Mechanisms with Verification for Any Finite Domain

Carmine Ventre*

Abstract

In this paper we generalize the concept of *verification* introduced by Nisan and Ronen [STOC 1999]. We assume to have selfish agents with general valuation functions and we study *mechanisms with verification* for optimization problems with these selfish agents. We provide a technique for designing *truthful* mechanisms with verification that optimally solve the underlying optimization problem. Our technique can be applied to a rich class of problems that includes, as special cases, *utilitarian* problems and many others considered in literature for so called *one-parameter* agents (e.g., the makespan studied by Archer and Tardos [STOC 2001]). Our technique extends the one recently presented by Auletta et al as it works for *any* finite multi-dimensional valuation domain. No method was known to deal with any domain. As special case we give a different proof (w.r.t. to the one given by Nisan and Ronen) of the existence of exact truthful mechanisms with verification for Scheduling Unrelated Selfish Machines. Furthermore, our technique also applies to the case of *compound* agents (i.e., agents declaring more than a value). No solution was known for designing mechanisms (with verification) for problems involving such general kind of agents. As an application we provide the *first* optimal truthful mechanism with verification for Scheduling Unrelated Selfish Compound Machines in which every agent controls more than one (unrelated) machine. This mechanism does not run in polynomial time. We then turn our attention to efficient approximating truthful mechanisms and provide a technique that transforms any approximation algorithm into a mechanism with verification with no significant loss of approximation ratio. This technique works for smooth problems involving compound one-parameter agents. We apply this technique to Scheduling Related Compound Machines problem (i.e, agents control more related machines). If the number of machines is constant then our solution runs in polynomial-time. Finally, we give some considerations on the construction of mechanisms (with verification) for infinite domains.

1 Introduction

Many computer scientists look at the world from a new perspective: they study problems assuming there are *selfish* entities working for their own interests rather than for community interests. This implies that one has to design new *algorithms* that have to deal, not just with the combinatorial structure of the problem, but also, and perhaps *mainly*, with *private* interests conflicting with the aim of *optimizing*. The new perspective is motivated by many real-life situations. Consider, for example, computations over the Internet. They often involve self-interested parties (*selfish agents*) which may manipulate the system by misreporting a fundamental piece of information they hold (their own *type* or *valuation*). The system runs some algorithm which, because of the misreported information, is no longer guaranteed to return a "globally optimal" solution (optimality is naturally expressed as a function of agents' types) [23, 21]. Since agents can manipulate the algorithm by misreporting their types, one augments algorithms with carefully designed payment functions which make disadvantageous for an agent to do so. A *mechanism* consists of an algorithm (also termed *social choice function*) and payment functions which associate a payment with every agent. Payments should guarantee that it is in the agent's interest to report her type correctly. A social choice function A is *implementable* if the *utility* that an agent derives from the outcome chosen by A and from the payment she receives is maximum when this agent reports her type correctly (see Sect. 2 for a formal definition of

*Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, via Ponte don Melillo, I-84084 Fisciano (SA), Italy. E-mail: {ventre}@dia.unisa.it. Research supported by the European Project FP6-15964, Algorithmic Principles for Building Efficient Overlay Computers (AEOLUS).

these concepts). When a social choice function A is implementable we refer to the pair (A, P) to as a *truthful mechanism*. The only known general technique for designing truthful mechanisms is the classical Vickrey-Clarke-Groves (VCG) paradigm [29, 15, 10]. These mechanisms suffer from two main limitations: (i) they can be used only for a limited family of optimization functions (see e.g. [21]) and (ii) they require the algorithm to compute exact solutions which, in many cases, is unfeasible if the mechanism has to run in polynomial time (see e.g. [22]). In their seminal work, Nisan and Ronen [20] introduce a mechanism design approach to computer science problems having *non-utilitarian* optimization functions, and show that even *exponential-time* mechanisms cannot achieve optimal solutions (in contrast with the unselfish counterpart where a $(1+\varepsilon)$-approximation can be obtained in polynomial-time). An alternative to VCG mechanisms is to restrict the *domain* of the agents (i.e., possible values they can report). For example the so-called *one-parameter* agents have been studied in [20, 2]. Unfortunately, these domains are rather limited: for instance, although they can model scheduling problems on related machines [2], they cannot model the unrelated case in [21], nor the case of agents owning more than one machine.

A quite innovative mechanism design approach has been introduced by Nisan and Ronen [21] in order to overcome the above mentioned difficulties for their scheduling problem: the mechanism can observe the job release time and provides payments *after* the solution has been implemented. These mechanisms are called *mechanisms with verification*. More "classical" mechanisms without verification award the payment associated to an agent unconditionally (i.e., without performing any kind of verification and solely based on the agents reported types). There are several reasons for being interested in mechanisms with verification. First of all, there are specific optimization problems for which verification allows to overcome certain impossibility results for mechanisms without verification [21, 4, 6] (which holds also for one-parameter agents). Moreover, mechanisms with verification are very natural and many real-life applications require and/or already implement this kind of approach: reputation is used by the e-Bay system to measure credibility of sellers; service providers offer connectivity to the Internet with the guarantee of a minimal rate. Finally, verification helps in designing truthful mechanisms.

1.1 Overview of the Results

In this paper we prove the first general result on mechanisms with verification and show that, for any finite domain, there is a mechanism that optimizes *any* minimization (resp. maximization) function monotone non-increasing (resp. non-decreasing) in the agents' valuations. This result applies to *any finite domain* and extend to a "multidimensional scenario" called *compound agents* (see Def. 12). We indeed provide a social choice function (i.e., an optimization algorithm) which is implementable with verification on finite domains and that maximizes any optimization function $\mu(\cdot)$ which is monotone in the agents valuations (i.e., $\mu(v_1(X), \ldots, v_n(X))$ is non-decreasing in each agent valuation $v_i(X)$). With our "always implementable" social choice function we are able to construct truthful mechanisms (with verification) optimizing the underlying optimization function (Cor. 9). Observe that VCG mechanisms [29, 10, 15] can only deal with particular functions of this form called *affine maximizers* (basically, the case $\mu(v_1(X), \ldots, v_m(X)) = \sum_i \beta_i v_i(X)$, with constants β_i defined by the mechanisms). The $Q||C_{\max}$ scheduling problem is an example of an optimization problem involving a monotone non-increasing function (thus our result applies to $Q||C_{\max}$) that is not an affine maximizer. Our result gives an alternative proof of the existence of an exact truthful mechanism with verification for unrelated machines [21].

Interestingly enough, our results extend to the case of compound agents (see Sec. 4). To the best of our knowledge, these are the first results/techniques on mechanisms with verification for such a "multidimensional" scenario (it should be noticed that already the "one-dimensional" case is a generalization of both one-parameter [2] and comparable types [6]). These results give us powerful tool to solve very general problems. Indeed, we present the first truthful exact mechanism with verification for scheduling unrelated machines when agents control *more than one machine* (thus generalizing the "one machine per agent" scenario/results in [21]). As expected, for hard problems, these exact mechanisms (and in general those obtained with our technique above) do not run in polynomial time. We thus move our attention to approximation polynomial-time mechanisms. We investigate the design of payment functions which implement classical approximation algorithms. In particular, we consider compound agents in which each "dimension" is a

one-parameter valuation. In this setting, we show that any approximation algorithm for a smooth problem (Def. 17) can be transformed into a truthful mechanism (with verification) for the problem. The resulting mechanism (essentially) preserves the approximation ratio of the algorithm. In order to guarantee a polynomial running time, we require a constant number of compound agents with constant dimensions (Th. 18). The most relevant application is a polynomial-time $c(1+\varepsilon)$-approximation mechanism for scheduling related machines when the agents control *more than one machine* (see Def. 20) (given a c-approximating algorithm for the problem). To the best of our knowledge, no solution was known for this natural extension of the problem studied in [2].

