Optimal collusion-resistant mechanisms with verification*

Paolo Penna† Carmine Ventre‡

Abstract

We present the first general positive result on the construction of *collusion-resistant* mechanisms, that is, mechanisms that guarantee dominant strategies even when agents can form arbitrary coalitions and exchange compensations (sometimes referred to as *transferable utilities* or *side payments*). This is a much stronger solution concept as compared to truthful or even group strategyproof mechanisms, and only impossibility results were known for this type of mechanisms in the "classical" model.

We describe collusion-resistant mechanisms *with verification* that return *optimal solutions* for a wide class of mechanism design problems (which includes utilitarian ones as a special case). Note that every collusion-resistant mechanism *without verification* must have an *unbounded* approximation factor and, in general, optimal solutions cannot be obtained even if we content ourselves with truthful ("non-collusion-resistant") mechanisms. All these results apply to problems that have been extensively studied in the algorithmic mechanism design literature like, for instance, task scheduling and inter-domain routing.

Key words: Game Theory, Algorithmic Mechanism Design, Transferable Utilities

1 Introduction

Computations over the Internet often involve self-interested parties which may manipulate the protocol by misreporting some information. For instance, the protocol may be designed to route packets so that no router is congested. The protocol takes its decision based on the information reported by the owners of the routers (in the Internet protocol these are the so-called Autonomous Systems [FPSS]) who might have an incentive in exaggerating their costs for forwarding packets (the protocol gives them less traffic or a higher compensation). In a *mechanism design* approach [NR] the algorithm used by the protocol is augmented with suitable payments which are supposed to motivate the parties involved (called *selfish agents*) to report the true information (their private costs or *types*).

A mechanism is said *truthful* if it guarantees that each agent can maximize her utility by truthfully reporting her type. Unfortunately, truthful mechanisms can be manipulated by colluding agents (several agents acting as a single one). This behavior is very common and thus one needs *collusion-resistant* mechanisms: no coalition of agents can raise the utilities of its members even if these can exchange compensations [Sch][GH].

*Work supported by the European Union under IST FET Integrated Project AEOLUS (IST-015964).

†Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, via Ponte Don Melillo, I-84084 Fisciano (SA), Italy. E-mail: penna@dia.unisa.it.

‡Computer Science Department, University of Liverpool, UK. E-mail: Carmine.Ventre@liverpool.ac.uk. The author is also supported by DFG grant Kr 2332/1-2 within Emmy Noether Program.

The construction of truthful mechanisms is already a challenging problem and it seems to be confined to the class of so called *utilitarian* problems (roughly speaking, the goal is to minimize the *sum* of all agents' costs). Many problems of practical interest are *not* utilitarian and truthful mechanisms for them do not exist, even if we consider approximate solutions and mechanisms running in exponential time (these inapproximability results stem from "incentive compatibility" considerations and not from computational ones).

The class of collusion-resistant mechanisms is even more limited as it essentially consists of only trivial mechanisms that output a fixed solution [Sch, GH]: these mechanisms ignore the input and thus are of no use (their approximation guarantee is *unbounded*).

1.1 Our contribution

This work presents the first general positive result on the construction of collusion-resistant mechanisms.[1] This becomes possible by considering so-called mechanisms *with verification*, that is, mechanisms that gain a limited information on the agents' types based on the observation of the computed solution. Mechanisms with verification are introduced in [NR] and further developed in [ADPP, ADP$^+$, Ven] where a generalization of the original notion is considered. Verification model we consider obeys to this more general definition (the interested reader may refer to Appendix D where we highlight the differences and show how the new model is more general than the original one) and is presented in Section 2 where we give examples and formal definitions.

The main result of this work is a general construction of mechanisms with verification which are *collusion-resistant* and return *optimal solutions* for a *generalization of utilitarian problems* (see Theorem 9). Under a mild assumption on the domain (an upper and a lower bound on the costs), these new mechanisms "kill two birds with one stone" because we can deal with non-utilitarian problems and guarantee the strongest solution concept simultaneously.

To get a feel of the implications of this result, we compare the performance of these new mechanisms with verification against the "classical" mechanism design approach, which in the sequel we denote to as mechanisms *without verification*. To this aim, we consider one of the most studied problems in (algorithmic) mechanism design, that is, scheduling selfish unrelated machines [NR, KV, MS, Gam]:

> **Without verification**, collusion-resistant mechanisms must have an *unbounded* approximation ratio even for two machines. The best known truthful ("non-collusion-resistant") mechanism is only $O(n)$-approximate, with n being the number of machines.
>
> **With verification**, there exists a collusion-resistant mechanism whose approximation factor is $(1+\varepsilon)$, for any $\varepsilon > 0$.

