Alternatives to Truthfulness are Hard to Recognize*

Vincenzo Auletta[1], Paolo Penna[1], Giuseppe Persiano[1], and Carmine Ventre[2]**

[1] Dipartimento di Informatica ed Applicazioni, Università di Salerno, Italy.
E-mail: {auletta,penna,giuper}@dia.unisa.it
[2] Computer Science Department, University of Liverpool, UK.
E-mail: Carmine.Ventre@liverpool.ac.uk

Abstract. The central question in mechanism design is how to implement a given social choice function. One of the most studied concepts is that of *truthful* implementations in which truth-telling is always the best response of the players. The Revelation Principle says that one can focus on truthful implementations without loss of generality (if there is no truthful implementation then there is no implementation at all). Green and Laffont [1] showed that, in the scenario in which players' responses can be *partially verified*, the revelation principle holds only in some particular cases.
When the Revelation Principle does not hold, non-truthful implementations become interesting since they might be the only way to implement a social choice function of interest. In this work we show that, although non-truthful implementations may exist, they are hard to find. Namely, it is NP-hard to decide if a given social choice function can be implemented in a non-truthful manner, or even if it can be implemented at all. This is in contrast to the fact that truthful implementability can be recognized efficiently, even when partial verification of the agents is allowed. Our results also show that there is no "simple" characterization of those social choice functions for which it is worth looking for non-truthful implementations.

1 Introduction

Social choice theory deals with the fact that individuals (agents) have different preferences over the set of possible alternatives or outcomes. A social choice function maps these preferences into a particular outcome, which is not necessarily the one preferred by the agents. The main difficulty in implementing a social choice function stems from the fact that agents can *misreport* their preferences. Intuitively speaking, a social choice function can be implemented if there is a method for selecting the desired outcome which cannot be manipulated by

* Research funded by the EU through IP AEOLUS.
** The author is also supported by DFG-funded project "Algorithmic Tools for Games with Applications to E-Commerce and Networks" within Emmy Noether Program.

rational agents. By 'desired outcome' we mean the one specified by the social choice function applied to the *true* agents' preferences.

More precisely, each agent has a *type* which specifies the utility he derives if some outcome is selected. When agents are also endowed with payments, we consider agents with quasi linear utility: the type specifies the gross utility and the agent's utility is the sum of gross utility and payment received. In either case, a rational agent reports a type so to maximize his own utility and the reported type must belong to a *domain* consisting of all possible types. In the case of *partially verifiable* information, the true type of an agent further restricts the set of types that he can possibly report [1].

One of the most studied solution concepts is that of *truthful* implementations in which agents always maximize their utilities by truthfully reporting their types. The *Revelation Principle* says that one can focus on truthful implementations without loss of generality: A social choice function is implementable if and only if it has a truthful implementation. Green and Laffont [1] showed that, in the case of partially verifiable information, the Revelation Principle holds only in some particular cases. When the Revelation Principle does not hold, non-truthful implementations become interesting since they might be the only way to implement a social choice function of interest. Although a non-truthful implementation may induce some agent to misreport his type, given that he reports the type maximizing his utility, it is still possible to compute the desired outcome "indirectly". Singh and Wittman [3] observed that the Revelation Principle fails in several interesting cases and show sufficient conditions for the existence of non-truthful implementations.

1.1 Our contribution

In this work, we study the case in which the declaration of an agent can be partially verified. We adopt the model of Green and Laffont [1] in which the ability to partially verify the declaration of an agent is encoded by a *correspondence* function M: $M(t)$ is the set of the possible declarations of an agent of type t. Green and Laffont [1] characterized the correspondences for which the Revelation Principle holds; that is, correspondences M for which a social choice function is either truthfully implementable or not implementable at all.

We show that although non-truthful implementations may exist, they are hard to find. Namely, it is NP-hard to decide if a given social choice function can be implemented for a given correspondence in a non-truthful manner. This is in contrast to the fact that it is possible to efficiently decide whether a social choice function can be truthfully implemented for a given correspondence. Our results show that there is no "simple" characterization of those social choice functions that violate the Revelation Principle. These are the social choice functions for which it is worth looking for non-truthful implementations since this might be the only way to implement them.

We prove these negative results for a very restricted scenario in which we have only one agent and at most two possible outcomes, and the given function does not have truthful implementations. We give hardness proofs both for the

case in which payments are not allowed and the case in which payments are allowed and the agent has quasi linear utility.

