New Constructions of Mechanisms with Verification

Vincenzo Auletta, Roberto De Prisco, Paolo Penna, Giuseppe Persiano, and Carmine Ventre

Dipartimento di Informatica ed Applicazioni, Università di Salerno, Italy
Research funded by the EU through IP AEOLUS

Abstract. A social choice function A is implementable with verification if there exists a payment scheme P such that (A, P) is a truthful mechanism for verifiable agents [Nisan and Ronen, STOC 99]. We give a simple sufficient condition for a social choice function to be implementable with verification for *comparable* types. Comparable types are a generalization of the well-studied one-parameter agents. Based on this characterization, we show that a large class of objective functions μ admit social choice functions that are implementable with verification and minimize (or maximize) μ. We then focus on the well-studied case of one-parameter agents. We give a general technique for constructing *efficiently computable* social choice functions that minimize or approximately minimize objective functions that are *non-increasing* and *neutral* (these are functions that do not depend on the valuations of agents that have no work assigned to them). As a corollary we obtain efficient online and offline mechanisms with verification for some hard scheduling problems on related machines.

1 Introduction

Computations over the Internet 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) [1]. Since agents can manipulate the algorithm by misreporting their types, one has to carefully design payment functions which make disadvantageous for an agent to do so. A *mechanism* $M = (A, P)$ consists of a *social choice function* A which, on input the reported types, chooses an outcome, and a payment function P which, on input the reported types, associate a payment to every agent. Payments should guarantee that it is in the agent's interest to report his type correctly. Social choice functions A for which there exists a payment P that guarantees that the *utility* that an agent derives from the chosen outcome and from the payment he receives is maximum when this agent reports his type correctly are called *implementable* (see Sect. 1 for a formal definition of these concepts). In this case the mechanism $M = (A, P)$ is called *truthful.* The main difficulty in designing truthful mechanisms stems from the fact that the utility itself depends on the type of the agent: for instance, payments designed to "compensate" certain costs of the agents should make impossible for an agent to speculate. It is well-known that certain social choice functions cannot be implemented. This poses severe limitations on the class of optimization problems involving selfish agents that one can optimally solve (see e.g. [1,2]).

Notation. The following notations will be useful. 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 $(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_m)$. For sets $D_1, \ldots, D_m$, we let D denote the Cartesian product $D_1 \times \cdots \times D_m$ and, for $1 \leq i \leq m$, we let D_{-i} denote the Cartesian product $D_1 \times \cdots \times D_{i-1} \times D_{i+1} \times \cdots \times D_m$.

Implementation with verification. In this paper we focus on so called mechanisms with *verification* as introduced in [1] and studied in [3]. These mechanisms award payments *after* the selected outcome has been "implemented" and this implementation allows some limited "verification" on the agents' reported types. We have a finite set $\mathcal{O}$ of possible outcomes and m selfish rational agents. Agent i has a *valuation* (or *type*) v_i taken from a finite set D_i called the *domain* of agent i. A valuation v_i is a function $v_i : \mathcal{O} \rightarrow \Re$; $v_i(X)$ represents how much agent i likes outcome $X \in \mathcal{O}$ (higher valuations correspond to preferred outcomes). The valuation v_i is known to agent i only. A *social choice function* $A : D \rightarrow \mathcal{O}$ maps the agents' valuations into a particular outcome $A(v_1, \ldots, v_m)$. A *mechanism* $M = (A, P)$ is a social choice function A coupled with a *payment scheme* $P = (P_1, \ldots, P_m)$, where each P_i is a function $P_i : D \rightarrow \Re$. The mechanism elicits from each agent his valuation and we denote by $b_i \in D_i$ the *reported* valuation of agent i. On input the vector $\mathbf{b} = (b_1, \ldots, b_m)$ of *reported* valuations, the mechanism selects outcome X as $X = A(\mathbf{b})$ and assigns agent i payment $P_i(\mathbf{b})$. We assume that agents have quasi-linear utilities; more specifically, the *utility* $u_i^M(\mathbf{b}|v_i)$ of agent i when $\mathbf{b}$ is the vector of reported valuations and v_i is the type of agent i is $u_i^M(\mathbf{b}|v_i) = P_i(\mathbf{b}) + v_i(A(\mathbf{b}))$. Agents are selfish and rational in the sense that each of them will report b_i which maximizes the corresponding utility. We stress that both the outcome and the payments depend on the *reported* valuations $\mathbf{b} = (b_1, \ldots, b_m)$. In particular, for a fixed $\mathbf{b}_{-i}$, the outcome $A(\mathbf{b}_{-i}, b_i)$ is a function $A_{\mathbf{b}_{-i}}(b_i)$ of the reported valuation b_i of agent i.