Finite Domains. The assumption of finite domains deserves some more discussions. From a practical point of view, many real-life applications involve agents with types from a finite and discrete domain (e.g., when costs or execution times are expressed as multiples of a given monetary or time unit and an upper bound is known). From a theoretical point of view, it is interesting to investigate how to use verification to overcome impossibility results proved for infinite domains (when verification is not allowed). The only case in which the assumption of finite domains has been removed is for problems involving one-parameter agents (see [12, 4]). In Sec. 5 we study truthful mechanisms with verification for infinite domains. We show that known techniques, developed for mechanisms without verification, seem to "cancel" the advantages given by the verification. Besides one-parameter agents, no result (neither positive nor negative) was known on the design of mechanism with verification for infinite domains. We stress that for finite domain several impossibility results for mechanisms without verification are known [2, 8], some of them applying to our optimization functions in the "multi-dimensional" scenario. This shows that verification does help for finite domains.

1.2 Related Works

The study of implementable social choice functions starts with classical Gibbard-Satterthwaite [13, 27] theorem that proved that in some cases only *dictatorial* social choice functions can be implemented. If one restricts to quasi-linear utility functions (i.e., payment received plus agent's monetary valuation), then affine maximizers (see above) can be implemented using the celebrated VCG mechanism [29, 15, 10]. Roberts [24] showed that VCG mechanisms are essentially the only truthful mechanisms if no hypothesis is made on the domains of the agents. Mechanisms for *one-parameter* agents have been characterized in [20, 2]. For one-parameter agent there exist truthful mechanisms for scheduling to minimize the makespan [2, 3, 1] and for some types of combinatorial auctions [17]. Lavi, Mu'alem and Nisan showed that a *weak monotonicity* condition (W-MON) characterizes *order-based* domains with range constraints [4]. Similar results hold for *linear inequality constraints* on the domain [16] and, more in general, for *convex domains* [26] (each class extending the prior one and the result for one-parameter agents). These results concern mechanisms which do *not* use verification and cannot be applied to our case (indeed, one wishes to use mechanisms with verification to solve problems which the other mechanisms cannot solve [21, 4, 6]).

The study of social choice functions implementable with verification was started by Nisan and Ronen [21], who gave a truthful $(1+\varepsilon)$-approximate mechanism for scheduling on (a constant number) of unrelated machines to minimize the makespan. Similar results have been obtained by Auletta *et al* [4] for scheduling on any number of related machines (see also [5] for the online case). These results are based on a characterization of mechanisms with verification [4] for one-parameter agents. Mechanisms with verification for one-parameter agents also appear in [12] where the main contribution is in providing payment schemes, computable in constant time (given the reading of the input), working with infinite domains. Augmenting an algorithm obeying some monotonicity properties, with such schemes leads to truthful mechanisms with verification enjoying the same execution time and approximation ratio of the algorithm. As already mentioned, a recent work [6] characterizes mechanisms with verification for a rich class of finite domains. Afterwards, they extend their result to the class of one-parameter agents providing different mechanisms for several scheduling problems.

The work [9] presents a general technique for constructing polynomial-time approximation mechanisms

for *utilitarian* problems (those affine maximizers in which one seeks to maximize the social welfare, i.e., sum of the valuation of the agents). The technique they use is similar to the one we use to obtain truthful mechanisms for multidimensional agents. Similar methods were already used in [6]. This approach derives from [19], where agents' types are even simpler than the one-parameter case (this kind of agents are called KSM bidders). Polynomial-time mechanisms which approximate the social welfare for certain auctions are given in [11]. Mechanisms in [9, 11] do not use verification, but all problems are utilitarian and solutions rely on VCG mechanisms. We are interested in truthful mechanisms with verification but we do not restrict ourselves to utilitarian problems (see for example the applications to non-utilitarian scheduling problems).

Outline. Since we focus on mechanisms with verification (or implementation with verification), through the paper we omit to explicitly write that our solutions use verification. We recall that in Sec. 2 we gave some preliminaries in order to enter in the setting and to be confident with our notation. In Sec. 3 we present our technique; its extensions to compound agents are discussed in Sec. 4 where we also present our method to implement given approximation algorithms. Finally, in Sec. 5 we present our results on mechanisms for infinite domains.

2 Preliminaries

We have a finite set $\mathcal{O} = \{X_1, \ldots, X_K\}$ of K possible *alternative outcomes*. We also have m selfish agents, each of them having a *valuation* (or *type*) $v_i \in D_i$, with D_i being the *domain* of agent i. Domains are multi-dimensional in the sense of [7]: The domain of v_i is $D_i \subset \mathbb{R}^K$ with the kth coordinate of type v_i being $v_i(X_k)$, this type's utility for outcome X_k (i.e., $v_i = (v_i(X_1), \ldots, v_i(X_K))$).[1] The valuation v_i is known to agent i only.

A *social choice function* $A : D \to \mathcal{O}$ maps the agents' valuations into a particular outcome $A(v_1, \ldots, v_m)$, where $D = D_1 \times \cdots \times D_m$ is the domain of function A.

A *mechanism* $M = (A, P)$ is a social choice function A augmented with a *payment scheme* $P = (P_1, \ldots, P_m)$, where each P_i is a function $P_i : D \to \mathbb{R}$. The mechanism elicits from each agent its valuation; an agent i can misreport her valuation to any $b_i \in D_i$. The mechanism, on input the *reported* valuations $\mathbf{b} = (b_1, \ldots, b_m)$, selects outcome X as $X = A(\mathbf{b})$ and assigns payment $P_i(\mathbf{b})$ to agent i. The *utility* of agent i, when receiving a payment $P_i(\mathbf{b})$, with valuation v_i is thus $P_i(\mathbf{b}) + v_i(X)$. This kind of utilities are commonly denoted to as *quasi-linear* utilities.

We let $b_i \in D_i$ denote the valuation (or type) *reported* by agent i and by $\mathbf{b}_{-i}$ the vector $(b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_m)$ of all valuations (or types) reported by the other agents. We stress that both the outcome and the payments depend on the *reported* valuations $\mathbf{b} = (b_1, \ldots, b_m)$. In particular, given $\mathbf{b}_{-i}$, the reported type b_i determines the outcome $A_{\mathbf{b}_{-i}}(b_i) := A(\mathbf{b})$ and the payment $P_i(\mathbf{b})$.

For a vector $\mathbf{x} = (x_1, \ldots, x_m)$, we let $\mathbf{x}_{-i}$ denote the vector $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m)$ and $(y, \mathbf{x}_{-i})$ the vector $(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_m)$; similarly, $D_{-i} := D_1 \times \cdots \times D_{i-1} \times D_{i+1} \times \cdots \times D_m$.

A *mechanism with verification* can detect whether $b_i \neq v_i$ if and only if $v_i(A_{\mathbf{b}_{-i}}(b_i)) < b_i(A_{\mathbf{b}_{-i}}(b_i))$; in this case, agent i will *not* receive the associated payment. This verification model generalizes the concept of verification introduced in [21].

Definition 1 ([21]) *A social choice function A is* implementable with verification *if there exists $P = (P_1, \ldots, P_m)$ such that, for all i, all $\mathbf{b}_{-i} \in D_{-i}$ the utility of agent i with type v_i is maximized by reporting $b_i = v_i$.*

In this case, the pair (A, P) is called a *truthful mechanism with verification*.