In general payments are intended as a tool for enlarging the class of social choice functions that can be implemented. We find that there is a rich class of correspondences for which it is NP-hard to decide if a social choice function can be implemented *without* payments, while for the same correspondences it is trivial to test truthful implementable with payments via the approach in [3]. Finally, we complement our negative results by showing a class of correspondences for which there is an efficient algorithm for deciding whether a social choice function can be implemented.

We note that the characterization of Green and Laffont [1] has no direct implication in our results. Indeed, the property characterizing the Revelation Principle can be tested efficiently. Moreover, when the Revelation Principle does not hold, we only know that there exists *some* social choice function which is only implemented in a non-truthful manner. Hence, we do not know if the social choice function of interest can be implemented or not. Note that this question can be answered efficiently when the Revelation Principle holds since testing the existence of truthful implementations is computationally easy.

Road map. We introduce the model with partial verification by Green and Laffont [1] in Section 2. The case with no payments is studied in Section 3. Section 4 presents our results for the case in which payments are allowed and the agent has quasi linear utility. We draw some conclusions in Section 5.

2 The Model

The model considered in this work is the one studied by Green and Laffont [1] who considered the so called principal-agent scenario. Here there are two players: the agent, who has a type t belonging to a domain D, and the principal who wants to compute a social choice function $f : D \to \mathcal{O}$, where $\mathcal{O}$ is the set of possible of outcomes. The quantity $t(X)$ denotes the *utility* that an agent of type t assigns to outcome $X \in \mathcal{O}$.

The agent observes his type $t \in D$ and then transmits some message $t' \in D$ to the principal. The principal applies the outcome function $g : D \to \mathcal{O}$ to t' and obtains outcome $X = g(t')$. We stress that the principal fixes the outcome function g in advance and then the agent *rationally* reports t' so to maximize his utility $t(g(t'))$. Even though the principal does not exactly know the type of the agent, it is reasonable to assume that some *partial* information on the type of the agent is available. Thus the agent is restricted to report a type t' in a set $M(t) \subseteq D$, which is specified by a *correspondence* function $M : D \to 2^D$. We will only consider correspondences $M(\cdot)$ for which truth-telling is always an option; that is, for all $t \in D$, $t \in M(t)$. Notice that the case in which the principal has no information (no verification is possible) corresponds to setting $M(t) = D$ for all t.

Definition 1 ([1]). *A mechanism (M, g) consists of a correspondence $M : D \to 2^D$ and an* outcome function *$g : D \to \mathcal{O}$. The outcome function g induces a*

best response rule $\phi_g : D \to D$ *defined by* $\phi_g(t) \in \arg\max_{t' \in M(t)}\{t(g(t'))\}$. *If* $t \in \arg\max_{t' \in M(t)}\{t(g(t'))\}$ *then we set* $\phi_g(t) = t$.

The correspondence M can be represented by a directed graph $\mathcal{G}_M$ (which we call the *correspondence graph*) defined as follows. Nodes of $\mathcal{G}_M$ are types in the domain D and an edge (t, t'), for $t \neq t'$, exists if and only if $t' \in M(t)$. We stress that the correspondence graph of M does not contain self-loops, even though we only consider correspondences M such that $t \in M(t)$ for all $t \in D$. We will often identify the correspondence M with its correspondence graph $\mathcal{G}_M$ and say, for example, that a correspondence is acyclic meaning that its correspondence graph is acyclic. Sometimes it is useful to consider a weighted version of graph $\mathcal{G}_M$. Specifically, for a function $g : D \to \mathcal{O}$, we define $\mathcal{G}_{M,g}$ to be the weighted version of graph $\mathcal{G}_M$ where edge (t, t') has weight $t(g(t)) - t(g(t'))$.

We study the class of M-implementable social choice functions $f : D \to \mathcal{O}$.

Definition 2 ([1]). *An outcome function* $g : D \to \mathcal{O}$ M-implements *social choice function* $f : D \to \mathcal{O}$ *if for all* $t \in D$ $g(\phi_g(t)) = f(t)$ *where* $\phi_g(\cdot)$ *is the best response rule induced by* g. *A social choice function* $f : D \to \mathcal{O}$ *is* M-implementable *if and only if there exists an outcome function* $g : D \to \mathcal{O}$ *that* M*-implements* f.

The social choice functions that can be truthfully M-implemented are of particular interest.