The classical notion of a mechanism assumes that there is no way of verifying whether an agent reported his type truthfully (that is, whether $b_i = v_i$). Therefore, a selfish rational agent can declare any type that will maximize his utility. In some cases, though, it is reasonable to assume that the mechanism has some limited way of verifying the reported types of the agents. In this paper, we consider *mechanisms with verification* which 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 any payment.

A scenario that is often considered when dealing with selfish rational agents consists of a social choice function that has to share some work-load among the agents. In this scenario, an outcome X specifies for each agent the task that the agent has to complete. It is thus natural to assume that the valuation $v_i(X)$ of agent i reflects how much time it takes agent i to complete the task assigned to him. For example, one could have $v_i(X) = -T_i(X)$ where $T_i(X)$ is the time needed by agent i to complete the task assigned to him by X; thus higher valuations correspond to outcomes X that assign to agent i tasks that can be completed faster. In this scenario, it is natural to assume that an agent can report to be slower than he actually is and delay the completion of the task assigned to him without being caught by the mechanism (this corresponds to the case in which agent i declares b_i such that $b_i(A_{\mathbf{b}_{-i}}(b_i)) \leq v_i(A_{\mathbf{b}_{-i}}(b_i))$. On the other hand, if agent i declares to be faster that he actually is (that is, he declares b_i such that $b_i(A_{\mathbf{b}_{-i}}(b_i)) > v_i(A_{\mathbf{b}_{-i}}(b_i)))$ then agent i will complete his task at time $-v_i(A_{\mathbf{b}_{-i}}(b_i))$ instead of time $-b_i(A_{\mathbf{b}_{-i}}(b_i))$ as expected by the mechanism, given his declared valuation b_i. The mechanism will thus punish agent i by not giving him any payment. The well-studied class of one-parameter agents [4,2] corresponds to the special case in which the task assigned to an agent is described by a weight and the time needed to complete a task is proportional to its weight. In this case, the type of the agent is determined by the time it takes the agent to complete a task of unitary weight. Let us now proceed more formally.

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

In this case, $M = (A, P)$ is called a *truthful mechanism with verification.* It is easy to see that, if A is implementable with verification then 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 inequalities hold:

$$v_i(A_{\mathbf{b}_{-i}}(v_i)) + P_i(v_i, \mathbf{b}_{-i}) \geq v_i(A_{\mathbf{b}_{-i}}(b_i)) \qquad \text{if } v_i(A_{\mathbf{b}_{-i}}(b_i)) < b_i(A_{\mathbf{b}_{-i}}(b_i)) \quad (1)$$
$$v_i(A_{\mathbf{b}_{-i}}(v_i)) + P_i(v_i, \mathbf{b}_{-i}) \geq v_i(A_{\mathbf{b}_{-i}}(b_i)) + P_i(b_i, \mathbf{b}_{-i}) \text{if } v_i(A_{\mathbf{b}_{-i}}(b_i)) \geq b_i(A_{\mathbf{b}_{-i}}(b_i)) \quad (2)$$

We are interested in social choice functions A which are implementable with verification and that optimize some *objective function* $\mu(\cdot)$ which depends on the agents' valuations $\mathbf{v} = (v_1, \ldots, v_m)$. For maximization (resp., minimization) functions, we let $\text{OPT}_\mu(\mathbf{v})$ be $\max_{X \in \mathcal{O}} \mu(X, \mathbf{v})$ (resp., $\min_{X \in \mathcal{O}} \mu(X, \mathbf{v})$). An outcome $X \in \mathcal{O}$ is an α-approximation of μ for $\mathbf{v} \in D$ if the ratio betweeen $\mu(X, \mathbf{v})$ and the optimum is at most α. A social choice function A is *α-approximate* for μ if, for every $\mathbf{v} \in D$, $A(\mathbf{v})$ is an α-approximation of μ for $\mathbf{v}$. In particular, we say that social choice function A maximizes function μ if, for all $\mathbf{v}$, $A(\mathbf{v}) = \arg\max_{X \in \mathcal{O}} \mu(X, \mathbf{v})$.