We give several other applications of our result to problems studied in the literature. The results are similar to those above with the notable fact that, in some cases, we obtain *polynomial-time* mechanisms, while the impossibility results (e.g., unbounded approximation ratio for collusion-resistant mechanisms) holds for mechanisms that are computationally unbounded (see Table 1).

Our construction relies on payments similar to those used in VCG mechanisms [Vic, Cla, Gro], and it gives an explicit and computationally-efficient way of constructing the whole mechanism (see Corollary 12). A well-known drawback of VCG mechanisms is that of being far from frugal (see, e.g.,

[1] Goldberg and Hartline [GH] describe several mechanisms that achieve a weaker "collusion-resistant with high probability" condition for a simple "unstructured" problem in which identical goods are sold to the users.

[AT2]) and therefore our mechanisms are not frugal as well. We note however that, when applied to the *two-values* scheduling problem in [LS], our mechanisms pay an amount which is comparable to what their mechanism pays in the worst case. In this work we do not consider frugality issues directly (that is, how much the mechanism overpays the agents) since our aim is that of giving a *general* technique for the construction of collusion-resistant mechanisms. The optimality of the payments is an important issue *in general* since even truthful (not necessarily VCG) mechanisms must have large payments for rather simple problems [ESS, KKT].

The conditions for obtaining our mechanisms are stated in terms of certain *algorithmic properties* so that the design of the entire mechanism reduces to the design of an algorithm that fulfills these conditions. Theorem 9 says that every algorithm that is optimal for what we call a *weakly utilitarian* cost function (see Definition 7) can be turned into a collusion-resistant mechanism with verification. The same result can be obtained if the algorithm satisfies a condition independent from the optimality (see Theorem 15). This poses a new algorithmic challenge: designing *computationally-efficient* algorithms that obey these conditions. This work provides the first positive results in this direction.

1.2 Related Work

Every utilitarian problem admits an exact truthful mechanism (without verification) via VCG mechanisms [Vic, Cla, Gro]. The inapproximability of non-utilitarian problems has been extensively studied [NR, AT1, CKV, MS, Gam, LS]. These works prove lower bounds on the approximation ratio that *any* truthful mechanism without verification must have on the problem(s). These lower bounds hold for exponential-time mechanisms and for finite domains, a special case of the bounded domains studied here.

Mechanisms with verification are introduced in [NR] in order to overcome these impossibility results and the limitations of VCG mechanisms. The model of verification is then generalized and truthful constructions are given for simple ("one-dimensional") domains in [ADPP, ADP$^+$] where they attain provably better approximations for specific problems. Exact mechanisms with verification for a very general class of problems and agents bidding from finite domains is given in [Ven] and extended to (not finite) bounded domains in [PV]. Interestingly enough, the technique in [PV] consists in designing collusion-resistant mechanisms with verification for very simple problems with single-parameter ("binary") agents, and then view a "multidimensional" agent as a coalition of these agents. The construction of collusion-resistant mechanisms therein requires, in addition to the optimality of the algorithm, certain additional technical conditions. In fact, that method in [PV] does not give collusion-resistant mechanisms even for the simple utilitarian path-auction problem described below as Problem 1 (see Appendix B for the details), while the method in this work deals with a *generalization* of utilitarian problems and general domains (and not just single-parameter agents). These considerations show how our technique is of a completely different flavor because of its wider applicability and provably generality.

Impossibility results for collusion-resistant mechanisms have been proved in [Sch] where, among others, it has been shown that for "sufficiently rich" domains, every such mechanism must output a *fixed* solution. The result is actually stronger since it even applies to *bribeproof* mechanisms, that is, mechanisms that resist to coalitions of two agents (one bribing another for misreporting the type). Analogous results are also obtained in [GH] for single-parameter domains and even for randomized mechanisms. These negative results do not apply to group strategyproof mechanisms which do not consider compensations among the agents (see discussion in [GH]).

