) are linear in $p(u)$ and $p(v)$, i.e., the system composed of (5) and (6) is a linear feasibility problem with $n(n-1) + n = n^2$ constraints. We will now show that if the set $P_f(\Lambda, \Psi)$ of all payment vectors satisfying conditions (5) and (6) is non-empty, then it contains a minimum element.

PROPOSITION 5.1. *If $P_f(\Lambda, \Psi) \neq \emptyset$, there exists a unique vector p^* such that $p^*(u) \leq p(u)$ for all $p \in P_f(\Lambda, \Psi)$ and all $u \in D$.*

PROOF. It is not hard to see that if p and q are two payment vectors in $P_f(\Lambda, \Psi)$, then the entry-wise minimum of p and q, $\min(p, q)$, is also a vector in $P_f(\Lambda, \Psi)$. Indeed, fix a type $u \in D$ and assume without loss of generality that $\min\{p(u), q(u)\} = p(u)$. Then we have

$$\lambda[u, v] \cdot \min\{p(v), q(v)\} - \min\{p(u), q(u)\} \leq \lambda[u, v] \cdot p(v) - p(u) \leq \hat{c}[u, v].$$

Also, we clearly have $\min\{p(u), q(u)\} \geq -u(f(u))$ for all $u \in D$. To complete the proof, it suffices to observe that for an arbitrary p the set $\{q \in P_f(\Lambda, \Psi) \mid q(u) \leq p(u) \text{ for all } u \in D\}$ is compact. $\square$

By combining the linear feasibility program (5)–(6) with the goal function $\min \sum_{u \in D} \pi_u p(u)$, we can find the optimal payment vector p^* in polynomial time. In fact, we can strengthen this result by observing that each constraint of our linear program contains at most two variables. Megiddo [1983], and, subsequently, Cohen and Megiddo [1994] show that for such linear programs a feasible solution can be found in strongly polynomial time; the algorithm in the latter paper can be used to find a lexicographically optimal solution. For our setting, the results of [Cohen and Megiddo 1994] imply an algorithm with running time $O(n^5 \log n)$. Moreover, a class of linear programs that is essentially equivalent to ours appears in the context of single-player discounted payoff games. Andersson [2006] provides a $O(n^4 \log n)$ algorithm for solving such linear programs; his procedure can be adapted to our case as well.

5.2. Optimal Non-truthful Implementation

As demonstrated by Example 2.2, non-truthful implementation may be less costly for the center than a truthful one. However, we will now argue that such implementations may be hard to find, even if we limit ourselves to partial implementation with a misreport graph that is acyclic and has maximum outdegree 2.

THEOREM 5.2. *Both* EXP-VER$(\mathcal{O}, D, M, \Pi, f)$ *and* MAX-VER$(\mathcal{O}, D, M, f)$ *are* NP-*hard. This holds even if $|\mathcal{O}| = 2$, M is acyclic and its maximum outdegree is 2.*

The proof of Theorem 5.2 can be modified to show that both of our problems remain hard for Λ-implementation: it suffices to set $\Lambda = \Lambda^M$ and observe that all types assign negative utilities to all outcomes, so all fines can be set to zero (see the discussion that precedes Section 4.1).

We remark that our proof can be used to strengthen a hardness result in [Auletta et al. 2011]. Specifically, Auletta et al. show that it is NP-hard to decide whether a given social choice function can be (non-truthfully) M-implemented without payments, even if $|\mathcal{O}| = 2$ and M is an acyclic graph with maximum outdegree 3. A simple modification of the proof of Theorem 5.2 allows us to extend this hardness result to misreport correspondences with maximum outdegree 2.

COROLLARY 5.3. *Given a social choice function $f : D \to \{F, T\}$ and a misreport graph M, it is* NP-*complete to decide whether f is implementable without payments. This holds even if M is acyclic and its maximum outdegree is 2.*

In light of Theorem 5.2, we will now consider a special case of our optimization problem, where the agent has non-positive valuations for all outcomes, and the elements of D can be ordered as $u_1, \ldots, u_n$ so that $\lambda[u, v] = 0$ unless $u = u_i$, $v = u_{i+1}$ for some

$i = 1, \ldots, n-1$; in other words, the graph M^Λ with the vertex set D and edge set $\{(u, v) \mid \lambda[u, v] > 0\}$ is a directed path. We will now show that for this setting both of our optimization problems admit an efficient algorithm.