Our results. We start by studying a generalization of one-parameter agents which we call *comparable* types. We give a simple sufficient condition for social choice function to be implementable with verification for comparable types and, based on this characterization, we show that a large class of objective functions μ admit social choice functions that are implementable with verification and minimize (or maximize) μ. In particular, we consider maximization (respectively, minimization) functions of the form $\mu(v_1(X), \ldots, v_m(X))$ which are monotone non-decreasing (respectively, non-increasing) in each agent valuation $v_i(X)$. Observe that VCG mechanisms [5,6,7] can only deal with particular functions of this form called *affine maximizers* and the $Q||C_{\max}$ scheduling problem is an example of an optimization problem involving a monotone non-decreasing function (thus our result applies to $Q||C_{\max}$) that is not an affine maximizer. We remark that agents with comparable types are more general than one-parameter agents. In the full version we shows a simple class of latencies (corresponding to comparable types) for which optimization is not implementable if verification is not allowed. We also *characterize* social choice functions implementable with verification for agents with *strongly comparable types*, a reach subclass of comparable types which has the well-studied one-parameter agents as a special instance. In Section 3, the focus is on *efficiently* computable social choice functions and one-parameter agents. We give a general transformation for turning any polynomial-time α-approximate algorithm A for the optimization problem with objective function μ into an $\alpha(1+\varepsilon)$-approximate social choice function $A^\star$ that is implementable with verification. If the number of agents is constant, $A^\star$ can be computed in polynomial-time and this gives immediate applications to NP-hard scheduling problems (see Section 4). Most of the proofs are omitted fromthis paper but can be found in the full version available from the authors' web pages.

Related work. Mechanisms for *one-parameter* agents have been characterized in [4,2]. Lavi, Mu'alem and Nisan [8] showed that a *weak monotonicity* condition (W-MON) characterizes *order-based* domains with range constraints and this result was extended, in a sequence of papers [9,10], to *convex domains.* These results concern mechanisms which do *not* use verification and cannot be applied to our case. We show that the "counterpart" of W-MON for mechanisms with verification (which we term WMonVer) is not always sufficient, unlike the cases considered in [8,9,10]. This gives evidence that the results about W-MON cannot be imported in mechanisms with verification. The study of social choice functions implementable with verification starts with the work of Nisan and Ronen [1], who gave a truthful $(1+\varepsilon)$-approximate mechanism for minimizing scheduling on a constant number of unrelated machines. Similar results have been obtained by Auletta *et al.* [3] for scheduling on any number of related machines (see also [11] for the online case). Also the works of [12,13] give mechanisms for agents which are verifiable.

2 Agents with Comparable Types

In this section we consider *comparable types.* The main result of this section (see Theorem 4) shows that, for any monotone non-decreasing function μ, there exists a social choice function A that maximizes μ and that is implementable with verification. In Theorem 5, we give a necessary and sufficient condition for a social choice function to be implementable with verification with respect to a subclass of comparable types which includes one-parameter agents.

Definition 2. *Let a and b be valuations. We say that a is* smaller or equal to *b, in symbols $a \leq b$, if, for all $X \in \mathcal{O}$, $a(X) \leq b(X)$. Domain D is* comparable *if for any $a, b \in D$ either $a \leq b$ or $b \leq a$.*

In this section we assume that for all i, the domain D_i of agent i is comparable. We also assume domains to have finite cardinality (even though this assumptions can be relaxed in some cases, e.g., for one-parameter agents). For fixed i and $\mathbf{b}_{-i}$, inequalities (1-2) give a system of linear inequalities with unknowns $P^x := P_i(x, \mathbf{b}_{-i})$, for $x \in D_i$. For $a, b \in D_i$ with $a \leq b$, Inequalities (1-2) are equivalent to the following two inequalities

$$P^a - P^b \geq a(A_{\mathbf{b}_{-i}}(b)) - a(A_{\mathbf{b}_{-i}}(a)) \qquad \text{if } a(A_{\mathbf{b}_{-i}}(b)) = b(A_{\mathbf{b}_{-i}}(b)), \tag{3}$$

$$P^b - P^a \geq b(A_{\mathbf{b}_{-i}}(a)) - b(A_{\mathbf{b}_{-i}}(b)). \tag{4}$$

As before, for fixed i and $\mathbf{b}_{-i}$ the two inequalities above give rise to a system of inequalities as a and b with $a \leq b$ range over D_i. This system of inequalities is compactly encoded by the following graph that is a modification of the graph introduced in [9] to study the case in which verification is not allowed.

Definition 3 (verification-graph). *Let A be a social choice function. For every i and $\mathbf{b}_{-i} \in D_{-i}$, the* verification-graph *$\mathcal{V}(\mathbf{b}_{-i})$ has a node for each type in D_i. The set of edges of $\mathcal{V}(\mathbf{b}_{-i})$ is defined as follows. For every $a \leq b$, add a directed edge (b, a) of weight $\delta_{b,a} := b(A_{\mathbf{b}_{-i}}(b)) - b(A_{\mathbf{b}_{-i}}(a))$ (encoding Inequality (4)). If $a(A_{\mathbf{b}_{-i}}(b)) = b(A_{\mathbf{b}_{-i}}(b))$, then also add directed edge (a, b) of weight $\delta_{a,b} := a(A_{\mathbf{b}_{-i}}(a)) - a(A_{\mathbf{b}_{-i}}(b))$ (encoding Inequality (3)).*

Theorem 1. *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 $\mathcal{V}(\mathbf{b}_{-i})$ 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 [15] and [9]).