Definition 3 ([1]). *An outcome function* $g : D \to \mathcal{O}$ truthfully M-implements *social choice function* $f : D \to \mathcal{O}$ *if* g M*-implements* f *and* $\phi_g(t) = t$ *for all* $t \in D$. *A social choice function* $f : D \to \mathcal{O}$ *is* truthfully M-implementable *if and only if there exists an outcome function* $g : D \to \mathcal{O}$ *that truthfully* M*-implements* f.

The classical notions of *implementation* and of *truthful implementation* are obtained by setting $M(t) = D$ for all $t \in D$. Actually in this case the two notions of implementable social choice function and of truthfully implementable social choice function coincide due to the well-known revelation principle.

Theorem 1 (The Revelation Principle). *If no verification is possible (that is,* $M(t) = D$ *for all* $t \in D$*), a social choice function is implementable if and only if it is truthfully implementable.*

The Revelation Principle does not necessarily hold for the notion of M-implementation and of truthful M-implementation. Green and Laffont [1] indeed give a necessary and sufficient condition on M for the revelation principle to hold. More precisely, a correspondence M satisfies the *Nested Range Condition* if the following holds: for any $t_1, t_2, t_3 \in D$ if $t_2 \in M(t_1)$ and $t_3 \in M(t_2)$ then $t_3 \in M(t_1)$.

Theorem 2 (Green-Laffont [1]). *If* M *satisfies the NRC condition then a social choice function* f *is* M*-implementable if and only if* f *is* M*-truthfully implementable. If* M *does not satisfy the NRC condition then there exists an* M*-implementable social choice function* f *that is not truthfully* M*-implementable.*

Besides its conceptual beauty, the Revelation Principle can also be used in some cases to decide whether a given social choice function f is M-implementable for a given correspondence M. Indeed, if the Revelation Principle holds for correspondence M, the problem of deciding M-implementability is equivalent to the problem of deciding truthful M-implementability which, in turn, can be efficiently decided.

Theorem 3. *There exists an algorithm running in time polynomial in the size of the domain that, given a social choice function f and a correspondence M, decides whether f is truthfully M-implementable.*

Proof. To test truthful M-implementability of f we consider graph $\mathcal{G}_{M,f}$ where edge (t, t') has weight $t(f(t)) - t(f(t'))$. Then it is obvious that f is M-truthful implementable if and only if no edge of $\mathcal{G}_{M,f}$ has negative weight. □

3 Hardness of the Implementability problem

In this section we prove that the following problem is NP-hard.

Problem 1. The IMPLEMENTABILITY problem is defined as follows.

INPUT: domain D, outcome set $\mathcal{O}$, social choice function $f : D \to \mathcal{O}$ and correspondence M.

TASK: decide whether there exists an outcome function g that M-implements f.

The following lemma, whose proof is immediate, gives sufficient conditions for an outcome function g to M-implement social choice function f.

Lemma 1. *For outcomes $\mathcal{O} = \{T, F\}$, if the following conditions are satisfied for all $a \in D$ then outcome function g M-implements social choice function f.*

1. *If $f(a) = T$ and $a(T) < a(F)$ then, for all $v \in M(a)$, we have $g(v) = T$.*
2. *If $f(a) = F$ and $a(T) < a(F)$ then, there exists $v \in M(a)$ such that $g(v) = F$.*
3. *If $f(a) = T$ and $a(T) > a(F)$ then, there exists $v \in M(a)$ such that $g(v) = T$.*
4. *If $f(a) = F$ and $a(T) > a(F)$ then, for all $v \in M(a)$, we have $g(v) = F$.*

The reduction. We reduce from 3SAT. Let Φ a Boolean formula in 3-CNF over the variables $x_1, \cdots, x_n$ and let $C_1, \cdots, C_m$ be the clauses of Φ. We construct D, $\mathcal{O}$, M and $f : D \to \mathcal{O}$ such that f is M-implementable if and only if Φ is satisfiable. We set $\mathcal{O} = \{T, F\}$. We next construct a correspondence graph $\mathcal{G}_M$ representing M. We will use variable gadgets (one per variable) and clause gadgets (one per clause).