THEOREM 5.4. *Both* EXP-VER$(\mathcal{O}, D, \Lambda, \Psi, \Pi, f)$ *and* MAX-VER$(\mathcal{O}, D, \Lambda, \Psi, f)$ *are polynomial-time solvable as long as M^Λ is a directed path and $v(o) \leq 0$ for all $v \in D$, $o \in \mathcal{O}$.*

PROOF. We denote by $[i : j]$ the set of consecutive integers $\{i, i+1, ..., j\}$ and assume that $[i : j]$ is empty when $i > j$. Without loss of generality, we assume that $D = [1 : n]$ and that M^Λ consists of directed edges $(k, k+1)$ for $k = 1, ..., n-1$. To avoid confusion, we denote by $v_i(o)$ the valuation of the agent for outcome $o \in \mathcal{O}$ when his true type is $i \in D$. For the sake of simplicity of the exposition, we assume that the fine matrix is an all-zero matrix and that $\pi_i = 1/n$ for each type $i \in D$ (so, we omit Ψ and Π from the notation). However, it is straightforward to adapt our proof to the case of general values of Ψ and Π. We begin with some definitions.

We denote by $C(f)$ the set of social choice functions g such that $f(\ell) \in \{g(\ell), g(\ell+1)\}$ and $g(n) = f(n)$. Clearly, a social choice function not belonging to $C(f)$ does not Λ-implement f. Consider the class $\mathcal{BR}$ of graphs with the types of D as nodes and one edge between any two consecutive types k and $k+1$ with direction either from k to $k+1$ (*right edge*) or from $k+1$ to k (*left edge*).

The graphs in $\mathcal{BR}$ can be used to model the incentives of the agent given a social choice function g and payments p. Specifically, we say that a graph G of $\mathcal{BR}$ is a *best response graph* for the pair (g, p) when for each right edge $(k, k+1)$ (resp., left edge $(k+1, k)$) of G, reporting $k+1$ (resp., k) is a best response of the agent when his true type is k. We remark that any possible best response of the agent can be modeled by a graph in $\mathcal{BR}$: indeed, due to the individual rationality condition and the assumption of non-positive valuations, declaring a type different than k or $k+1$ has negative utility for the agent when his true type is k. We say that (g, p) *implements f with best response graph* G if (g, p) is individually rational, G is a best response graph for (g, p), $g(n) = f(n)$, $g(k+1) = f(k)$ for each right edge $(k, k+1)$ of G, and $g(k) = f(k)$ for each left edge $(k+1, k)$. We say that g *implements f with a best response graph G* if there is a payment function p such that (g, p) implements f with best response graph G.

Given a graph $G \in \mathcal{BR}$, the class of social choice functions $C(f, G)$ consists of functions $g \in C(f)$ such that $g(n) = f(n)$, $g(k+1) = f(k)$ for each right edge $(k, k+1)$ of G, and $g(k) = f(k)$ for each left edge $(k+1, k)$. Observe that if there is a payment function p such that G is a best response graph for (g, p), then (g, p) implements f with best response graph G. Also, a social choice function not belonging to $C(f, G)$ does not implement f with a best response graph G. Furthermore, it is not hard to see that $\bigcup_{G \in \mathcal{BR}} C(f, G) = C(f)$.

After these initial definitions, we can give a roadmap of the proof. First, we show that for each graph $G \in \mathcal{BR}$ and each social choice function $g \in C(f, G)$, g implements f with best response graph G. In other words, we prove that $C(f, G)$ consists of the social choice functions that implement f with best response graph G and, consequently, $C(f)$ consists of the social choice functions that Λ-implement f. Further, we argue that the set of all payment functions p such that (g, p) implements f with best response graph G contains a unique coordinate-wise minimal element $p^*_{g,G}$. Finally, we show how to find the optimal solution to MAX-VER$(\mathcal{O}, D, \Lambda, f)$ and EXP-VER$(\mathcal{O}, D, \Lambda, f)$ by optimizing over all possible best response graphs. This is done by a path computation in an appropriately defined network.