[1] An alternative, but completely equivalent, way to define these multidimensional valuations is the following: Each valuation v_i is a function $v_i : \mathcal{O} \to \mathbb{R}$ representing a monetary valuation $v_i(X)$ that agent i associates to outcome $X \in \mathcal{O}$. This definition has been used, for example, in [6].

A different way of reading Def. 1 is that there exists $P = (P_1, \ldots, P_m)$ such that, for all $v_i, b_i \in D_i$ and $\mathbf{b}_{-i} \in D_{-i}$, the following inequality holds:

$$v_i(A_{\mathbf{b}_{-i}}(v_i)) + P_i(v_i, \mathbf{b}_{-i}) \geq v_i(A_{\mathbf{b}_{-i}}(b_i)) \tag{1}$$

if $v_i(A_{\mathbf{b}_{-i}}(b_i)) < b_i(A_{\mathbf{b}_{-i}}(b_i))$, and the following inequality holds:

$$v_i(A_{\mathbf{b}_{-i}}(v_i)) + P_i(v_i, \mathbf{b}_{-i}) \geq v_i(A_{\mathbf{b}_{-i}}(b_i)) + P_i(b_i, \mathbf{b}_{-i}) \tag{2}$$

if $v_i(A_{\mathbf{b}_{-i}}(b_i)) \geq b_i(A_{\mathbf{b}_{-i}}(b_i))$.

We are interested in social choice functions which are implementable with verification and that optimize some *objective function* $\mu(\cdot)$ which depends on the agent valuations $\mathbf{v} = (v_1, \ldots, v_m)$. For maximization (respectively, minimization) functions, we let $\text{OPT}_\mu(\mathbf{v})$ be $\max_{X \in \mathcal{O}} \mu(X, \mathbf{v})$ (respectively, $\min_{X \in \mathcal{O}} \mu(X, \mathbf{v})$). An outcome $X \in \mathcal{O}$ is an α-approximation of μ for $\mathbf{v} \in D$ if

$$\max \left\{ \frac{\mu(X, \mathbf{v})}{\text{OPT}_\mu(\mathbf{v})}, \frac{\text{OPT}_\mu(\mathbf{v})}{\mu(X, \mathbf{v})} \right\} \leq \alpha.$$

A social choice function A is *α-approximate* for μ if, for every $\mathbf{v} \in D$, $A(\mathbf{v})$ is an α-approximation for μ and $\mathbf{v}$. We stress that, in this paper we consider social choice functions that are implementable with verification and either optimize or α-approximate a function μ. The approximation only refers to how good the selected outcome is and not to the utilities of the agents (which are always maximized by reporting the true valuation). Recall that we assume domains to have finite cardinality.

2.1 Truthful Mechanisms with Verification for Any Finite Domain

For fixed i and $\mathbf{b}_{-i}$, Eq.s 1 and 2 give a system of linear inequalities with unknowns $P^x := P_i(x, \mathbf{b}_{-i})$, for $x \in D_i$.

As in [6], we observe that constraints of Eq. 1 are easily satisfied for finite domains. Indeed, for finite domains (as we are assuming), by computing (in time polynomial in $|D_i|$) $\max_{a \in D_i}\{a(X) - a(A_{\mathbf{b}_{-i}}(a)) | X \in \mathcal{O}\}$ and by setting, for any $x \in D_i$, P^x to at least as such quantity one gets rid of them. Therefore, we can just study the constraints given by Eq. 2. Such a system of inequalities is compactly encoded by the following graph.

Definition 2 (verification-graph) *Let A be a social choice function. For every i and $\mathbf{b}_{-i} \in D_{-i}$, the* verification-graph $\mathcal{V}(A_{\mathbf{b}_{-i}})$ *has a node for each type in D_i. The set of edges of $\mathcal{V}(A_{\mathbf{b}_{-i}})$ is defined as follows. For every $a, b \in D_i$ add an edge (a, b) whenever the solution $Y = A_{\mathbf{b}_{-i}}(b)$ is such that $a(Y) \geq b(Y)$. The weight of edge (a, b) (if any) is $\delta(a, b) := a(X) - a(Y)$ where $X = A_{\mathbf{b}_{-i}}(a)$.*

The definition of the verification-graph is a modification of the graph introduced in [18] (and subsequently used by [6]) to study the case in which verification is not allowed. The graph of [18] has an edge of weight $\delta_{a,b}$ between *all* types a and b. Instead in the verification-graph the edge between a and b is missing, if $a(A_{\mathbf{b}_{-i}}(b)) < b(A_{\mathbf{b}_{-i}}(b))$. The existence of the edge (a, b) encodes the fact that an agent with a valuation a can declare to have type b without being caught by the verification. The next theorem states that, in order to check that a social choice function is implementable, it suffices to check that cycles of the verification-graph have a nonnegative weight.

Theorem 3 *A social choice function A is implementable with verification if and only if, for all i and $\mathbf{b}_{-i} \in D_{-i}$, the graph $\mathcal{V}(A_{\mathbf{b}_{-i}})$ does not have negative weight cycles.*

The theorem follows from the observation that the system of linear inequalities involving the payment functions is the linear programming dual of the shortest path problem on the verification-graph. Therefore, a simple application of Farkas lemma shows that the system of linear inequalities has solution if and only if the verification-graph has no negative weight cycle. The same argument has been used for the case in which verification is not allowed albeit on a different graph (see [25, 18, 16]).

Computing the payments. When verification is not allowed the verification-graph is complete. If the system is feasible, one solution is to set each P^x equal to the length of the shortest path from an arbitrarily chosen root vertex to x. The fact that the graph is complete implies that all P^x assume a finite value. When verification is allowed the graph is not complete and, possibly, the graph is not connected. Thus, setting payments to shortest paths from an arbitrary vertex could lead to unbounded payments. However, it is always possible to design bounded payments satisfying truthfulness conditions. A simpler solution would augment the graph $\mathcal{V}(A_{\mathbf{b}_{-i}})$ with a fake node. Such node will be connected with all valuations in the domain through edges weighting 0. In this case the payments are simply defined as shortest paths from the fake node in the augmented graph. Here, every node has an associated payment which is finite.

3 MAX_μ Social Choice Function

In this section we present our technique for obtaining truthful mechanisms with verification for any finite domain. We use the social choice function defined below.

Definition 4 ([6]) *Let $\mu(\cdot)$ be any function monotone non-decreasing in each of its arguments $b_1(X), \ldots, b_m(X)$, with $X \in \mathcal{O}$ and $b_i \in D_i$. For any $X_1, \ldots, X_\ell \in \mathcal{O}$, let MAX_μ be the social choice function that, on input $(b_1, \ldots, b_m) \in D$, returns the solution X_j of minimum index that maximizes the value*

$$\mu(b_1(X_j), \ldots, b_m(X_j)).$$

Notice that MAX_μ social choice function uses a precedence relation among outcomes in $\mathcal{O}$. Indeed, when more solutions lead to the same value of the objective function μ, MAX_μ always selects the "minimum" solution. Therefore, the following straightforward fact holds.

Fact 5 *For any i and any $\mathbf{b}_{-i} \in D_{-i}$, let a and b two valuations in D_i and denote $X = MAX_\mu(a, \mathbf{b}_{-i})$ and $Y = MAX_\mu(b, \mathbf{b}_{-i})$. If $\mu(a(X), \mathbf{b}_{-i}(X)) = \mu(b(Y), \mathbf{b}_{-i}(Y))$ then $X = Y$.*

The MAX_μ function can be also used to minimize a function μ which is non-increasing in each of its arguments: in fact, given such a μ, simply running $\text{MAX}_{-\mu}$ one obtains the minimum of μ. As observed MAX_μ social choice function has been introduced in [6]. In that work it is shown that MAX_μ function is implementable for comparable types.