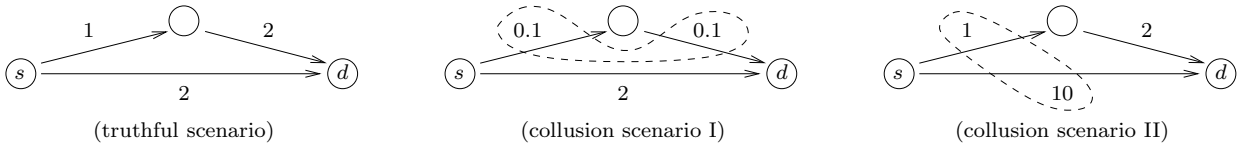

Figure 1: Collusion and verification.

Roadmap. We introduce mechanisms with verification in Section 2, where we give a few examples followed by the formal model (Section 2.1). The construction of collusion-resistant mechanisms is given in Section 3. We present various applications of this result in Section 4. In Section 5 we give the algorithmic approach to the design of collusion-resistant mechanisms with verification. For sake of readability, some of the proofs are postponed to the appendix.

2 Examples and the model

We begin by giving three examples of widely-studied problems in the mechanism design literature. These problems have increasing difficulty and they will serve to introduce the formal model of mechanisms with verification.

Problem 1 (Path auction [NR]) Suppose you want to route traffic through a network and, naturally, you want to pick the path of minimal cost connecting the source to the destination in a given network. This is modelled by the following *path auction* problem. Here, each link of the network corresponds to a router that is owned by a selfish agent. The cost for using link i is known only to agent i and it is equal to some nonnegative value t_i which is the time needed by the link to forward traffic. This is also the cost that agent i incurs if her link is used to route traffic. VCG mechanisms [Vic, Cla, Gro] provide an elegant solution to this problem: Each selected link is payed its "marginal contribution", that is, the difference between the best *alternative* path minus the cost of the chosen path without counting the link itself. For the simple network in Fig. 1 in the truthful scenario, the lower path is chosen, the lower link is payed an amount 3, and the other two non-selected links receive no money. Since true cost for the lower link is 2, the corresponding agent derives a utility of $3-2=1$. We describe two examples in which two colluding agents (shown in the dashed area) misreport their costs and improve their utilities, compared to the truthful scenario. In both cases, the mechanism chooses the upper ("wrong") path and the agent whose link costs 1 is offering a suitable compensation to the other colluding agent. In the collusion scenario I, each of the chosen link is payed an amount $2-0.1$ and a compensation equal to 0.2 makes both the two utilities strictly positive (while in the truthful scenario these agents are not payed). Similarly, in collusion scenario II the agent whose link costs 1 is payed $10-1$ by the mechanism and offering a compensation equal to 2 makes the utilities of both of them largely better than in the truthful scenario. Note that there are several other possible collusion scenarios, including the one in which all three agents collude. ◇

The above problem has a rather simple ("one-dimensional") structure since the cost of each agent is either 0, when her link is not selected, or a private value otherwise. In the next problem agents have a more complex ("multidimensional") type.

Problem 2 (Scheduling unrelated machines [NR]) We have a number of different tasks to execute and n machines are available for this purpose. Each machine is of a different *type*, meaning that a machine of type t_i can execute task j in t_i^j time steps. Each machine is owned by a selfish agent who is the only one to know the type of her machine. Agents can misreport the execution time of the jobs, that is, the type of her machine. Each allocation of the tasks to the machines specifies a cost of the corresponding agent that is the completion time of her machine (how much time the machine has to work for the "system"). The objective is to pick an allocation minimizing the *makespan*, that is, the maximum completion time over all machines. ◇

The above problem turns out to be a special case of the following one for which we give only a very high-level description.

Problem 3 (Inter-domain routing [FPSS, MS]) We are given an undirected graph where each node represents an Autonomous System (AS) of which the Internet is comprised. Traffic originating at different sources towards a common destination must be routed through a confluent tree (this condition is imposed by the fact that loops cannot occur). Every such tree specifies a cost for each AS which depends on the amount of traffic to be forwarded and on the next hop in the routing. In particular, each AS has a *per-packet cost* associated to each of its neighbors. The per-packet cost is determined by a number of parameters "internal" to the AS and it is essentially the latency experienced by a packet traveling "inside" the AS. Two objectives have been considered: minimizing the *total cost* [FPSS] or minimizing the *maximum cost* [MS]. ◇