We proceed by presenting a simple representation of the graphs of $\mathcal{BR}$, which we will call the *hill decomposition*. Consider a connected non-trivial subgraph of a graph of $\mathcal{BR}$ defined on the consecutive types $[i : k]$. This subgraph is called a *hill* when it consists of left edges only and $i = 1$ (such a hill is represented as $(1, 1, k)$), or it consists of right edges only and $k = n$ (such a hill is represented as (i, n, n)), or there is a type j with $1 \le i < j < k \le n$ such that the edges between types t and $t + 1$ are right for $t \in [i : j - 1]$ and left for $t \in [j : k - 1]$ (such a hill is represented as (i, j, k)). Every graph $G \in \mathcal{BR}$ admits a unique hill decomposition, which can be constructed as follows. Let $S(G)$ be the set of types t in $[2 : n - 1]$ such that G contains a left edge $(t, t - 1)$ and a right edge $(t, t + 1)$. To obtain the hill decomposition of G, we simply replace each type t in $S(G)$ with two copies, and connect each of the two incident edges of t to a different copy.

Let G be a graph of $\mathcal{BR}$ and $h = (i, j, k)$ a hill in its hill decomposition. Abusing notation, we will say that a social choice function $g \in C(f)$ *belongs to* $C(f, h)$ if $g(\ell) = f(\ell - 1)$ for each type $\ell \in [i + 1 : j]$ and $g(\ell) = f(\ell)$ for each type $\ell \in [j : k - 1]$. Also, let h_1, h_2, ..., h_s be the consecutive hills in the hill decomposition of G; clearly, $C(f, G) = C(f, h_1) \cap ... \cap C(f, h_s)$.

We say that a pair of outcomes $O^h = (o_i, o_k)$ is *consistent with a hill* $h = (i, j, k)$ if there is a social choice function in $C(f, h)$ returning outcome o_i for type i and outcome o_k for type k. The definitions above imply that o_i is necessarily $f(1)$ when $i = j = 1$, but can be any outcome in $\mathcal{O}$ otherwise. Similarly, o_k is necessarily $f(n)$ when $k = n$, but can be any outcome in $\mathcal{O}$ otherwise. Consider such a hill $h = (i, j, k)$ and a pair of outcomes $O^h = (o_i, o_k)$ consistent with h. Denote by $\mathsf{HCon}(h, O^h)$ the following set of constraints on the payment entries $p(i)$, $p(i + 1)$, ..., $p(k)$.

$$p(i+1) \ \ge\ \frac{p(i) + v_i(o_i) - v_i(f(i))}{\lambda[i, i+1]}, \text{ if } i < j \tag{7}$$

$$p(\ell) \ \ge\ \frac{p(\ell-1) + v_{\ell-1}(f(\ell-2)) - v_{\ell-1}(f(\ell-1))}{\lambda[\ell-1, \ell]}, \text{ for } \ell \in [i+2 : j] \tag{8}$$

$$p(\ell) \ \ge\ \lambda[\ell, \ell+1]p(\ell+1) + v_\ell(f(\ell+1)) - v_\ell(f(\ell)), \text{ for } \ell \in [j : k-2] \tag{9}$$

$$p(k-1) \ \ge\ \lambda[k-1, k]p(k) + v_{k-1}(o_k) - v_{k-1}(f(k-1)), \text{ if } j < k \tag{10}$$

$$p(\ell) \ \ge\ -\frac{v_{\ell-1}(f(\ell-1))}{\lambda[\ell-1, \ell]}, \text{ for } \ell \in [i+1 : j] \tag{11}$$

$$p(\ell) \ \ge\ -v_\ell(f(\ell)), \text{ for } \ell \in [j : k-1] \tag{12}$$

$$p(n) \ \ge\ -v_n(f(n)), \text{ if } k = n \tag{13}$$

Here, constraints (7) and (8) guarantee that, for each $\ell \in [i : j - 1]$ (if any), reporting type $\ell + 1$ is a best response for the agent when his true type is ℓ. To see why this is the case, consider a social choice function $g' \in C(f, h)$ with $g'(i) = o_i$ and $g'(k) = o_k$: by definition, it has $g'(\ell) = f(\ell - 1)$ for $\ell \in [i + 1 : j]$ and inequalities (7) and (8) can be rewritten as