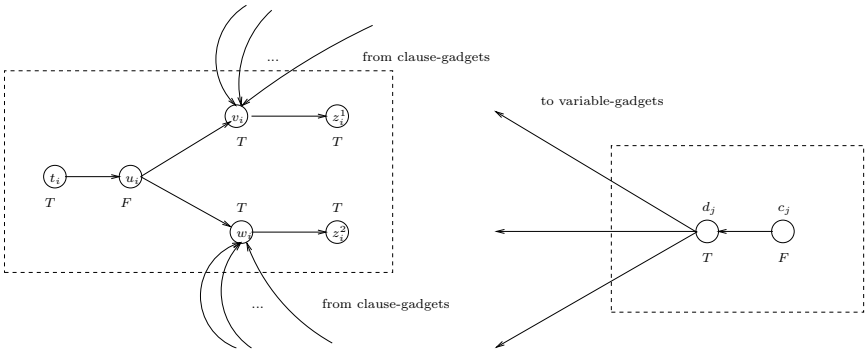
The variable gadget for the variable x_i is depicted in Figure 1(a). Each variable x_i of the formula Φ adds six new types to the domain D of the agent, namely, t_i, u_i, v_i, w_i, z_i^1 and z_i^2 satisfying the following relations:

$$t_i(F) > t_i(T), \tag{1}$$

$$u_i(F) > u_i(T), \tag{2}$$

$$v_i(T) > v_i(F), \tag{3}$$

$$w_i(T) > w_i(F). \tag{4}$$



(a) The variable gadget (b) The clause gadget

Fig. 1. Gadgets used in the reduction.

The labeling of the vertices defines the social choice function f; that is, $f(t_i) = T, f(v_i) = T, f(w_i) = T, f(z_i^1) = T, f(z_i^2) = T,$ and $f(u_i) = F$. Directed edges of the gadget describe the correspondence M (rather the correspondence graph). Thus, for example, $M(t_i) = \{t_i, u_i\}$ and $M(u_i) = \{u_i, v_i, w_i\}$. Nodes v_i and w_i have incoming edges from the clause gadgets. The role of these edges will be clear in the following.

We observe that (1) implies that the social choice function f is not truthfully M-implementable. Indeed t_i prefers outcome $F = f(u_i)$ to $T = f(t_i)$ and $u_i \in M(t_i)$. Moreover, by Lemma 1, for any outcome function g implementing f we must have $g(t_i) = g(u_i) = T$. On the other hand, since $f(u_i) = F$ it must be the case that any g that M-implements f assigns outcome F to *at least* one node in $M(u_i) \setminus \{u_i\}$. Intuitively, the fact that every outcome function g that M-implements f must assign F to at least one between v_i and w_i corresponds to assigning "false" to respectively literal x_i and $\bar{x}_i$.

The clause gadget for clause C_j of Φ is depicted in Figure 1(b). Each clause C_j adds types c_j and d_j to the domain D of the agent such that

$$c_j(T) > c_j(F), \tag{5}$$

$$d_j(T) > d_j(F). \tag{6}$$

As before the labeling defines the social choice function f and we have $f(d_j) = T$ and $f(c_j) = F$. Moreover, directed edges encode correspondence M. Besides the directed edge (c_j, d_j), the correspondence graph contains three edges directed from d_j towards the three variable gadgets corresponding to the variables appearing in the clause C_j. Specifically, if C_j contains the literal x_i then d_j has an outgoing edge to node v_i. If C_j contains the literal $\bar{x}_i$ then d_j has an outgoing edge to node w_i. Similarly to the variable gadget, we observe that (5) implies that for any g M-implementing f it must be $g(d_j) = F$. Therefore, for g to M-implement f it must be the case that, for at least one of the neighbors a of

d_j from a variable gadget, we have $g(a) = T$. We will see that this happens if and only if the formula Φ is satisfiable. This concludes the description of the reduction.

We next prove that the reduction is correct. Suppose that Φ is satisfiable, let τ be a satisfying truth assignment and let g be the outcome function defined as follows. For the i-th variable gadget we set $g(t_i) = g(u_i) = g(z_i^1) = g(z_i^2) = T$. Moreover, if x_i is true in τ, then we set $g(v_i) = T$ and $g(w_i) = F$; otherwise se set $g(v_i) = F$ and $g(w_i) = T$. For the j-th clause gadget, we set $g(d_j) = g(c_j) = F$.

Thus, to prove that the outcome function produced by our reduction M-implements f, it is sufficient to show for each type a the corresponding condition of Lemma 1 holds. We prove that conditions hold only for $a = u_i$ and $a = d_j$, the other cases being immediate. For u_i we have to verify that Condition 2 of Lemma 1 holds. Since τ is a truth assignment, for each i vertex u_i has a neighbor vertex for which the outcome function g gives F. For d_j we have to verify that Condition 3 of Lemma 1 holds. Since τ is a satisfying truth assignment, for each j there exists at least one literal of C_j that is true in τ; therefore, vertex d_j has a neighbor vertex for which the outcome function g gives T.