We now want to move towards a fully general model which turns out to be a generalization of the one introduced in [NR] (see Appendix D for a discussion). Each feasible solution, say x, specifies some cost which is the time $t_i(x)$ that agent i has to work if solution x is implemented. Note that the function t_i is not publicly known and thus agent i could report a (possibly different) cost function b_i, which in the auction theory terminology is called the bid of agent i. Now observe that the solution is computed according to the bid b_i and, if the computed solution x_{chosen} is such that $b_i(x_{chosen}) < t_i(x_{chosen})$, the mechanism can detect that agent i misreported her type. Indeed, the cost $t_i(x_{chosen})$ is the (minimal) time needed by i to provide her "contribution" to the solution (time to forward traffic, time to execute tasks, etc.). Thus agent i is *caught lying* because she is not able to provide her contribution within the time $b_i(x_{chosen})$ that she reported to the mechanism. Observe that there are still many ways in which an agent can lie to the mechanism without being caught lying. For instance, in the scheduling problem, she can always report *higher* execution times for the tasks that the mechanism assigns to her machine. Moreover, the mechanism cannot gain any extra information on the task that are *not* allocated to this agent/machine. In the path-auction problem above, instead, verification would be effective only in the collusion scenario I and *not* in the collusion scenario II.

2.1 Model

We have a finite set of feasible solutions $\mathcal{S}$. The set of possible types or *domain* includes real-valued functions $t_i(x)$, where x is any feasible solution. We focus on *arbitrary bounded domains*, that is, there exist two constants $t_{\inf}$ and $t_{\sup}$ such that,

$$t_{\inf} \leq t_i(x) \leq t_{\sup},$$

for every type t_i and every x. This is the only restriction we make and thus each agent can report *any* type b_i such that $t_{\inf} \leq b_i(x) \leq t_{\sup}$, for all feasible solutions x.

A *mechanism* is a pair (A, p) where A is an algorithm (or social choice function) and p is a suitable payment function. The input to the mechanism are the types *reported* by each agent, that is, the *bids* $\mathbf{b} = (b_1, \ldots, b_n)$ with b_i being the type reported by agent i. The mechanism returns a feasible solution $A(\mathbf{b})$ and a payment associated to every agent which is equal to $p_i(\mathbf{b})$. Note that p is a vector of n functions $p_1(), \ldots, p_n()$, where the domain consists of all possible bid vectors $\mathbf{b}$ as above.[2]

Definition 4 *Given a mechanism (A, p) and bids $\mathbf{b} = (b_1, \ldots, b_i, \ldots, b_n)$, we say that agent i is* caught lying *if her type t_i and her bid b_i are such that $b_i(x) < t_i(x)$, where $x = A(\mathbf{b})$. A* mechanism with verification *(A, p) provides no payment to agent i if she is caught lying, and it provides her a payment $p_i(\mathbf{b})$ otherwise.*

An agent naturally derives a *utility* which is the difference between the money she receives minus the cost she associates to the chosen solution. In the case of a mechanism with verification (A, p) this is equal to

$$\text{utility}_i(\mathbf{b}) := \left\{ \begin{array}{ll} p_i(\mathbf{b}) & \text{if } i \text{ is not caught lying} \\ 0 & \text{otherwise} \end{array} \right\} - t_i(\mathbf{b}).$$

On the contrary, in a *mechanism without verification* (A, p) each agent receives always her associated payment and thus the utility of agent i is equal to $p_i(\mathbf{b}) - t_i(\mathbf{b})$.

We say that a mechanism satisfies the *voluntary participation* condition if truthtelling agents have always a nonnegative utility. We consider mechanisms for which truthtelling is a dominant strategy for every possible coalition of agents that can exchange side payments:

Definition 5 ([GH]) *A mechanism (with verification) is* collusion-resistant *if, for any coalition of agents and any bid of agents not in the coalition, the sum of the utilities of the agents in the coalition is maximized when all agents in the coalition are truthtelling.*[3]

Since the above condition must hold for *all* possible bids of agents outside the coalition under consideration, one can restrict the analysis to the case in which these agents are actually truthtelling (see Appendix A.1 for a proof of the following known fact):

Fact 6 *A mechanism is collusion-resistant if and only if for any true type vector $\mathbf{t}$ and for any coalition C, it holds that*

$$\sum_{i \in C} \text{utility}_i(\mathbf{t}) \geq \sum_{i \in C} \text{utility}_i(\mathbf{b}) \tag{1}$$

for any type vector $\mathbf{b}$ in which some agents in C are lying and all other agents bid as in $\mathbf{t}$.