$$p(\ell) + v_\ell(g'(\ell)) \le \lambda[\ell, \ell+1]p(\ell+1) + v_\ell(g'(\ell+1))$$

for $\ell \in [i : j-1]$. Similarly, constraints (9) and (10) guarantee that, for each $\ell \in [j : k-1]$ (if any), reporting type ℓ is a best response for the agent when his true type is ℓ. To see this, consider a social choice function g' that has $g'(\ell) = f(\ell)$ for $\ell \in [j : k - 1]$ (since $g' \in C(f, h)$). Then, inequalities (9) and (10) become

$$p(\ell) + v_\ell(g'(\ell)) \ge \lambda[\ell, \ell+1]p(\ell+1) + v_\ell(g'(\ell+1))$$

for $\ell \in [j : k-1]$. Constraints (11)–(12) ensure individual rationality for the agent when his true type is $\ell \in [i : k - 1]$. Finally, constraint (13) ensures individual rationality for the agent when his true type is n.

An argument similar to the one used in the proof of Proposition 5.1 shows that the set of constraints $\mathsf{HCon}(h, O^h)$ admits a solution in which the entries $p(i)$, ..., $p(k)$ are simultaneously minimized. In fact, this solution can computed in time $O(|j-i+1|)$. To do so, we will process the types one by one, by moving from two sides to the top of the hill, i.e., in the order $i, i+1, \ldots, j-1, k, k-1, \ldots, j$. At each type ℓ, we look at all inequalities with $p(\ell)$ on the left hand side of $\geq$. Either the payment involved on the right hand side is decided already (inequalities (7)–(10)), or the right hand side is simply a constant (inequalities (11)–(13)). This allows us to decide the payment at type ℓ by taking the maximum of the right hand side of these inequalities.

Let us denote by $p^*_{h,O^h}(i)$, ..., $p^*_{h,O^h}(k)$ the minimum values of the entries $p(i)$, ..., $p(k)$. Then p^*_{h,O^h} minimizes also the following two functions of $p(i), ..., p(k)$:

$$\mathsf{nhmax}(h, O^h, p) = \begin{cases} \max_{\ell=i+1}^{k-1} p(\ell), & \text{if } k \neq n \\ \max_{\ell=i+1}^{n} p(\ell), & \text{otherwise} \end{cases}$$

and

$$\mathsf{nhexp}(h, O^h, p) = \begin{cases} \frac{1}{n}\left(\sum_{\ell=i}^{j-1} \lambda[\ell,\ell+1]p(\ell+1) + \sum_{\ell=j}^{k-1} p(\ell)\right), & \text{if } k \neq n \\ \frac{1}{n}\left(\sum_{\ell=i}^{j-1} \lambda[\ell,\ell+1]p(\ell+1) + \sum_{\ell=j}^{n} p(\ell)\right), & \text{otherwise} \end{cases}$$

Observe that, for each type $t \in \{i,k\} \setminus \{1,n\}$ (if any), the entry $p(t)$ is lower-bounded only by the constraint $p(t) \geq 0$ and, hence, $p^*_{h,O^h}(t) = 0$.

Now, consider a graph $G \in \mathcal{BR}$ whose unique representation consists of s consecutive hills $h_1, h_2, ..., h_s$. Also, let g be a social choice function of $C(f,G)$ returning the pair of outcomes $O^h = \{o_i, o_j\}$ on input types i and k for each hill $h = (i,j,k)$ in $\{h_1, ..., h_s\}$. Define the payment function $p^*_{g,G}$ as p^*_{h,O^h} on the types of each hill $h \in \{h_1, ..., h_s\}$. Note that this definition is consistent: by the observation above, if two different hills $h, h' \in \{h_1, ..., h_s\}$ share a type t, then this type has $p^*_{h,O^h}(t) = p^*_{h',O^{h'}}(t) = 0$. Furthermore, it guarantees that $p^*_{g,G}$ has all its entries simultaneously minimized among all payment functions p such that the union of the sets of constraints $\mathsf{HCon}(h, O^h)$ for $h \in \{h_1, ..., h_s\}$ is satisfied; this union consists of the conditions ensuring that (g,p) implements f with best response graph G. Hence, under the same constraint, $p^*_{g,G}$ minimizes the function $\max_{h\in\{h_1,...,h_s\}} \mathsf{nhmax}(h, O^h, p)$ and $\sum_{h\in\{h_1,...,h_s\}} \mathsf{nhexp}(h, O^h, p)$ as well. It is not hard to see that the former is equal to the maximum payment to the agent while the latter is the expected payment to the agent.