Definition 6 *Agent i's domain D_i is* comparable *if for any $a, b \in D$ either $a \leq b$ or $b \leq a$ (where $a \leq b$ means that for all $X \in \mathcal{O}$, $a(X) \leq b(X)$).*

Next we summarize some of the results in [6] in the following concise theorem.

Theorem 7 ([6]) *Let $\mu(\cdot)$ be any function monotone non-decreasing in each of its arguments $b_1(X), \ldots, b_m(X)$, with $X \in \mathcal{O}$ and $b_i \in D_i$. MAX_μ is implementable for comparable types.*

The proof of the previous result essentially exploits monotonicity of MAX_μ, monotonicity of μ and monotonicity of comparable types. It turns out that, monotonicity of comparable types can be overcome using properties of $\mathcal{V}_{\mathbf{b}_{-i}}(\text{MAX}_\mu)$, and thus MAX_μ is *always* implementable for *any* finite domain. Let us proceed more formally. The main theorem of the section is the following:

Theorem 8 *Let $\mu(\cdot)$ be any function monotone non-decreasing in each of its arguments $b_1(X), \ldots, b_m(X)$, with $X \in \mathcal{O}$ and $b_i \in D_i$, then MAX_μ is implementable.*

Proof. By Th. 3, we have to show that for any i and any $\mathbf{b}_{-i} \in D_{-i}$, there exists no cycle in $\mathcal{V}_{\mathbf{b}_{-i}}(\text{MAX}_\mu)$ of negative weight. Fix i and $\mathbf{b}_{-i} \in D_{-i}$. Let us assume, for sake of contradiction, that there exists a negative-weight cycle in $\mathcal{V}_{\mathbf{b}_{-i}}(\text{MAX}_\mu)$. Assume w.l.o.g. that cycle involves k edges: (a_j, a_{j+1}) for any

$j = 1, \ldots, k-1$ and (a_k, a_1) as last edge. Moreover, let $\text{MAX}_\mu(a_j, \mathbf{b}_{-i}) = X^j$ for any $j = 1, \ldots, k$ (not necessarily all solutions are different). For any $j = 1, \ldots, k-1$, since $\text{MAX}_\mu(a_j, \mathbf{b}_{-i}) = X^j$ then[2]

$$\mu(a_j(X^j), \mathbf{b}_{-i}(X^j)) \geq \mu(a_j(X^{j+1}), \mathbf{b}_{-i}(X^{j+1})). \tag{3}$$

Since edge (a_j, a_{j+1}) belongs to $\mathcal{V}_{\mathbf{b}_{-i}}(\text{MAX}_\mu)$ then $a_j(X^{j+1}) \geq a_{j+1}(X^{j+1})$ and by monotonicity of μ it holds:

$$\mu(a_j(X^{j+1}), \mathbf{b}_{-i}(X^{j+1})) \geq \mu(a_{j+1}(X^{j+1}), \mathbf{b}_{-i}(X^{j+1})). \tag{4}$$

Same reasoning holds for objective function μ applied to valuations a_k, a_1 in relation to solutions X^k, X^1. Namely, it holds the following.

$$\mu(a_k(X^k), \mathbf{b}_{-i}(X^k)) \geq \mu(a_k(X^1), \mathbf{b}_{-i}(X^1)) \geq \mu(a_1(X^1), \mathbf{b}_{-i}(X^1)).$$

Unfolding Inequalities 3 and 4, and using inequalities above we obtain:

$$\begin{aligned}
\mu(a_1(X^1), \mathbf{b}_{-i}(X^1)) &\geq \mu(a_1(X^2), \mathbf{b}_{-i}(X^2)) \\
&\geq \mu(a_2(X^2), \mathbf{b}_{-i}(X^2)) \\
&\geq \ldots \\
&\geq \mu(a_k(X^k), \mathbf{b}_{-i}(X^k)) \\
&\geq \mu(a_k(X^1), \mathbf{b}_{-i}(X^1)) \\
&\geq \mu(a_1(X^1), \mathbf{b}_{-i}(X^1)).
\end{aligned}$$

This implies that all inequalities above hold with the "=" sign and by Fact 5:

$$X^1 = X^2 = \ldots = X^k.$$

This contradicts the hypothesis that the cycle has negative weight as all edges have weight 0. □

Consequences of Th. 8. Theorem above shows that there exists a social choice function which returns an outcome maximizing a given monotone non-decreasing objective function: MAX_μ. In other words, a direct consequence of the last theorem is the following.

Corollary 9 *Let $\mu(\cdot)$ be any monotone non-decreasing function in each of its arguments $b_1(X), \ldots, b_m(X)$, with $X \in \mathcal{O}$ and $b_i \in D_i$. Then, there exists a social choice function OPT_μ which maximizes $\mu(\cdot)$ and is implementable.*

If the set $\mathcal{O}$ of outcomes is very large, then social choice function MAX_μ could not be efficiently computable. Our next result can be used to derive efficiently-computable social choice functions which approximate the objective function by restricting to a suitable subset of the possible outcomes.

Definition 10 ([6]) *A set $\mathcal{O}' \subseteq \mathcal{O}$ is* α-approximation preserving for μ *if, for every $\mathbf{b} \in D$, the set $\mathcal{O}'$ contains a solution X' which is an α-approximation of μ for $\mathbf{b}$.*

Given any α-approximation preserving set $\mathcal{O}' = \{X_1, \ldots, X_l\}$ consider the social choice function MAX_μ that on input $\mathbf{b} \in D$ returns the solution X_j of minimum index in $\mathcal{O}'$ maximizing $\mu(b_1(X_j), \ldots, b_m(X_j))$. We call such a social choice function $APX_\mu^{\mathcal{O}'}$. Theorem 8 implies the following.

Corollary 11 *Let $\mu(\cdot)$ be any optimization function. Assume $\mu(\cdot)$ is monotone non-decreasing in its arguments $b_1(X), \ldots, b_m(X)$, with $X \in \mathcal{O}$ and $b_i \in D_i$. For any α-approximation preserving set $\mathcal{O}'$ the social choice function $APX_\mu^{\mathcal{O}'}$ is an α-approximation for μ and is implementable with verification. Moreover, social choice function $APX_\mu(\mathbf{b})$ can be computed in time proportional to the time needed for computing values $\mu(X, \mathbf{b})$, for $X \in \mathcal{O}'$.*

Notice that last results generalize results presented in [6]. Indeed, they show that the technique works for *any* valuation domain and not just for comparable types. Moreover, these results lead to interesting applications in studying (and solving) problems like Scheduling Unrelated Machines with finite domains.

[2] With $\mathbf{b}_{-i}(X)$ we denote the vector $(b_1(X), \ldots, b_{i-1}(X), b_{i+1}(X), \ldots, b_m(X))$.

Scheduling unrelated selfish machines. The valuations we are studying model well the case of scheduling unrelated selfish machines (see [21]). Looking for mechanisms that are truthful but that do not run in polynomial time, then Cor. 9 gives an alternative (to the one proposed in [21]) technique to design *optimal* truthful mechanisms with verification for the $Q||C_{max}$ problem on unrelated machines. As one could expect, these mechanisms do not run in polynomial-time (as in [21]) since, in the case of unrelated machines, the number of solutions to examine is exponentially large in the size of the input. Recall that for scheduling unrelated machines, valuations – the input of the mechanism – can be compactly represented in the following way: each machine declares n speeds where n is the number of jobs to execute (speeds depend on which job the machine has to execute).