Conversely, consider an outcome function g which M-implements the social choice function f. This means that, for each clause C_j, d_j is connected to at least one node, call it a_j, from a variable gadget such that $g(a_j) = T$. Then the truth assignment that sets to true the literals corresponding to nodes $a_1, \cdots, a_m$ (and gives arbitrary truth value to the other variables) satisfies the formula.

The following theorem follows from the above discussion and from the observation that the reduction can be carried out in polynomial time and the graph we constructed is acyclic with maximum outdegree 3.

Theorem 4. *The* Implementability *Problem is NP-hard even for outcome sets of size* 2 *and acyclic correspondences of maximum outdegree* 3*.*

3.1 Corrrespondences with outdegree 1

In this section, we study correspondences of outdegree 1.

We start by reducing the problem of finding g that M-implements f, for the case in which $\mathcal{G}_M$ is a line, to the problem of finding a satisfying assignment for a formula in 2CNF (that is every clause has at most 2 literals). We assume $D = \{t_1, \cdots, t_n\}$, $\mathcal{O} = \{o_1, \cdots, o_m\}$ and that, for $i = 2, \cdots, n$, $M(t_i) = \{t_i, t_{i-1}\}$ and $M(t_1) = \{t_1\}$. We construct a formula Φ in 2CNF in the following way. The formula Φ has the variables x_{ij} for $1 \leq i \leq n$ and $1 \leq j \leq m$. The intended meaning of variable x_{ij} being set to true is that $g(t_i) = o_j$. We will construct Φ so that every truth assignment that satisfies Φ describes g that M-implements f. We do so by considering the following clauses:

1. Φ contains clauses $(x_{if(t_i)} \vee x_{i-1f(t_i)})$, for $i = 2, \cdots, n$, and clause $x_{1f(t_1)}$. These clauses encode the fact that for g to M-implement f it must be the case that there exists at least one neighbor a of t_i in $\mathcal{G}_M$ such that $g(a) = f(t_i)$.
2. Φ contains clauses $(x_{ij} \rightarrow \overline{x}_{ik})$, for $i = 1, \cdots, n$ and for $1 \leq k \neq j \leq m$. These clauses encode the fact that g assigns at most one outcome to t_i.

3. Φ contains clauses $(x_{if(t_i)} \rightarrow \overline{x}_{i-1k})$ for all $i = 2, \cdots, n$ and for all k such that $t_i(o_k) > t_i(f(t_i))$.
   These clauses encode the fact that if g M-implements f and $g(t_i) = f(t_i)$ then agent of type t_i does not prefer $g(t_{i-1})$ to $g(t_i)$. Therefore, in this case t_i's best response is t_i itself.
4. Φ contains clauses $(x_{i-1f(t_i)} \rightarrow \overline{x}_{ik})$ for all $i = 2, \cdots, n$ and for all k such that $t_i(o_k) \geq t_i(f(t_i))$.
   These clauses encode the fact that if g M-implements f and $g(t_{i-1}) = f(t_i)$ then agent of type t_i does not prefer $g(t_i)$ to $g(t_{i-1})$. Therefore, in this case t_i's best response is t_{i-1}.

It is easy to see that Φ is satisfiable if and only if f is M-implementable. The above reasoning can be immediately extended to the case in which each node of $\mathcal{G}_M$ has outdegree at most 1 (that is $\mathcal{G}_M$ is a collection of cycles and paths). We thus have the following theorem.

Theorem 5. *The* Implementability *Problem can be solved in time polynomial in the sizes of the domain and of the outcome sets for correspondences of maximum outdegree* 1.

4 Implementability with quasi linear utility

In this section we consider mechanisms with payments; that is, the mechanism picks an outcome and a payment to be transferred to the agent, based on the reported type of the agent. Therefore a mechanism is now a pair (g, p) where g is the outcome function and $p : D \rightarrow \mathbb{R}$ is the payment function. We assume that the agent has quasi linear utility.

Definition 4. *A mechanism (M, g, p) for an agent with quasi-linear utility is a triplet where $M : D \rightarrow 2^D$ is a correspondence, $g : D \rightarrow D$ is an outcome function, and $p : D \rightarrow \mathbb{R}$ is a payment function.*

The mechanism defines a best-response function $\phi_{(g,p)} : D \rightarrow D$ where $\phi_{(g,p)}(t) \in \arg\max_{t' \in M(t)}\{t(g(t')) + p(t')\}$. If $t \in \arg\max_{t' \in M(t)}\{t(g(t')) + p(t')\}$ then we set $\phi_g(t) = t$.