We are ready to present our construction. We build a network consisting of $2 + nm$ nodes partitioned into $n+2$ levels $0, 1, ..., n+1$. Level 0 has node w_{start} and level $n+1$ has node w_{end}. For $\ell = 1, ..., n$, level ℓ contains node $w_{\ell,o}$ for each $o \in \mathcal{O}$. There are edges of cost 0 from w_{start} to all nodes at level 1, and from all nodes at level n to w_{end}. Also, for every possible hill $h = (i,j,k)$ and every pair of outcomes $O^h = (o_i, o_k)$ that is consistent with h, there is a directed edge from node w_{i,o_i} to node w_{k,o_k}. This construction guarantees that every path from node w_{start} to node w_{end} in this network corresponds to a pair (G,g), where G is a graph in $\mathcal{BR}$ and g is a social choice function in $C(f,G)$. Furthermore, observe that the number of nodes and edges in this network is polynomial in n and $|\mathcal{O}|$. To complete the construction, it remains to define the cost of the edges that correspond to hills.

In the network used to solve MAX-VER$(\mathcal{O}, D, \Lambda, f)$, the cost of an edge corresponding to a hill h and a pair of outcomes O^h consistent with h is $\mathsf{nhmax}(h, O^h, p^*_{h,O^h})$. This guarantees that the maximum cost among the edges in a path from w_{start} to w_{end} that corresponds to the pair (g,G) is equal to the maximum payment to the agent in the implementation $(g, p^*_{g,G})$ of f. In the network used to solve $\textsc{Exp-Ver}(\mathcal{O}, D, \Lambda, f)$, the cost of the same edge is $\mathsf{nhexp}(h, O^h, p^*_{h,O^h})$. This guarantees that the total cost of the edges in a path from w_{start} to w_{end} that corresponds to the pair (g, G) is equal to the expected payment to the agent in the implementation $(g, p^*_{g,G})$ of f. In both cases, the path corresponding to the optimal solution of $\textsc{Max-Ver}(\mathcal{O}, D, \Lambda, f)$ or $\textsc{Exp-Ver}(\mathcal{O}, D, \Lambda, f)$ can be found in time polynomial in n and m by a simple shortest path computation. $\square$

One may think that hardness of optimal non-truthful implementation stems from the fact that we have to guess the social choice function g, in addition to the payment function p, and, consequently, that finding the optimal payment vector p that turns a given social choice function g into an individually rational mechanism that Λ-implements f is easy. The results of Section 5.1 confirm this intuition for the case $g = f$, and the proof of Theorem 5.4 effectively shows that finding an optimal p for a given g is easy if M^Λ is a directed path. However, in general, finding an optimal implementation may be hard even if both f and g are fixed. We will now prove this for M-implementation; the proof generalizes easily to Λ-implementation.

THEOREM 5.5. *Given a space of outcomes $\mathcal{O}$, a finite domain D, a misreport graph M, a distribution Π on D, a pair of social choice functions $f, g : D \to \mathcal{O}$, and a parameter t, it is* NP*-hard to decide whether there exists a payment function p such that (g, p) is an individually rational M-implementation of f and the center's expected payment under (g, p) is at most t. This holds even if $|\mathcal{O}| = 2$ and M is acyclic and its maximum outdegree is* 3.

6. CONCLUSION

We have introduced a model for mechanism design with probabilistic verification. Our work suggests several interesting open problems. First, the negative results in Section 3 are limited to truthful implementation; can we expect to circumvent them by resorting to non-truthful implementation? Also, there are interesting variants of the probabilistic verification problem, which we have not fully explored. For instance, what happens if verification happens prior to outcome selection, and if the agent is caught lying, the mechanism implements the outcome that corresponds to his true type? How would our results change if we were to assume that the center can pocket the fine that the agent pays when he is caught lying? What if the fine the agent pays can only depend on his true type or, conversely, his declared type? More broadly, what if the center has some verification "budget" and can use it to determine the verification probabilities?