Recall that solutions of [6] do not apply to problems with "multidimensional" (in the sense of [6]) agents. Indeed, [6] partially solves the case of comparable agents in this more general setting, that here we call compound types. The general case, as well as complete solutions for comparable compound types, were left open. In the next section we present a general solution exploiting the MAX_μ technique.

4 Compound Agents

In this section we show that our technique presented in Sec. 3 solves the problem of designing truthful mechanisms with verification for what we call *compound* agents.

Definition 12 (Compound types) *The type $\bar{v} = (v^1, v^2, \ldots, v^d)$ of a d-dimensional compound agent has d components each of which is a single valuation. A* d-dimensional compound type set *is simply the cross product of d type sets and we will denote by $D_i = D_i^1 \times \ldots \times D_i^d$ the type set of agent i. We call an agent with compound type set a* compound agent*.*

A solution $X \in \mathcal{O}$ consists of d components (i.e., $X = (X^1, \ldots, X^d)$). The valuation $\bar{v}$ for a given solution X is a function of v^1, v^2, ..., v^d (e.g., $\bar{v}(X) = v^1(X) + \ldots + v^d(X)$). More specifically, given $f : \mathbb{R}^d \to \mathbb{R}$, $\bar{v}(X) = f(v^1(X), \ldots, v^d(X))$. We assume that the mechanism is able to verify each of the d coordinates independently. This implies that if an agent is caught lying over one of the d components (for example, the agent declared b_1 as first component of its valuation instead of v_1 and the solution computed by the mechanism X is such that $b_1(X) > v_1(X)$) then the agent receives no payment.[3]

We stress that we do not require here that valuation along one dimension is one-parameter, i.e., for all $1 \leq i \leq d$, v^i is *any* valuation in a domain defined in Sec. 2. For the d-dimensional compound agent i having a finite domain means that every D_i^j is finite and that d is finite as well in such a way that the overall domain D_i is finite. We remark that obtaining optimal truthful mechanisms with verification for compound agents corresponds to achieving optimal mechanisms with verification that are collusion-resistant [28, 14] with respect to known (in advance) coalitions.[4] This is, in general, impossible to achieve with mechanisms without verification (e.g., the lower bound in [8] easily extends to compound agents).

For compound agents it holds, as for mechanisms with verification dealing with classical agents, that the constraints of Eq. 1 are easily satisfied for finite domains. Therefore, to work with such kind of agents we need to define the following generalization of the verification-graph. (Notations used so far for multi-dimensional agents naturally extend to compound ones.)

Definition 13 (compound-verification-graph) *Let A be a social choice function. For every i and $\bar{\mathbf{b}}_{-i} \in D_{-i}$, the* compound-verification-graph $\mathcal{M}(A_{\bar{\mathbf{b}}_{-i}})$ *has a node for each type in D_i. The set of edges of $\mathcal{M}(A_{\bar{\mathbf{b}}_{-i}})$ is defined as follows. For every $\bar{a}, \bar{b} \in D_i$ add an edge $(\bar{a}, \bar{b})$ whenever the solution $Y = A_{\mathbf{b}_{-i}}(\bar{b})$ is such that*

[3] One could also define mechanisms with verification for compound agents as mechanisms able to verify only the whole valuation. That is, an agent would be able to cheat along as many coordinates as he wants in a way that the d-dimensional reported valuation is undetectable for the mechanism. In this case, the approach of Sec. 3 works: indeed we can consider these agents as ones defined in Sec. 2. But, with such definition, the concept of truthful mechanism would be quite counterintuitive.

[4] A mechanism is collusion-resistant if, for every coalition of agents, the *sum* of the utilities of colluding agents is maximized when agents in the coalition report the true.

for any $1 \leq j \leq d$, it holds $a^j(Y) \geq b^j(Y)$. The weight of edge $(\bar{a}, \bar{b})$ (if any) is $\delta(\bar{a}, \bar{b}) := \bar{a}(X) - \bar{a}(Y)$ where $X = A_{\mathbf{b}_{-i}}(\bar{a})$.

Notice that, in the spirit of verification setting, we add an edge $(\bar{a}, \bar{b})$ if valuation $\bar{a}$ can safely simulate to be a valuation $\bar{b}$. Since the mechanism is able to verify each coordinate independently, then the only case in which it is possible to safely simulate is when each coordinate is safe as well.

For the graph just defined, it holds a theorem similar to Th. 3, i.e., a social choice function is implementable with verification if and only if, for all i and all $\bar{\mathbf{b}}_{-i} \in D_{-i}$, $\mathcal{M}(A_{\bar{\mathbf{b}}_{-i}})$ does not contain negative-weight cycles. Using a technique similar to the one used in Sec. 3 we are able to prove a theorem similar to Th. 8 for objective functions of the form $\mu(b_1^1(X), \ldots, b_1^d(X), \ldots, b_m^1(X), \ldots, b_m^d(X))$ that are monotone non-decreasing in any of its inputs.

Theorem 14 *Let $\mu(\cdot)$ be any function monotone non-decreasing in each of its arguments $b_1^1(X), \ldots, b_m^d(X)$, with $X \in \mathcal{O}$ and $\bar{b}_i \in D_i$, then MAX_μ is implementable.*

Proof. Fix i and $\bar{\mathbf{b}}_{-i} \in D_{-i}$. Let us assume, for sake of contradiction, that there exists a negative-weight cycle in $\mathcal{M}_{\bar{\mathbf{b}}_{-i}}(\mathrm{MAX}_\mu)$. Assume w.l.o.g. that cycle involves k edges: $(\bar{a}_j, \bar{a}_{j+1})$ for any $j = 1, \ldots, k-1$ and $(\bar{a}_k, \bar{a}_1)$ as last edge. Moreover, let $\mathrm{MAX}_\mu(\bar{a}_j, \bar{\mathbf{b}}_{-i}) = X^j$ for any $j = 1, \ldots, k$ (not necessarily all solutions are different). For any $j = 1, \ldots, k-1$, since $\mathrm{MAX}_\mu(\bar{a}_j, \bar{\mathbf{b}}_{-i}) = X^j$ then[5]

$$\mu(a_j^1(X^j), \ldots, a_j^d(X^j), \bar{\mathbf{b}}_{-i}(X^j)) \geq \mu(a_j^1(X^{j+1}), \ldots, a_j^d(X^{j+1}), \bar{\mathbf{b}}_{-i}(X^{j+1})). \tag{5}$$

Since edge $(\bar{a}_j, \bar{a}_{j+1})$ belongs to $\mathcal{M}_{\bar{\mathbf{b}}_{-i}}(\mathrm{MAX}_\mu)$ then, for all $1 \leq k \leq d$, $a_j^k(X^{j+1}) \geq a_{j+1}^k(X^{j+1})$ and by monotonicity of μ it holds:

$$\begin{aligned}
\mu(a_j^1(X^{j+1}), a_j^2(X^{j+1}), \ldots, a_j^d(X^{j+1}), \bar{\mathbf{b}}_{-i}(X^{j+1})) &\geq \mu(a_{j+1}^1(X^{j+1}), a_j^2(X^{j+1}), \ldots, a_j^d(X^{j+1}), \bar{\mathbf{b}}_{-i}(X^{j+1})) \\
&\geq \mu(a_{j+1}^1(X^{j+1}), a_{j+1}^2(X^{j+1}), \ldots, a_j^d(X^{j+1}), \bar{\mathbf{b}}_{-i}(X^{j+1})) \\
&\geq \ldots \\
&\geq \mu(a_{j+1}^1(X^{j+1}), \ldots, a_{j+1}^d(X^{j+1}), \bar{\mathbf{b}}_{-i}(X^{j+1})).
\end{aligned} \tag{6}$$