We next show that there exists an interesting class of social choice functions whose verification graphs have no cycle with negative weights. As we shall prove below, these functions can be used to design optimal truthful mechanisms with verification.

Definition 4 (stable social choice function). *A social choice function A is* stable *if, for all i, for all $\mathbf{b}_{-i} \in D_{-i}$, and for all $a, b \in D_i$, with $a \leq b$, if $a(A_{\mathbf{b}_{-i}}(b)) = b(A_{\mathbf{b}_{-i}}(b))$, then we have that $A_{\mathbf{b}_{-i}}(a) = A_{\mathbf{b}_{-i}}(b)$.*

The following result is based on the fact that stable social choice functions guarantee that, if $\mathcal{V}(\mathbf{b}_{-i})$ contains a cycle, then all edges in that cycle have zero weight (see full version for a proof).

Theorem 2. *Every stable social choice function A is implementable with verification.*

We use the above result to show that it is possible to implement social choice functions which select the best outcome out of a fixed subset of possible outcomes:

Theorem 3. *For any $X_1, \dots, X_\ell \in \mathcal{O}$, let $A = MAX_\mu(X_1, \dots, X_\ell)$ be the social choice function that, on input $(b_1, \dots, b_m) \in D$, returns the solution X_j of minimum index that maximizes the value*

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

If $\mu(\cdot)$ is monotone non-decreasing in each of its arguments then A is stable.

Proof. Fix an agent i and the reported types $\mathbf{b}_{-i} \in D_{-i}$ of all the other agents. Let $a, b \in D_i$ with $a \le b$, and denote $X_{i_a} := A_{\mathbf{b}_{-i}}(a)$ and $X_{i_b} := A_{\mathbf{b}_{-i}}(b)$. To prove that A is stable we have to show that, if $a(A_{\mathbf{b}_{-i}}(b)) = b(A_{\mathbf{b}_{-i}}(b))$, then $X_{i_a} = X_{i_b}$. Observe that

$$\begin{aligned}
\mu(b_1(X_{i_b}), \dots, b_{i-1}(X_{i_b}), b(X_{i_b}), \dots, b_m(X_{i_b})) = &\quad \text{(by } a(X_{i_b}) = b(X_{i_b})\text{)} && (5)\\
\mu(b_1(X_{i_b}), \dots, b_{i-1}(X_{i_b}), a(X_{i_b}), \dots, b_m(X_{i_b})) \le &\quad \text{(definition of } A \text{ and } X_{i_a}\text{)} && (6)\\
\mu(b_1(X_{i_a}), \dots, b_{i-1}(X_{i_a}), a(X_{i_a}), \dots, b_m(X_{i_a})) \le &\quad (a \le b \text{ and } \mu \text{ non decr.}) && (7)\\
\mu(b_1(X_{i_a}), \dots, b_{i-1}(X_{i_a}), b(X_{i_a}), \dots, b_m(X_{i_a})) \le &\quad \text{(definition of } A \text{ and } X_{i_b}\text{)} && (8)\\
\mu(b_1(X_{i_b}), \dots, b_{i-1}(X_{i_b}), b(X_{i_b}), \dots, b_m(X_{i_b})). & && (9)
\end{aligned}$$

This implies that all inequalities above hold with "=". Since A chooses the optimal solution of minimal index, equality between (5) and (8) yields $i_b \le i_a$. Similarly, the equality between (7) and (6) yields $i_a \le i_b$, thus implying $X_{i_a} = X_{i_b}$.

Combining Theorem 3 and Theorem 2 we obtain the main result of this section.

Theorem 4. *Let $\mu(\cdot)$ be any function monotone non-decreasing in its arguments $b_1(X), \dots, b_m(X)$, 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 with verification.*

In the full version we exhibit a social choice function which satisfies the hypothesis of Theorem 3 (and thus is implementable with verification) but is not implementable if verification is not allowed. This shows that, for comparable types, verification does help.

If the set $\mathcal{O}$ of outcomes is very large, then social choice function A 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 the search to a suitable subset of the possible outcomes.

Definition 5 (approximation preserving). *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 $\mathbf{b}$.*

Theorem 3 implies the following.