We remark that by focusing on a single agent we implicitly assume that the verification probabilities do not depend on other agents' type declarations. (Note that this is also the case for the partial verification model of [Green and Laffont 1986; Auletta et al. 2011].) While this assumption is reasonable for some applications (such as the voting scenario described in the introduction), there are settings where it may fail to hold: for instance, in the job scheduling example a cheating machine cannot be detected unless it is assigned some load, and its load may depend on the speeds declared by other machines. It would be interesting to extend our model so as to account for this possibility.

On a more technical level, the proof of Theorem 5.4 works under the assumption that valuations are non-positive. For the case where this assumption is removed, we have a (significantly more complicated) extension of our algorithm that computes the optimal non-truthful implementation in time polynomial in n and $2^{|\mathcal{O}|}$. It is not clear whether the exponential dependence on $|\mathcal{O}|$ is necessary here. Of course, extending our positive result to graphs of maximum outdegree 1 is also an interesting challenge.

REFERENCES

ANDERSSON, D. 2006. Improved combinatorial algorithms for discounted payoff games. M.S. thesis, Uppsala University.

ARCHER, A. AND KLEINBERG, R. 2008. Truthful germs are contagious: a local to global characterization of truthfulness. In *ACM EC'08*. 21–30.

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.

AULETTA, V., PRISCO, R. D., PENNA, P., AND PERSIANO, G. 2009. The power of verification for one-parameter agents. *Journal of Computer and System Sciences 75,* 3, 190–211.

AULETTA, V., PRISCO, R. D., PENNA, P., PERSIANO, G., AND VENTRE, C. 2006. New constructions of mechanisms with verification. In *ICALP'06*. 596–607.

COHEN, E. AND MEGIDDO, N. 1994. Improved algorithms for linear inequalities with two variables per inequality. *SIAM Journal of Computing 23,* 6, 1313–1347.

GIBBARD, A. 1973. Manipulation of voting schemes. *Econometrica 41,* 4, 587–601.

GREEN, J. AND LAFFONT, J.-J. 1986. Partially verifiable information and mechanism design. *Review of Economic Studies 53*, 447–456.

GUO, M. AND CONITZER, V. 2010. Computationally feasible automated mechanism design: General approach and case studies. In *AAAI'10*. 1676–1679.

KRYSTA, P. AND VENTRE, C. 2010. Combinatorial auctions with verification are tractable. In *ESA'10*. 39–50.

MEGIDDO, N. 1983. Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal of Computing 12,* 2, 347–353.

MYERSON, R. 1981. Optimal auction design. *Mathematics of Operations Research 6,* 1, 58–73.

NISAN, N. 2007. Introduction to mechanism design. In *Algorithmic Game Theory*, N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, Eds. Chapter 9, 209–242.

NISAN, N. AND RONEN, A. 2001. Algorithmic mechanism design. *Games and Economic Behavior 35*, 166–196.

PENNA, P. AND VENTRE, C. 2008. Collusion-resistant mechanisms with verification yielding optimal solutions. In *ESA'08*. 708–719.

PENNA, P. AND VENTRE, C. 2009. Optimal collusion-resistant mechanisms with verification. In *EC'09*. 147–156.

ROBERTS, K. 1979. The characterization of implementable choice rules. In *Aggregation and Revelation of Preferences. Papers presented at the 1st European Summer Workshop of the Econometric Society*, J.-J. Laffont, Ed. 321–349.

ROCHET, J.-C. 1987. A condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics 16*, 191–200.

SAKS, M. AND YU, L. 2005. Weak monotonicity suffices for truthfulness on convex domains. In *ACM EC'05*. 286–293.

SATTERTHWAITE, M. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory 10,* 2, 187–217.

SINGH, N. AND WITTMAN, D. 2001. Implementation with partial verification. *Review of Economic Design 6*, 63–84.

VOHRA, R. 2011. *Mechanism Design: A Linear Programming Approach*. Cambridge University Press.

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