[2] Observe that the domain of agent i is R^k with $R = [t_{\inf}, t_{\sup}]$ since each b_i can be seen as a k-tuple $b_i = (b_i(x_1), \ldots, b_i(x_k))$ where $\{x_1, \ldots, x_k\}$ is the set of feasible solutions. The domain is the cross-product of all agents domains, that is, a subset of R^{nk}.

[3] These mechanisms are also termed t-truthful to denote the fact that they resist to coalitions of up to t agents [GH]. Truthful mechanisms are the special case of them for which $t = 1$ and the negative results in [GH, Sch] are proved for 2-truthful mechanisms or even weaker notions.

A *cost function* is a function cost() that associates, to every feasible solution, a real-valued cost which depends on the costs (types) of all agents. Naturally, the cost associated to a solution cannot decrease if the cost of one agent increases and all others remain the same. That is, for every feasible solution x, for every vector $\mathbf{b}$, and every b'_i such that $b'_i(x) > b_i(x)$, it holds that

$$\text{cost}(x, \mathbf{b}) \leq \text{cost}(x, (b'_i, \mathbf{b}_{-i}))$$

where $(b'_i, \mathbf{b}_{-i}) = (b_1, \ldots, b_{i-1}, b'_i, b_{i+1}, \ldots, b_n)$ is the vector obtained from $\mathbf{b}$ by changing b_i into b'_i. In such a case we say that the cost function is *monotone nondecreasing*. A cost function satisfies the *boundness* condition if, for every $t_{\inf}$ and $t_{\sup}$ as above, there exist $\inf \text{cost}$ and $\sup \text{cost}$ such that

$$\inf \text{cost} \leq \text{cost}(x, \mathbf{b}) \leq \sup \text{cost},$$

for all $\mathbf{b}$ such that $t_{\inf} \leq b_i(x) \leq t_{\sup}$.

3 Construction of optimal collusion-resistant mechanisms

In this section we present the main positive result of this work which is a class of collusion-resistant mechanisms with verification. We shall see that any algorithm minimizing a weakly utilitarian cost function (see below) can be turned into a collusion-resistant mechanism with verification.

Definition 7 *A cost function* $\text{cost}^+()$ *is* weakly utilitarian *if there exist* $\alpha_i > 0$*, for* $i = 1, \ldots, n$*, and an arbitrary monotone nondecreasing cost function* cost() *satisfying the boundness condition such that*

$$\text{cost}^+(x, \mathbf{t}) = \text{cost}(x, \mathbf{t}) + \sum_i \alpha_i \cdot t_i(x) \tag{2}$$

for every x *and* $\mathbf{t}$*.*

Our goal is to construct collusion-resistant mechanisms that minimize any given weakly utilitarian cost function. We shall prove a stronger result; these mechanisms exist for the following algorithms:

Definition 8 *We say that an algorithm* A *is* weakly utilitarian *if there exists some* $\mathcal{S}' \subseteq \mathcal{S}$ *such that, for all type vectors* $\mathbf{t}$*, it holds that*

$$A(\mathbf{t}) \in \arg\min_{x \in \mathcal{S}'} \text{cost}^+(x, \mathbf{t}), \tag{3}$$

for some weakly utilitarian cost function $\text{cost}^+()$*.*

This is the main result of this section:

Theorem 9 *Every weakly utilitarian algorithm over an arbitrary bounded domain admits a collusion-resistant mechanism with verification. The mechanism satisfies the voluntary participation condition.*

Before proving this theorem, we wish to give an idea of the power of such a result. First, *utilitarian* algorithms are a special case of weakly utilitarian algorithms since they concern with the case $\text{cost}^+(x, \mathbf{t}) = \sum_i \alpha_i \cdot t_i(x)$, and thus we have exact collusion-resistant mechanisms with verification for a generalization of utilitarian problems. Second, one can easily obtain approximations for more problems in which the objective is to minimize an arbitrary cost function cost(). This can be done by "tuning" the constants α_i so that the inequalities

$$\text{cost}(x, \mathbf{t}) \leq \text{cost}^+(x, \mathbf{t}) \leq c \cdot \text{cost}(x, \mathbf{t})$$

hold for all x and $\mathbf{t}$, with $c \geq 1$. The resulting weakly utilitarian algorithm A returns a c-approximation for cost() and so does the mechanism in Theorem 9.