Same reasoning holds for objective function μ applied to valuations $\bar{a}_k$, $\bar{a}_1$ in relation to solutions X^k, X^1. Unfolding Inequalities 5 and 6 we obtain:

$$\begin{aligned}
\mu(a_1^1(X^1), \ldots, a_1^d(X^1), \bar{\mathbf{b}}_{-i}(X^1)) &\geq \mu(a_1^1(X^2), \ldots, a_1^d(X^2), \bar{\mathbf{b}}_{-i}(X^2)) \\
&\geq \mu(a_2^1(X^2), \ldots, a_2^d(X^2), \bar{\mathbf{b}}_{-i}(X^2)) \\
&\geq \ldots \\
&\geq \mu(a_k^1(X^k), \ldots, a_k^d(X^k), \bar{\mathbf{b}}_{-i}(X^k)) \\
&\geq \mu(a_k^1(X^1), \ldots, a_k^d(X^1), \bar{\mathbf{b}}_{-i}(X^1)) \\
&\geq \mu(a_1^1(X^1), \ldots, a_1^d(X^1), \bar{\mathbf{b}}_{-i}(X^1)).
\end{aligned}$$

This implies that all inequalities above hold with the "=" sign. Extending Fact 5 to "compound" μ's we obtain the following.

$$X^1 = X^2 = \ldots = X^k.$$

This contradicts the hypothesis that the cycle has negative weight as all edges have weight 0. □

Observe that Th. 14 applies also to the case in which each agent i has (potentially) different dimension d_i. Indeed, slightly changing the definition of compound-verification-graph (edges are added only between pairs of valuations of the same dimension), the proof of theorem above still holds. Obviously, Cor.s 9 and 11 hold also for compound types. Last theorem gives a powerful method to solve the following problem in which we assume agents to have finite domains.

[5] We let $\bar{\mathbf{b}}_{-i}(X) := (b_1^1(X), \ldots, b_i^d(X), \ldots, b_{i-1}^1(X), \ldots, b_{i-1}^d(X), b_{i+1}^1(X), \ldots, b_{i+1}^d(X), \ldots, b_m^1(X), \ldots, b_m^d(X))$.

Definition 15 (Scheduling Unrelated Compound Machines) *There are n jobs that need to be allocated to M machines. Machines are owned by $m \leq M$ agents. Each agent owns d_i machines, with $\sum_{i=1}^{m} d_i = M$. Agent i's valuation $\bar{v}_i$ is the vector $(v_i^1, \ldots, v_i^{d_i})$. Each v_i^j is equal to the vector $(v(X_1), \ldots, v(X_{M^n}))$ for some $v \in D_i^j$ where $v(X_k)$ is the opposite of the completion time of the work assigned to jth machine controlled by agent i by the solution X_k.*

The goal is to minimize the completion time of the last assignment (the makespan).

Theorem 16 *There exists an exact truthful mechanism with verification for the problem Scheduling Unrelated Compound Machines.*

PROOF. We define $\mu(a_1^1, \ldots, a_1^{d_1}, \ldots, a_m^1, \ldots, a_m^{d_m})$ as the maximum completion time of a schedule using M machines encoded by valuations in input. Since we need to minimize μ we left to verify that μ is non-increasing in each of its inputs. This is straightforward due to the fact that valuations are the opposite of completion times. Thus, we can use $\text{MAX}_{-\mu}$ social choice function. By Th. 14 $\text{MAX}_{-\mu}$ is implementable. □

The resulting mechanism does not run in polynomial-time as the solutions to work on are in a exponential number (namely M^n) while the input consists of $M \cdot n$ values. Indeed, as argued above for scheduling unrelated machine problem, the valuations can be represented simply by the speeds of the machines (speeds are dependent from the jobs, i.e., every machine declares one speed for each job). MAX_{μ} social choice function is not able to construct, in polynomial-time, M^n different values (the solutions it has to look at) from $M \cdot n$ values.

4.1 Implementing Classical Algorithms for Compound One-Parameter Agents

All solutions presented so far could not run in polynomial-time and, in order to be implementable, use a new algorithm. In this section we turn our attention to the existence of implementable social choice functions using classical algorithms. We look for solutions that run in polynomial-time given certain conditions.

In this section we will consider compound one-parameter agents, that is, compound agents in which each coordinate is a one-parameter valuation (i.e., for agent i and solution X, $v_i(X) = -t_i \cdot w_i(X)$ where t_i is i's type and $w_i(X)$ is the work assigned to i by solution X). Observe that, for one-parameter agents any valuation (as defined in Sec. 2) is in the form $(-t_i \cdot w_i(X_1), \ldots, -t_i \cdot w_i(X_K))$ given the type t_i. Thus any vector is just represented by the type t_i and viceversa. Due to this we can consider a mechanism as collecting from any d-dimensional agent a vector of d types. Therefore, with a slight abuse of notation, we will denote by D_i^j the set of possible types that agent i can declare as jth coordinate.

We next introduce the class of smooth functions, for which there exists small α-approximation preserving set of outcomes.

Definition 17 *Fix $\varepsilon > 0$ and $\gamma > 1$. A function μ is (γ, ε)–*smooth *if, for any pair of declarations $\bar{\mathbf{b}}$ and $\tilde{\bar{\mathbf{b}}}$ such that, for any $1 \leq j \leq d$, $b_i^j \geq \tilde{b}_i^j \geq \gamma b_i^j$ for $i = 1, 2, \ldots, m$, and for all possible outcomes X, it holds that $\mu(X, \bar{\mathbf{b}}) \leq \mu(X, \tilde{\bar{\mathbf{b}}}) \leq (1 + \varepsilon) \cdot \mu(X, \bar{\mathbf{b}})$.*

For smooth functions μ, on types we are studying, we can transform any α-approximate polynomial-time algorithm A (which is not necessarily implementable with verification) into a social choice function for a constant number of d-dimensional agents which is computable in polynomial-time (with d being constant), implementable with verification and $\alpha(1 + \varepsilon)$-approximates μ.

Theorem 18 *Let A be a polynomial-time α-approximate algorithm for non-decreasing (in each input) (γ, ε)-smooth objective function $\mu(\cdot)$. If the problem involves compound one-parameter agents then, for any $\varepsilon > 0$, there exists an $\alpha(1+\varepsilon)$-approximate social choice function $A^\star$ implementable with verification. If the number of d-dimensional agents is constant and d is constant, $A^\star$ can be computed in polynomial time.*

PROOF. Let $\mathcal{O}'$ be the set of outcomes returned by A when run on bid vectors whose components are powers of γ. For m agents, $|\mathcal{O}'| = O(\max_{1 \leq i \leq m, 1 \leq j \leq d} \{\log_\gamma |D_i^j|\}^{dm})$ which is polynomial for fixed m and d. Since μ is (γ, ε)-smooth, $\mathcal{O}'$ is an $\alpha(1 + \varepsilon)-$approximation preserving set for μ.