Corollary 1. *Let $\mu(\cdot)$ be any optimization function 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 $\mathcal{O}' \subseteq \mathcal{O}$ the social choice function $APX_\mu := MAX_{X \in \mathcal{O}'}\{X\}$ 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 $X \in \mathcal{O}'$.*

Characterization. The following definition is adapted to the verification setting from the W-MON condition (see [8]) which has been proved necessary and sufficient for implementation without verification for convex domains.

Definition 6 (WMonVer). *A social choice function A is* WMonVer *for domains $D_1, \ldots, D_m$, if, for all i, for all $\mathbf{b}_{-i} \in D_{-i}$, the graph $\mathcal{V}(\mathbf{b}_{-i})$ does not contain 2-cycles of negative weight.*

Obviously, condition WMonVer is necessary for A to be implementable with verification. Next we prove that for strongly comparable types (a restriction of comparable types that includes one-parameter types) WMonVer is a necessary and sufficient condition for a social choice-function A to be implementable with verification. In the full version, we give an example of a WMonVer social choice function that is not implementable with verification for comparable types.

Definition 7 (strongly comparable types). *A domain with comparable types D_i is* with strongly comparable types *if there exists $\overline{v}_i \in \Re$ such that, for all $X \in \mathcal{O}$: (i) $a(X) \leq \overline{v}_i$, for all $a \in D_i$, and (ii) for all $a, b \in D_i$, $a(X) = b(X)$ implies $a(X) = \overline{v}_i$.*

Theorem 5. *For domains with strongly comparable types, social choice function A is implementable with verification if and only if A is WMonVer.*

3 One-Parameter Agents

In this section we present our results about *one-parameter* agents. One-parameter agents are a special case of agents with strongly comparable types, and thus Theorem 5 gives us a necessary and sufficient condition for a social choice function to be implementable with verification. In this section, the focus is on *efficiently* computable social choice functions (which will also be referred to as *algorithms*). The main result of this section (see Theorem 9) shows, for a large class of optimization functions μ (see Definitions 10 and 11), how to transform a polynomial-time α-approximate algorithm for μ into an efficiently computable social choice function that is implementable with verification for one-parameter agents and $\alpha(1+\varepsilon)$-approximates μ. The class of function μ to which our transformation applies include several classical scheduling problems (see Section 4). In Theorem 11, we give a similar result for *online* settings.

Definition 8. *[2] The valuation v_i of a* one-parameter agent *can be written as $v_i(X) = -w_i(X) \cdot t_i$, for some publicly known non-negative function $w_i(\cdot)$ and some real number $t_i \geq 0$ that is privately known to agent i.*

Observe that the valuation of a one-parameter agent is non-positive. We assume that when asked to report his type, an agents replies with a real number r_i, implying that he reports his valuation to be $b_i(X) = -w_i(X) \cdot r_i$. We consider optimization functions $\mu(X, b_1, \ldots, b_m)$ (as opposed to functions of the form $\mu(b_1(X), \ldots, b_m(X))$ of the previous section) that are non-decreasing in each valuation b_i and thus, equivalently, non-increasing in each reported type r_i.

In the rest of this section, we will show how to design social choice functions for one-parameter agents that are implementable with verification and that can be computed in polynomial time. By virtue of Theorem 5, it suffices to focus on social choice functions that are WMonVer for one-parameter agents. We first observe the following:

Fact 1 *For one-parameter agents, a social choice function A is WMonVer if and only if, for all i, $\mathbf{r}_{-i}$ there exists a* critical value *$\theta_i \in (\Re^+ \cup \infty)$ such that (i) $w_i(r_i, \mathbf{r}_{-i}) = 0$ for $r_i > \theta_i$, and (ii) $w_i(r_i, \mathbf{r}_{-i}) > 0$ for $r_i < \theta_i$.*

Notice that with a slight abuse of notation we have denoted the critical value with θ_i even though it depends on i and $\mathbf{r}_{-i}$. The above property is called *weak monotonicity* in [3], and Theorem 5 implies one of the main results in that work. *The MAX operator.* We are given a function μ and want to design a social choice function A that is implementable with verification (i.e., WMonVer) and, for a given vector $\mathbf{b}$ of declared types, returns an outcome X such that $\mu(X, \mathbf{b})$ is close to the maximum of μ over all choices of $X \in \mathcal{O}$. Moreover, we want A to be efficiently computable. A natural approach is to start from simple social choice functions and combine them together. Mu'Alem and Nisan [14] consider the following "MAX" operator:

$\text{MAX}_\mu(A_1, A_2)$ operator
- compute $X_1 = A_1(\mathbf{b})$ and $X_2 = A_2(\mathbf{b})$;
- if $\mu(X_1, \mathbf{b}) \geq \mu(X_2, \mathbf{b})$ then return X_1 else return X_2.