3.1 The mechanism and its analysis

In this section we show that, for any weakly utilitarian algorithm A, there exist payments p such that (A, p) is a collusion-resistant mechanism with verification. The idea is to relate the utility of an agent to the global cost and use the algorithm "optimality" to reduce the analysis to the case in which at least one agent in the coalition is caught lying.

We give an explicit formula for the payments which are defined as follows:

$$p_i(\mathbf{b}) := \hbar - \frac{1}{\alpha_i} \left(\sum_{j \neq i} \alpha_j \cdot b_j(A(\mathbf{b})) + \text{cost}(A(\mathbf{b}), \mathbf{b}) \right) \tag{4}$$

with $\hbar$ being a suitable *constant* to be given below. These payments actually "relate" the utility of each agent to the global cost function:

Lemma 10 *For every* $\mathbf{b}$ *and* $\mathbf{t}$ *such that agent* i *is not caught lying, it holds that*

$$\text{utility}_i(\mathbf{b}) \leq \hbar - \frac{\text{cost}^+(A(\mathbf{b}), (t_i, \mathbf{b}_{-i}))}{\alpha_i}.$$

Moreover, for $\mathbf{b} = \mathbf{t}$*, the inequality holds with '='.*

Proof. Since agent i is not caught lying, she receives her payment and her utility is

$$\begin{aligned}
\text{utility}_i(\mathbf{b}) &= p_i(\mathbf{b}) - t_i(A(\mathbf{b})) \\
&= \hbar - \frac{1}{\alpha_i} \left(\sum_{j \neq i} \alpha_j \cdot b_j(A(\mathbf{b})) + \text{cost}(A(\mathbf{b}), \mathbf{b}) \right) - t_i(A(\mathbf{b})) \\
&= \hbar - \frac{1}{\alpha_i} \left(\alpha_i \cdot t_i(A(\mathbf{b})) + \sum_{j \neq i} \alpha_j \cdot b_j(A(\mathbf{b})) + \text{cost}(A(\mathbf{b}), \mathbf{b}) \right).
\end{aligned}$$

Since i is not caught lying the inequality $t_i(A(\mathbf{b})) \leq b_i(A(\mathbf{b}))$ holds. Because cost() is monotone nondecreasing, we have that $\text{cost}(A(\mathbf{b}), (t_i, \mathbf{b}_{-i})) \leq \text{cost}(A(\mathbf{b}), \mathbf{b})$ which, combined with (5), implies

$$\text{utility}_i(\mathbf{b}) \leq \hbar - \frac{\text{cost}^+(A(\mathbf{b}), (t_i, \mathbf{b}_{-i}))}{\alpha_i}.$$

This proves the first part of the lemma. For $\mathbf{b} = \mathbf{t}$, the quantity in (5) is precisely $\hbar - \text{cost}^+(A(\mathbf{t}), \mathbf{t})/\alpha_i$, and thus the second part of the lemma holds. □

The second main ingredient is to show that truthtelling minimizes the "global cost", when no agent is caught lying. The proof of this lemma adapts an idea in [Ven].

Lemma 11 *Every weakly utilitarian algorithm A satisfies the inequality*

$$\text{cost}^+(A(\mathbf{t}), \mathbf{t}) \leq \text{cost}^+(A(\mathbf{b}), (t_i, \mathbf{b}_{-i})).$$

for all $\mathbf{t}$ *and* $\mathbf{b}$ *such that no agent is caught lying.*

PROOF. Consider any two vectors $\mathbf{t}$ and $\mathbf{b}$ such that no agent is caught lying. That is, $t_j(A(\mathbf{b})) \leq b_j(A(\mathbf{b}))$ for all $1 \leq j \leq n$. Since $\alpha_i > 0$ and cost() is nondecreasing, it holds that cost$^+$() is nondecreasing as well. This implies

$$\text{cost}^+(A(\mathbf{b}), \mathbf{t}) \leq \text{cost}^+(A(\mathbf{b}), (t_i, \mathbf{b}_{-i})).$$

Since A is a weakly utilitarian algorithm, condition (3) says that

$$\text{cost}^+(A(\mathbf{t}), \mathbf{t}) \leq \text{cost}^+(A(\mathbf{b}), \mathbf{t}).$$

The lemma follows from these two inequalities. □

We are now in a position to prove the main result of this section:

PROOF OF THEOREM 9. Consider an arbitrary coalition C and let $\mathbf{t}$ and $\mathbf{b}$ be defined as in Fact 6. If no agent in the coalition is caught lying, then Lemma 10 and Lemma 11 imply that the utility of each single agent is better in the truthful case, that is

$$\begin{aligned} \text{utility}_i(\mathbf{b}) &\leq \hbar - \frac{\text{cost}^+(A(\mathbf{b}), (t_i, \mathbf{b}_{-i}))}{\alpha_i} \\ &\leq \hbar - \frac{\text{cost}^+(A(\mathbf{t}), \mathbf{t})}{\alpha_i} \\ &= \text{utility}_i(\mathbf{t}) \end{aligned}$$

which clearly implies (1).

Let us then assume that at least one agent in the coalition is caught lying. In this case, we make use of the following simple observation. From the boundness condition of cost() and because the domain is bounded, there exist two constants κ_{inf} and $\kappa_{\text{sup}} \geq \kappa_{\text{inf}}$ such that $\hbar - \kappa_{\text{sup}} \leq p_i(\mathbf{b}) \leq \hbar - \kappa_{\text{inf}}$ for all bid vectors $\mathbf{b}$. (See Appendix A.2 for the details proving these inequalities.) Since at least one agent $j \in C$ does not receive any payment for $\mathbf{b}$, by setting $\ell = \min\{0, t_{\text{inf}}\}$, we have

$$\sum_{i \in C} \text{utility}_i(\mathbf{b}) \leq (|C| - 1)(\hbar - \kappa_{\text{inf}} - t_{\text{inf}}) - \ell$$

Since in $\mathbf{t}$ all agents get their payments, we have

$$\sum_{i \in C} \text{utility}_i(\mathbf{t}) \geq |C|(\hbar - \kappa_{\text{sup}} - t_{\text{sup}}).$$

In order to guarantee (1) it is enough to set

$$\hbar := |C|(\kappa_{\sup} - \kappa_{\inf} + t_{\sup} - t_{\inf}) + \kappa_{\inf} + t_{\inf} + \ell$$

for $|C| = n$. This value of $\hbar$ guarantees that (1) holds for any C. Fact 6 implies that the mechanism (A, p) is collusion-resistant.

Finally, the voluntary participation condition can be guaranteed by setting $\hbar \geq t_{\sup} + \kappa_{\sup}$ since this inequality yields $p_i(t_i, \mathbf{b}_{-i}) \geq \hbar - \kappa_{\sup} \geq t_{\sup} \geq t_i(x)$ for all x. □

Note that the payments in (4) can be computed in polynomial time if the algorithm is polynomial-time computable. Hence, the following holds.

Corollary 12 *Every* polynomial-time *weakly utilitarian algorithm over an arbitrary bounded domain admits a* polynomial-time *collusion-resistant mechanism with verification. The mechanism satisfies the voluntary participation condition.*

4 Applications

We apply our technique to a number of mechanism design problems studied in the literature. We start by studying *min-max* mechanism design problems, that is, problems in which the objective is to compute a solution x minimizing

$$\max_i t_i(x)$$

and agents' costs $t_i(x)$ are always non-negative, that is, $t_{\inf} \geq 0$.[4] Min-max problems have been widely studied in the algorithmic mechanism design literature and most of the known negative results on truthful mechanisms concern with this class of problems [NR, AT1, MS, CKV, Gam].

We next show how simple it is to obtain $(1 + \varepsilon)$-approximation mechanisms, for every $\varepsilon > 0$, using the result in the previous section. Consider the weakly utilitarian algorithm minimizing

$$\text{cost}^+(x, \mathbf{t}) = \max_i t_i(x) + \varepsilon \cdot \frac{\sum_i t_i(x)}{n}. \tag{5}$$

Simple calculations show that the optimum for this weakly utilitarian cost function is a $(1 + \varepsilon)$-approximation for the corresponding min-max problem (see Appendix A.3 for the details). The following is thus a simple corollary of Theorem 9.