Consider social choice function $A^\star$ that on the solution set $\mathcal{O}'$ runs MAX_μ. By Corollary 11 on compound types, $A^\star$ is implementable with verification and $\alpha(1+\varepsilon)$-approximates μ. Thus, for constant m and d, $A^\star$ is polynomial-time computable. □

Th. 18 applies also to the case in which each agent i has (potentially) different dimension d_i. In this more general case, the social choice function $A^\star$ runs in polynomial time if m is constant and $\sum_{i=1}^m d_i = O(1)$: Indeed, the set of outcomes would count $O(\max_i\{\log_\gamma |D_i|\}^{m(\Sigma_{i=1}^m d_i)})$ elements.

Above theorem does not directly imply that there exists a polynomial-time truthful mechanism with verification for smooth problems, involving compound one-parameter agents, with nondecreasing objective functions. Indeed, using the verification-graph to compute the payments would lead to polynomial-time mechanisms only whenever $|D_i|$ is polynomial in the size of the input of the mechanism (i.e., the m compound valuations declared). This would be more restrictive than requiring to have finite domains.

However, we can define a family of mechanisms indexed by m finite sets (i.e., m finite domains) $S_1, \ldots, S_m$, that is, $\mathbf{M}(S_1, \ldots, S_m) = (A^{\mathbf{S}}, P^{\mathbf{S}})$. For an instance with domains $D_1, \ldots, D_m$, one selects the mechanism $M^{\mathbf{D}} = (A^{\mathbf{D}}, P^{\mathbf{D}}) = \mathbf{M}(D_1, \ldots, D_m)$. The mechanism $M^{\mathbf{D}}$ uses knowledge on the domains D_i to precompute the verification graphs (along with their edges weights) and the differences $\bar{a}(X) - \bar{a}(A^{\mathbf{D}}(\bar{a}, \bar{\mathbf{b}}_{-i}))$ (for any $X \in \mathcal{O}$ and any $\bar{a} \in D_i$) useful to satisfy the constraints of Eq. 1. Such precomputation will make computing the payments in polynomial time possible. Thus, from Th. 18 we obtain the following.

Corollary 19 *Let A be a polynomial-time α-approximate algorithm for non-decreasing (in each input) (γ, ε)-smooth objective function $\mu(\cdot)$. If the problem involves compound one-parameter agents then, for any $\varepsilon > 0$, there exists an $\alpha(1+\varepsilon)$-approximate truthful mechanism with verification using algorithm A. For fixed domains $D_1, \ldots, D_m$, constant number of d-dimensional agents, and constant d the mechanism runs in polynomial time.*

4.1.1 Polynomial-time Scheduling Mechanisms for Compound One-Parameter Agents

Here we provide a small but nice application of technique presented in Th. 18. We assume compound agents to have a finite domain. We solve the following problem.

Definition 20 (Scheduling Related Compound Machines) *There are n jobs that need to be allocated to M machines. Machines are owned by $m \leq M$ agents. Each agent owns d_i machines, with $\sum_{i=1}^m d_i = M$. Each agent i's valuation $\bar{v}_i$ is, for each machine j he controls, the opposite of the completion time of the work assigned to machine j. That is, $\bar{v}_i = (v_i^1, \ldots, v_i^{d_i})$ and $v_i^j(\cdot) = -t_j w_j(\cdot)$ where t_j is machine j's type (inverse of its speed) and $w_j(\cdot)$ is the amount of work assigned to machine j by a given solution.*

The goal is to minimize the completion time of the last assignment (the makespan).

We present a polynomial-time truthful mechanism with verification for the *Scheduling Related Compound Machines* problem when M is constant.

Theorem 21 *Let A be a c-approximating algorithm for the make-span problem. There exists a truthful $c(1+\varepsilon)$-approximating mechanism with verification for the Scheduling Related Compound Machines using algorithm A. For fixed domains and constant number of machines the mechanism runs in polynomial time.*

Proof. The mechanism is the one given by the technique of Th. 18 and Cor. 19. We have just to show that Scheduling Related Compound Machines satisfies hypotheses of Th. 18. Firstly, make-span is $(1+\varepsilon, \varepsilon)$ smooth. We define $\mu(a_1^1, \ldots, a_1^{d_1}, \ldots, a_m^1, \ldots, a_m^{d_m})$ as the maximum completion time of a schedule using machines with speeds encoded by valuations in input. The goal is to minimize $\mu(\cdot)$ which is equivalent to $\mathrm{MAX}_{-\mu}$. We left to verify that $-\mu$ is not decreasing in each valuation. This is straightforward due to the fact that valuations are the opposite of completion times. As for the second part of the theorem we just observe that given M constant every d_i is constant as well. □

5 Mechanisms with Verification for Infinite Domains

It should be clear that verification helps in defining payments. In fact, mechanism is able to fine some kind of (detectable) lie. As stated in Sec. 2, payment is a function going from D to $\mathbb{R}$. It is well known that, for mechanisms without verification, defining payments on $\mathcal{O} \times D_{-i}$ is completely equivalent to our payment functions definition, i.e., the class of implementable (without verification) social choice functions is the same with both definitions. This consideration leads to the classical technique used, designing mechanisms without verification, for dealing with infinite domains. In this section we show that this technique, in general, cancels advantages of verification and, thus, cannot be used, at least tout-court, in the verification setting. Let us proceed more formally.

The main tool to implement the classical technique, adapted to the verification setting, is the following.

Definition 22 (allocation-verification-graph) *Let A be a social choice function. For every i and $\mathbf{b}_{-i} \in D_{-i}$, the* allocation-verification-graph $\mathcal{G}(A_{\mathbf{b}_{-i}})$ *has a node for each outcome in $\mathcal{O}$. The set of edges of $\mathcal{G}(A_{\mathbf{b}_{-i}})$ is defined as follows. For every $X, Y \in \mathcal{O}$ add an edge (X, Y) if there exists valuations $a, b \in D_i$ such that $A_{\mathbf{b}_{-i}}(a) = X$, $A_{\mathbf{b}_{-i}}(b) = Y$ and $a(Y) \geq b(Y)$ with $X \neq Y \in \mathcal{O}$. The weight of edge (X, Y) (if any) is $\delta(X, Y) := \inf_{a \in \mathcal{R}_X^Y}\{a(X) - a(Y)\}$, where $\mathcal{R}_X^Y = \{a \in \mathcal{R}_X | \exists b \in \mathcal{R}_Y \text{ s.t. } a(Y) \geq b(Y)\}$ with $\mathcal{R}_X = \{a \in D_i | A_{\mathbf{b}_{-i}}(a) = X\}$.*

This definition derives from classical allocation graphs defined for mechanisms without verification (see, for example, [16]). Classical allocation graphs have edges between all pairs of outcomes. Moreover, the weight of an edge (X, Y) is defined similarly to allocation-verification-graph: The infimum of valuation differences is taken over valuations in $\mathcal{R}_X$ instead of over valuations in $\mathcal{R}_X^Y$. Notice that, defining payments on the allocation-verification-graph, it is not necessary to assume finite domains, but it is sufficient to assume finite outcome set. We say that A is implementable (with verification) *using the allocation-verification-graph* if (A, P) is a truthful mechanism (with verification) and P is computed on the allocation-verification-graph.

We use, as witness of the fact that allocation-verification-graph does not help when verification is allowed, the following type domain.

Definition 23 *Let a and b be valuations. We say that a is* smaller *than b, in symbols $a < b$, if, for all $X \in \mathcal{O}$, $a(X) < b(X)$. Moreover, agent i's domain D_i is* strictly comparable *if for any $a, b \in D_i$ either $a < b$ or $b < a$.*

These domains, when finite, have enormous advantages from verification. Indeed, next theorem states that *all* social choice functions are implementable with verification for these finite domains.