Definition 5. *The pair (g, p) M-implements social choice function $f : D \rightarrow \mathcal{O}$ for an agent with quasi-linear utility if for all $t \in D$, $g(\phi_{(g,p)}(t)) = f(t)$.*

The pair (g, p) truthfully M-implements social choice function f for an agent with quasi-linear utility if (g, p) M-implements f and, for all $t \in D$, $\phi_{(g,p)}(t) = t$.

In the rest of this section we will just say that (g, p) M-implements (or truthfully M-implements) f and mean that M-implementation is for agent with quasi-linear utility.

Testing truthful M-implementability of a social choice function f can be done in time polynomial in the size of the domain by using the following theorem that gives necessary and sufficient conditions. The proof is straightforward from the proof of [2] (see also [4]).

Theorem 6. *Social choice function f is truthfully M-implementable if and only if $\mathcal{G}_{M,f}$ has no negative weight cycle.*

Therefore, as in the previous case when payments were not allowed, if M has the NRC property then the Revelation Principle holds and the class of M-implementable social choice functions coincides with the class of truthfully M-implementable social choice functions. We next ask what happens for correspondences M for which the NRC property does not hold. Our answer is negative as we show that the following problem is NP-hard.

Problem 2. The Quasi-Linear Implementability problem is defined as follows.

Input: domain D, outcome set $\mathcal{O}$, social choice function $f : D \to \mathcal{O}$ and correspondence M.

Task: decide whether there exists (g, p) that M-implements f.

We start with the following technical lemma.

Lemma 2. *Let M be a correspondence and let f be a social choice function for which correspondence graph has a negative-weight cycle $t \to t' \to t$ of length 2. If (g, p) M-implements f then*

$$\{\phi_{(g,p)}(t), \phi_{(g,p)}(t')\} \not\subseteq \{t, t'\}.$$

Proof. Let us assume for sake of contradiction that (g, p) M-implements f and that

$$\{\phi_{(g,p)}(t), \phi_{(g,p)}(t')\} \subseteq \{t, t'\}. \tag{7}$$

Since cycle $C := t \to t' \to t$ has weight

$$t(f(t)) - t(f(t')) + t'(f(t')) - t'(f(t)) < 0 \tag{8}$$

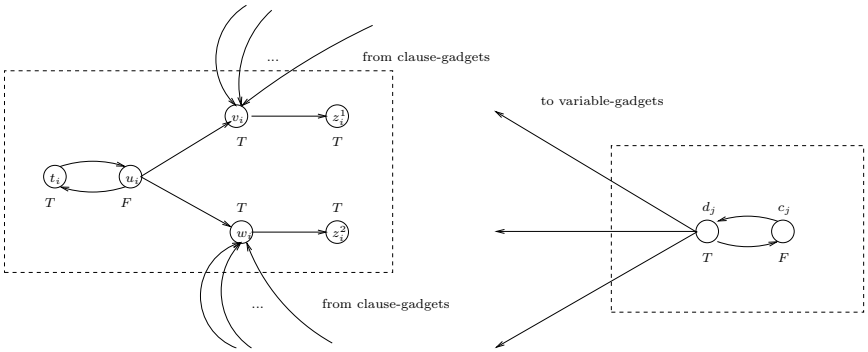
then $f(t) \neq f(t')$. Therefore, since (g, p) M-implements f, it holds $\phi_{(g,p)}(t) \neq \phi_{(g,p)}(t')$ and thus (7) implies that $\{\phi_{(g,p)}(t), \phi_{(g,p)}(t')\} = \{t, t'\}$.

Suppose that $\phi_{(g,p)}(t) = t'$ and thus $\phi_{(g,p)}(t') = t$. Then for (g, p) to M-implement f it must be the case that $g(t) = f(t')$, $g(t') = f(t)$. But then the payment function p must satisfy both the following:

$$p(t') + t(f(t)) \geq p(t) + t(f(t')),$$
$$p(t) + t'(f(t')) \geq p(t') + t'(f(t)),$$

which contradicts (8). The same argument can be used for the case $\phi_{(g,p)}(t) = t$ and $\phi_{(g,p)}(t') = t'$. □

The reduction. We are now ready to show our reduction from 3SAT to the Quasi-Linear Implementability problem. The reduction is similar in spirit to the one of the previous section. We start from a Boolean formula Φ in conjunctive normal form whose clauses contain exactly 3 literals and we construct a domain



(a) The variable gadget (b) The clause gadget

Fig. 2. Gadgets used in the reduction.