For minimization problems, one can simply consider a 'MIN' operator defined as $\text{MIN}_\mu(A_1, A_2) := \text{MAX}_{-\mu}(A_1, A_2)$. Notice the slight abuse of notation in using MAX_μ both with social choice functions (as in the description of the MAX_μ operator) and outcomes (as in Theorem 3) as arguments. In general, the fact that A_1 and A_2 are WMonVer does not guarantee that $\text{MAX}_\mu(A_1, A_2)$ is also WMonVer. We borrow (and adapt) the following definition.

Definition 9 ([14]). *A social choice function A is* bitonic *w.r.t. $\mu(\cdot)$ if it is WMonVer and, for every i and $\mathbf{r}_{-i}$, one of the following two conditions holds for the function $g(x) := \mu(A(x, \mathbf{r}_{-i}), (x, \mathbf{r}_{-i}))$: (i) $g(x)$ is non-increasing for $x < \theta_i$ and non-decreasing for $x \geq \theta_i$; or (ii) $g(x)$ is non-increasing for $x \leq \theta_i$ and non-decreasing for $x > \theta_i$, where θ_i is the critical value.*

The following is the main technical contribution of this sections and will be used to prove Theorem 9.

Theorem 6. *If each A_i is bitonic w.r.t. $\mu(\cdot)$ then social choice function MAX_μ $(A_1, A_2, \ldots, A_k)$:=$MAX_\mu(MAX_\mu(A_1, \ldots, A_{k-1}), A_k)$ is bitonic w.r.t. $\mu(\cdot)$ and WMonVer for one-parameter agents.*

The same results hold for the 'MIN' operator if each A_i is bitonic w.r.t. $-\mu(\cdot)$. Theorem 6 is proved by showing a connection between WMonVer social choice functions and monotone social choice functions for known single minded bidders (a special type of agents for combinatorial auctions studied in [14]).

Efficient WMonVer social choice functions. Theorem 6 provides a powerful tool for efficiently building social choice functions starting from simpler ones. In particular, we will use this result to extend Theorem 3 to a wider class of optimization functions of the form $\mu(X, b_1, \ldots, b_m)$. This allows us to deal with certain scheduling problems where the measure depends on the scheduling policy internal to the machines and therefore cannot be expressed as the machines completion times (i.e., as a function of $w_i(X) \cdot t_i$). We start by defining the notion of neutral functions.

Definition 10. *A function $\mu(\cdot)$ is* neutral *if, for every X such that $w_i(X) = 0$, it holds that $\mu(X, (b_i, \mathbf{b}_{-i})) = \mu(X, (b'_i, \mathbf{b}_{-i}))$, for every b_i, b'_i and every $\mathbf{b}_{-i}$.*

We have the following technical lemma.

Lemma 1. *Let $\mu(X, b_1, \ldots, b_m)$ be neutral and non-decreasing in each b_i, for every $X \in \mathcal{O}$. Then any algorithm returning a fixed outcome X is bitonic w.r.t. $\mu(\cdot)$.*

Theorem 7. *Let $\mu(X, b_1, \ldots, b_m)$ be neutral and non-decreasing in each b_i, for every $X \in \mathcal{O}$. Then, for any $X_1, \ldots, X_\ell \in \mathcal{O}$, the social choice function $A = MAX_\mu(X_1, \ldots, X_\ell)$ is bitonic w.r.t. $\mu(\cdot)$. Hence A is implementable with verification.*

Proof Sketch. The proof is based on the observation that a fixed outcome X_j can be seen as an algorithm returning X_j for all inputs. We show that such an algorithm is bitonic and then apply Theorem 6. $\square$

Theorem 7 above has two important consequences. First of all, we can obtain a result (similar to Theorem 4) that shows that optimization of neutral monotone functions $\mu(X, b_1, \ldots, b_m)$ for one parameter agents can be implemented with verification.

Theorem 8. *For one-parameter agents and for any function $\mu(X, b_1, \ldots, b_m)$ which is neutral and non-decreasing in each b_i, there exists a social choice function OPT_μ that maximizes $\mu(\cdot)$ and is implementable with verification.*

Another consequence is that, if we have an α-approximation preserving set of outcomes $\mathcal{O}'$ for μ, we can apply the above theorem to all outcomes $X \in \mathcal{O}'$.

This gives us a social choice function A which is implementable with verification, α-approximates μ and can be computed in time polynomial in $|\mathcal{O}'|$.