Theorem 24 *All social choice functions on strictly comparable finite domains are implementable with verification.*

Proof. Since the verification-graph is in this case a DAG, then it has no cycles. Thus, by Th. 3, *all* social choice functions are implementable with verification. □

Therefore, for strictly comparable types, working on verification-graph implies to have full freedom on the social choice functions one wants to implement. Recall that the use of verification-graph just works for finite domains.

Analyzing the case of infinite domains, we next turn our attention to the use of allocation-verification-graph. In particular, we show that for a class of social choice function, working on particular instances of strictly comparable types, verification on allocation-verification-graph is completely useless. Consider the following definition of outcomes in relation to social choice functions.

Definition 25 *An outcome $X \in \mathcal{O}$ is fully reachable w.r.t. a social choice function A, if for any i, any $\mathbf{b}_{-i}$ and any $X \neq Y \in \mathcal{O}$ there exist valuations $a, b \in D_i$ such that $A_{\mathbf{b}_{-i}}(a) = X$ and $A_{\mathbf{b}_{-i}}(b) = Y$ with $b(X) \geq a(X)$.*

Notice that, it is, in general, possible that a social choice function A dealing with strictly comparable types admits fully reachable outcomes. It is also possible that a social choice function has neutral domains.

Definition 26 *Agent i's domain D_i is neutral w.r.t. a social choice function A if for all $X, Y \in \mathcal{O}$ s.t. (X, Y) belongs to $\mathcal{G}(A_{\mathbf{b}_{-i}})$ it holds that* $\inf_{a \in \mathcal{R}_X^Y}\{a(X) - a(Y)\} = \inf_{a \in \mathcal{R}_X}\{a(X) - a(Y)\}$.

Observe that, if D_i is neutral w.r.t. social choice function A then all edges in $\mathcal{G}(A_{\mathbf{b}_{-i}})$ have the same weight of corresponding edges in allocation graph without verification. Next theorem states that verification's advantages are canceled (on allocation-verification-graph) for social choice functions, for which all outcomes are fully reachable and all domains are neutral w.r.t. them.

Theorem 27 *Let A be a social choice function. If every outcome in $\mathcal{O}$ is fully reachable w.r.t. A and every agent domain is neutral w.r.t. A then, using the allocation-verification-graph, A is implementable with verification if and only if A is implementable without verification.*

Proof. Fix i and $\mathbf{b}_{-i}$. It suffices to show that the graph $\mathcal{G}(A_{\mathbf{b}_{-i}})$ is isomorphic to the corresponding allocation graph constructed in general settings. We start by showing that $\mathcal{G}(A_{\mathbf{b}_{-i}})$ is a complete graph (as it would be in the case verification is not allowed). Since any outcome X is fully reachable then there is an edge from any outcome $Y \neq X$ to X. On the other hand, any outcome X has an edge directed to any other outcome $Y \neq X$ (due to the fact that any Y is fully reachable). This shows that the graph $\mathcal{G}(A_{\mathbf{b}_{-i}})$ is complete. Since D_i is neutral w.r.t. A then any edge in $\mathcal{G}(A_{\mathbf{b}_{-i}})$ has the same weight as it would have in the case verification is not allowed. This concludes the proof. □

Thus, we showed that, defining payments on verification-graph allows to fully take advantages of verification settings. Defining payments on allocation-verification-graph implies that social choice functions implementable are, in general, a subset of the ones implementable on verification-graph. Actually, it turns out that the set of social choice functions implementable (with verification) on allocation-verification-graph is, in some cases, the same of the set of social choice functions implementable without verification. In a sense, this is quite obvious. Verification applies to valuations and not to outcomes and, thus, payments definition takes advantages when defined on valuations rather than on outcomes. Constructing truthful mechanisms with verification for infinite domains seems to be a compelling challenge.

Acknowledgements. We wish to thank Paolo Penna for helpful discussions.

References

[1] N. Andelman, Y. Azar, and M. Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In Volker Diekert and Bruno Durand, editors, *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS 05)*, Lecture Notes in Computer Science, pages 69–82. Springer Verlag, 2005.

[2] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the Annual IEEE Symposium on Foundations of Computing (FOCS)*, pages 482–491, 2001.

[3] V. Auletta, R. De Prisco, P. Penna, and G. Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *Proc. of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS 04)*, volume 2996 of *Lecture Notes in Computer Science*, pages 608–619. Springer Verlag, 2004.

[4] V. Auletta, R. De Prisco, P. Penna, and G. Persiano. The power of verification for one-parameter agents. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 3142 of *LNCS*, pages 171–182, 2004.

[5] V. Auletta, R. De Prisco, P. Penna, and G. Persiano. On designing truthful mechanisms for online scheduling. In *Proc. of the 12th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 3499 of *Lecture Notes in Computer Science*, pages 3–17. Springer Verlag, 2005.

[6] V. Auletta, R. De Prisco, P. Penna, G. Persiano, and C. Ventre. New constructions of mechanisms with verification. In *Proc. of the 33rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 4051 of *Lecture Notes in Computer Science*, pages 596–607. Springer, 2006.

[7] Sushil Bikhchandani, Shurojit Chatterji, Ron Lavi, Ahuva Mualem, Noam Nisan, and Arunava Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 2006. To appear.

[8] Davide Bilò, Luciano Gualà, and Guido Proietti. On the existence of truthful mechanisms for the minimum-cost approximated shortest-paths tree problem. In *Proc. of the 13th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 4056 of *Lecture Notes in Computer Science*, pages 295–309. Springer, 2006.

[9] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanims design. In *Proc. of 37th Annual ACM Symposium on Theory of Computing (STOC)*, ACM Press, pages 39–48, 2005.

[10] E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.

[11] S. Dobzinski, N. Nisan, and M. Schapira. Approximation Algorithms for Combinatorial Auctions with Complement-Free Bidders. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 610–618, 2005.

[12] A. Ferrante, G. Parlato, F. Sorrentino, and C. Ventre. Improvements for truthful mechanisms with verifiable one-parameter selfish agents. In *Proc. of 3rd Workshop on Approximation and Online Algorithms (WAOA)*, volume 3879 of *LNCS*, pages 147–160, 2005.

[13] A. Gibbard. Manipulating of voting schemes: a general result. *Econometrica*, 41(4):587–601, 1973.

[14] A. V. Goldberg and J. D. Hartline. Collusion-resistant mechanisms for single-parameter agents. In *Proc. of the 16th*, pages 620–629. SIAM, 2005.

[15] T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.

[16] Hongwei Gui, Rudolf Müller, and Rakesh V. Vohra. Dominant strategy mechanisms with multidimensional types. Technical report, 2004.

[17] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[18] Alexey Malakhov and Rakesh V. Vohra. Single and multi-dimensional optimal auctions - a network approach. Technical report, 2004.

[19] A. Mu'Alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. of 18th National Conference on Artificial intelligence (AAAI)*, pages 379–384, 2002.

[20] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

[21] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 129–140, 1999.

[22] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proc. of the ACM Conference on Eletronic Commerce (EC)*, 2000.

[23] C. H. Papadimitriou. Algorithms, games, and the Internet. In *Proc. of Annual ACM Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

[24] K. Roberts. The Characterization of Implementable Choice Rule. In J. J. Laffont, editor, *Aggregation and Revelation of Preferences*, 1979.

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

[26] Michael Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 286–293, New York, NY, USA, 2005. ACM Press.

[27] M. Satterthwaite. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[28] J. Schummer. Manipulation through bribes. *Journal of Economic Theory*, 91(3):180–198, 2000.

[29] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.