D, a set of outcomes $\mathcal{O}$, a social choice function f, and a correspondence M such that there exists (g, p) that M-implements f if and only if Φ is satisfiable.

We set $\mathcal{O} = \{T, F\}$ and fix constants $0 < \beta < \delta$. Let $x_1, \ldots, x_n$ be the variables and $C_1, \ldots, C_m$ be the clauses of Φ. The reduction uses two different gadgets: variable gadgets and clause gadgets.

We have one variable gadget for each variable; the gadget for x_i is depicted in Figure 2(a) where the depicted edges are edges of $\mathcal{G}_M$. Each variable x_i of the formula Φ adds six new types to the domain D: $t_i, u_i, v_i, w_i, z_i^1$, and z_i^2 satisfying the following:

$$t_i(T) - t_i(F) < u_i(F) - u_i(T), \tag{9}$$
$$u_i(T) - u_i(F) = \beta. \tag{10}$$

Nodes v_i and w_i have incoming edges from the clause gadgets. The role of these edges will be clear in the following. The labeling of the nodes describes the social choice function f to be implemented. More precisely, we have that $f(t_i) = f(v_i) = f(w_i) = f(z_i^1) = f(z_i^2) = T$ and $f(u_i) = F$.

We observe that, by (9), cycle $C := t_i \to u_i \to t_i$ has negative weight. Moreover, since $\phi_{(g,p)}(t_i) \in M(t_i) = \{t_i, u_i\}$, by Lemma 2, it must be the case that $\phi_{(g,p)}(u_i) \notin \{t_i, u_i\}$. Therefore, if (g, p) M-implements f then $g(\phi_{(g,p)}(u_i)) = f(u_i) = F$, and thus g assigns outcome F to *at least* one of the neighbors of u_i. Intuitively, the fact that the outcome function g assigns F to at least one between v_i and w_i corresponds to assigning "false" to literal x_i and $\bar{x}_i$.

We have one clause gadget for each clause; the gadget for clause C_j is depicted in Figure 2(b). Each clause C_j of Φ adds two new types to the domain D: c_j and d_j satisfying

$$c_j(F) - c_j(T) < d_j(T) - d_j(F), \tag{11}$$
$$d_j(T) - d_j(F) = \delta. \tag{12}$$

Node d_j has three edges directed towards the three variable gadgets corresponding to the variables appearing in the clause C_j. Specifically, if the clause C_j contains the literal x_i then d_j is linked to the node v_i. Conversely, if C_j contains the literal $\bar{x}_i$ then d_j is connected to the node w_i. The social choice function f is defined by the labeling of the nodes; that is, $f(d_j) = T$ and $f(c_j) = F$.

Similarly to the variable gadget, we observe that (11) implies that c_j and d_j constitute a cycle of negative weight of length 2. Since $\phi_{(g,p)}(c_j) \in \{c_j, d_j\}$, then, by Lemma 2, it must be the case that $\phi_{(g,p)}(d_j) \notin \{c_j, d_j\}$. Since for any (g, p) that M-implements f it must be the case that g assigns T to d_j's best response, then g assigns outcome T to at least one of the neighbors of d_j from a variable gadget. We will see that this happens for all clauses if and only if the formula Φ is satisfiable. This concludes the description of the reduction.

We next prove that the reduction described above is correct. Suppose Φ is satisfiable, let τ be a satisfying assignment for Φ, let γ be a constant such that $\beta < \gamma < \delta$ and consider the following pair (g, p). For $i = 1, \cdots, n$, we set $g(a) = T$ and $p(a) = 0$ for all nodes a of the variable gadget for x_i except for v_i and w_i. Then, if $\tau(x_i) = 1$, we set $g(v_i) = T$, $p(v_i) = 0$, $g(w_i) = F$ and $p(w_i) = \gamma$. If instead $\tau(x_i) = 0$, we set $g(v_i) = F$, $p(v_i) = \gamma$, $g(w_i) = T$ and $p(w_i) = 0$. For $j = 1, \cdots, m$, we set $g(c_j) = g(d_j) = F$ and $p(c_j) = p(d_j) = 0$.