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

Definition 11. *Fix $\varepsilon > 0$ and $\gamma > 1$. A function μ is (γ, ε)–*smooth *if, for any pair of declarations $\mathbf{r}$ and $\tilde{\mathbf{r}}$ such that $r_i \leq \tilde{r}_i \leq \gamma r_i$ for $i = 1, 2, \ldots, m$, and for all possible outcomes X, it holds that $\mu(X, \mathbf{r}) \leq \mu(X, \tilde{\mathbf{r}}) \leq (1 + \varepsilon) \cdot \mu(X, \mathbf{r})$.*

For smooth, neutral functions μ 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 agents which is computable in polynomial-time, implementable with verification and $\alpha(1 + \varepsilon)$-approximates μ.

Theorem 9. *Let A be a polynomial-time α-approximate algorithm for a neutral, non-decreasing (in each b_i) (γ, ε)-smooth objective function $\mu(\cdot)$. Then, for any $\varepsilon > 0$, there exists an $\alpha(1 + \varepsilon)$-approximate social choice function $A^\star$ implementable with verification. If the number of agents is constant, $A^\star$ can be computed in polynomial time.*

Proof Sketch. 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, $|\mathcal{O}'|$ is $O(\max_i\{\log_\gamma |D_i|\}^m)$ which is polynomial for fixed m, and, 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 input $\mathbf{r}$ outputs the outcome $X \in \mathcal{O}'$ that maximizes $\mu(X, \mathbf{r})$. By Theorem 7, $A^\star$ is WMonVer and $\alpha(1+\varepsilon)$-approximates μ. Moreover, for constant m, $A^\star$ is polynomial-time computable. □

Online mechanism. A natural way of designing an *online* algorithm for scheduling problems is to iterate a "basic-step" algorithm B which, given the current assignment X, the processing requirement of the new job J and the reported types $b_1, \ldots, b_m$ (that is, the reported speed of machine i is $1/r_i$) outputs the index $B(X, J, \mathbf{b})$ of the machine to which the job must be assigned. For algorithm B, the set of outcomes $\mathcal{O}$ consists of all allocations that can be obtained from X by allocating job J to one of the m machines.

```
Algorithm B-iterated(b)
• X := ∅;
• while a new job J arrives do
•     assign job J to machine of index B(X, J, b) and modify X accordingly.
```

Observe that a basic-step algorithm B that is implementable with verification does not necessarily remains implementable with verification when iterated and we need the stronger property of stability.

Theorem 10. *If B is stable then algorithm B-iterated is stable as well.*

Therefore, by Theorem 2, B-iterated is implementable with verification. For example, Graham's [16] online greedy algorithm for $Q||C_{\max}$ can be seen as the iterated version of a simple basic stable step and thus it is implementable with verification. This property holds more in general. Consider the *greedy* algorithm which, at every step, assigns a newly arrived job to the machine that, given the current assignment of previous jobs, maximizes the increase of the objective function $\mu(\cdot)$; ties are broken in a fixed manner, and $-\mu(\cdot)$ is typically a cost function that one wishes to minimize (e.g., the L_p norm defined as $\sqrt[p]{\sum_i (w_i(X) \cdot t_i)^p}$). Then next Theorem says that greedy is stable and thus if the greedy algorithm is α-approximating for $\mu(\cdot)$, then one has a α-approximating algorithm implementable with verification.

Theorem 11. *The greedy algorithm is stable for cost functions $\mu(X, b_1, \ldots, b_m)$ that are neutral and non-decreasing in each b_i, for every $X \in \mathcal{O}$.*

4 Applications

We consider scheduling problems on related machines owned by selfish agents as in [2]. We are given a set of m *related machines* and a set of n jobs. Each job has a weight and a job can be assigned to any machine. Assigning a job to machine i makes the work w_i of that machine to increase by an amount equal to the job weight. Each machine i has a *speed* s_i, and the completion time of machine i is w_i/s_i, where w_i is the *work* assigned to machine i. In the *online* setting, jobs arrive one-by-one, the k-th job must be scheduled before next one arrives, and jobs cannot be reallocated. For an assignment X, we let $w_i(X)$ be the work that this solution assigns to machine i. Each machine i corresponds to a *selfish agent* whose valuation is $-w_i(X)/s_i = -w_i(X) \cdot t_i$ for $t_i = 1/s_i$. The speed of machine i is known to agent i only (everything else is known to the mechanism) and her valuation is the opposite of the completion time of her machine. An agent can thus misreport her speed (i.e., declare $r_i \neq t_i$). Mechanisms with verification compute, for each agent i, an associated payment and award agent i her payment if and only if all jobs assigned to machine i have been released by time $w_i(X) \cdot r_i$, where X is the outcome selected by the mechanism [1,3]. Machine i can misreport her speed and still receive her associated payment if one of the following happens: (i) the declared speed is worse (i.e., $r_i < t_i$) and jobs are released accordingly by adding some delay; (ii) the declared speed is better (i.e., $r_i < t_i$) but this makes the allocation algorithm A to compute an allocation X which does not assign any job to machine i (i.e., $w_i(X) = 0$, in which case no verification is possible).