We now show that (g, p) M-implements f. We show this only for types u_i from variable gadgets and types d_j from clause gadgets, as for the other types the reasoning is immediate. Notice that by definition, g assigns F to exactly one of v_i and w_i and T to the other. Thus, denote by a the vertex $a \in \{v_i, w_i\}$ such that $g(a) = F$ and by b the vertex $b \in \{v_i, w_i\}$ such that $g(b) = T$. We show that a is u_i's best response under (g, p). Observe that $u_i(g(a)) + p(a) = u_i(F) + \gamma > u_i(F) = u_i(g(t_i)) + p(t_i)$. Therefore t_i is not u_i's best response. On the other hand, we have $u_i(g(b)) + p(b) = u_i(T)$. But then, since $\gamma > \beta = u_i(T) - u_i(F)$, we have that a is u_i's best response under (g, p).

For d_j, we observe that, since τ satisfies clause C_j, there must exists at least one literal of C_j that is true under τ. By the definition of g, there exists at least one neighbor, call it a_j, of d_j from a variable gadget such that $g(a_j) = T$. We next show that a_j is d_j's best response. Notice that $p(a_j) = 0$. For all vertices b adjacent to d_j for which $g(b) = F$, we have $p(b) \leq \gamma$. But then, since $\gamma < \delta = d_j(T) - d_j(F)$ we have that a_j is d_j's best response under (g, p).

Conversely, consider an outcome function (g, p) that implements f and construct truth assignment τ as follows. Observe that, for any clause C_j, d_j and c_j constitute a cycle of negative weight and length 2. Moreover, c_j's best response is either c_j or d_j and thus, by Lemma 2, it must be the case that d_j's best response is a vertex, call it a_j, from a variable gadget such that $g(a_j) = T$. Then if $a_j = v_i$ for some i then we set $\tau(x_i) = 1$; if instead $a_j = w_i$ for some i we set $\tau(x_i) = 0$. Assignment τ (arbitrarily extended to unspecified variables) is easily seen to satisfy Φ.

The above discussion and the observation that the reduction can be carried out in polynomial time proves the following theorem.

Theorem 7. *The* Quasi-Linear Implementability *problem is NP-hard even for outcome sets of size* 2.

5 Conclusions

We have seen that is it NP-hard to decide if a given social choice function can be implemented even under the premise that the function does not admit a truthful implementation. Indeed, for these function it is NP-hard to decide if there is a non-truthful implementation, which in turn is the only way to implement them. An important factor here is the structure of the domain and the partial information, which we encode in the correspondence graph. In particular, we have the following results:

Correspondence Graph	No Payments	Payments and Quasi-linear Agent
Path	Polynomial [Th. 5]	Always implementable [3, Th. 4]
Directed acyclic	NP-hard [Th. 4]	Always implementable [3, Th. 4]
Arbitrary	NP-hard [Th. 4]	NP-hard [Th. 7]

Note that for directed acyclic graphs, the Quasi Linear Implementability Problem (where we ask implementability with payments) is trivially polynomial since all social choice functions are implementable wheras it is NP-hard to decide if an implementation without payments exists. So, it is also difficult to decide if payments are necessary or not for implementing a given function. Once again, this task becomes easy when restricting to truthful implementations [4].

Another interesting fact is that the problem without payments is not difficult because there are many possible outcomes, but because an agent may have several ways of misreporting his type. Indeed, the problem is easy if the agent has at most one way of lying (Theorem 5), but becomes NP-hard already for three (Theorem 4). The case of two remains open.

Finally, the fact that we consider the principal-agent model (the same as in [1]) only makes our negative results stronger since they obviously extend to the case of several agents (simply add extra agents whose corresponding function is $M(t) = \{t\}$).

On the other hand it remains open whether the positive result for graphs of outdegree at most 1 can be extended to many agents. Here the difficulty is the inter-dependance between the best response rules of the agents.

References

1. Jerry R. Green and Jean-Jacques Laffont. Partially Verifiable Information and Mechanism Design. *The Review of Economic Studies*, 53:447–456, 1986.
2. Jean-Charles Rochet. A Condition for Rationalizability in a Quasi-Linear Context. *Journal of Mathematical Economics*, 16:191–200, 1987.
3. Nirvikar Singh and Donald Wittman. Implementation with partial verification. *Review of Economic Design*, 6(1):63–84, 2001.
4. Rakesh V. Vohra. Paths, cycles and mechanism design. Technical report, Kellogg School of Management, 2007.