We consider several variants of this problem depending on the optimization function adopted. All of these problems are *minimization problems* for which it is NP-hard to compute exact solutions, even for $m = 2$. The table below summarizes some of the applications of our techniques to scheduling problems. For the first three problems, no mechanism without verification can attain an approximation factor better than $2/\sqrt{3} > 1$, for all $m \geq 2$ [2]. Our upper bounds (in bold) are the first bounds on these problems which are all NP-hard to solve exactly; bounds for $Q||\sum_j \mathrm{w_j C_j}$ and $Q|\mathrm{r_j}|\sum_j \mathrm{w_j C_j}$ break the $2/\sqrt{3}$ lower bound in [2], which holds also for exponential-time mechanisms; upper bound for the L_p norm is obtained via online mechanisms based on the greedy algorithm (for $p = 2$, the bound is $1 + \sqrt{2}$). Our techniques can be used also to obtain mechanisms without verification for some graph problems (see full version).

Problem version	**Upper Bound**	
	Exp Time (any m)	**Polytime (m constant)**
$Q\|\|\sum_j \mathrm{w_j C_j}$	OPT [3]	$(1+\varepsilon)$**-approximate [Thm. 9 & [17]]**
$Q\|\mathrm{r_j}\|\sum_j \mathrm{w_j C_j}$	**OPT [Thm. 8]**	$(1+\varepsilon)$**-approximate [Thm. 9 & [17]]**
$Q\|prec, r_j\|\sum_j \mathrm{w_j C_j}$	**OPT [Thm. 8]**	$O(\log m)$**-approximate [Thm. 9 & [18]]**
L_p norm	**OPT [Thm. 8]**	$O(p)$**-competitive [Thm. 11]**

References

1. Nisan, N., Ronen, A.: Algorithmic Mechanism Design. In: Proc. of the STOC. (1999) 129–140
2. Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: Proc. of FOCS. (2001) 482–491
3. Auletta, V., De Prisco, R., Penna, P., Persiano, G.: The power of verification for one-parameter agents. In: Proc. of ICALP. Volume 3142 of LNCS. (2004) 171–182
4. Myerson, R.: Optimal auction design. Mathematics of Operations Research **6** (1981) 58–73
5. Vickrey, W.: Counterspeculation, Auctions and Competitive Sealed Tenders. Journal of Finance (1961) 8–37
6. Clarke, E.: Multipart Pricing of Public Goods. Public Choice (1971) 17–33
7. Groves, T.: Incentive in Teams. Econometrica **41** (1973) 617–631
8. Lavi, R., Mu'Alem, A., Nisan, N.: Towards a characterization of truthful combinatorial auctions. In Proc. of FOCS. (2003)
9. Gui, H., Muller, R., Vohra, R.V.: Dominant strategy mechanisms with multidimensional types. Technical report (2004)
10. Saks, M., Yu, L.: Weak monotonicity suffices for truthfulness on convex domains. In Proc. of EC 2005. 286–293
11. Auletta, V., De Prisco, R., Penna, P., Persiano, G.: On designing truthful mechanisms for online scheduling. In Proc. of SIROCCO 2005. 3–17.
12. Hajiaghayi, M.T., Kleinberg, R.D., Mahdian, M., Parkes, D.C.: Online auctions with re-usable goods. In Proc. of EC 2005. 165–174.
13. Porter, R.: Mechanism design for online real-time scheduling. In Proc. of EC 2004. 61–70
14. Mu'Alem, A., Nisan, N.: Truthful approximation mechanisms for restricted combinatorial auctions. In: Proc. of 18th AAAI. (2002) 379–384
15. Rochet, J.C.: A condition for rationalizability in a quasi-linear context. Journal of Mathematical Economics **16** (1987) 191–200
16. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell System Technical Journal **45** (1966) 1563–1581
17. Chekuri, C., Khanna, S.: A PTAS for minimizing weighted completion time on uniformly related machines. In Proc. of ICALP 2001. Volume 2076 of LNCS.
18. Chudak, F., Shmoys, D.: Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. Journal of Algorithms **30**(2) (1999) 323–343