Corollary 13 *Every min-max problem over an arbitrary bounded domain admits a* $(1+\varepsilon)$*-approximation collusion-resistant mechanism with verification, for every* $\varepsilon > 0$*. The mechanism satisfies the voluntary participation condition.*

This general result immediately applies to a number of problems which do not even admit optimal or $(1 + \varepsilon)$-approximate truthful ("non-collusion-resistant") mechanisms without verification. These problems include the scheduling unrelated machines (Problem 2) and the workload minimization in inter-domain routing (Problem 3). It is well-known that both problems are min-max problems [MS] and the bounded domain condition holds in practice: for instance, it is unreasonable to consider the possibility that routing packets over the Internet takes *thousands years*.

Problem	Without verification lower bounds		With verification upper bounds
	truthful	collusion-resistant	collusion-resistant
Inter-domain routing			
lowest cost (utilitarian)	$\boxed{\text{exact}}$	∞	$\boxed{\text{exact}}$
workload (min-max)	2	∞	$1+\varepsilon$
Scheduling selfish machines			
unrelated (min-max)	$1+\Phi \approx 2.618$	∞	$1+\varepsilon$
related* (min-max)	exact	∞	$\boxed{1+\varepsilon}$
weighted sum$^\#$	$2/\sqrt{3}$	∞	$\boxed{1+\varepsilon}$

Table 1: Known lower bounds and new upper bounds derived in this work. Framed bounds can be achieved in polynomial-time. The mechanism marked by '$*$' (resp.'#') is an FPTAS (resp. PTAS) for any fixed number of agents. (However, claimed bounds can be derived in general though not in polynomial time.)

All the applications of our technique are not confined to min-max problems. All mechanisms we obtain by applying our technique are shown in Table 1.

We next argue about the the lower bounds in the table. The results in [Sch] say that the only collusion-resistant mechanisms without verification are those that output a *fixed* solution. This implies an *unbounded* approximation ratio which holds even for mechanisms that resist to coalitions of only two agents.The existing lower bounds on truthful ("non-collusion-resistant") mechanisms proved in the literature also hold for exponential running-time mechanisms and bounded domains. That is, the proof exhibits a bounded domain for which the lower bound holds even for inefficient mechanisms (see results on unrelated machines [NR, KV, MS, Gam], workload minimization in inter-domain routing [MS, Gam], and weighted sum schedule [AT1]).

We next argue about the upper bounds in the table. For min-max problems they are a direct consequence of Corollary 13. (We give further details respectively in Appendix C.1 for scheduling unrelated machines problems and C.4 for workload minimization in inter-domain routing.)

As for the lowest total cost minimization problem in inter-domain routing (see Example 3 and Section C.4) we observe that it belongs to the class of so-called utilitarian problems [FPSS]. The polynomial-time exact algorithm for the problem in [FPSS], which the authors give to obtain a truthful mechanism without verification, can be also used in our construction for obtaining an exact polynomial-time collusion-resistant mechanism with verification. The key observation is that such an algorithm is, by definition, a weakly utilitarian algorithm and thus Corollary 12 can be applied.

A second efficient collusion-resistant mechanism can be obtained for another scheduling problem studied in [AT1] in which the objective is to minimize the makespan on related machines. For a constant number of machines, we obtain the first FPTAS collusion-resistant mechanism with verification for this problem (no positive result on polynomial-time mechanisms was known). We use an idea suggested in [AAS]. We compute a polynomial number of feasible solutions $\mathcal{S}'$ (this step is independent of the bids) such that the best solution in $\mathcal{S}'$ has a makespan which is a $(1+\varepsilon')$-approximation of the optimum in $\mathcal{S}$. (This step relies on an FPTAS for the makespan, e.g., the

[4]With such an assumption min-max problems preserve the essence of their *raison d'être* as many of them involving agents supporting costs are NP-hard.

one in [HS].) We then select an optimal solution for the weakly utilitarian cost function (5) among the computed ones in time $|\mathcal{S}'|$. (We remark that the size of $\mathcal{S}'$ is also polynomial in $1/\varepsilon'$.) This solution will be a $(1+\varepsilon)\cdot(1+\varepsilon')$-approximation of the optimum in $\mathcal{S}$. Since $\varepsilon, \varepsilon' > 0$ can be chosen arbitrarily small, Corollary 12 implies the existence of an FPTAS collusion-resistant mechanism with verification. (Details can be found in Appendix C.2.)

Another efficient mechanism can be derived for a different scheduling problem again on related machines studied in [AT1] in which the objective is to minimize the weighted sum of all jobs completion times. We derive a PTAS collusion-resistant mechanism with verification for this problem. Also this mechanism is based on the idea of computing a subset of "good" solutions $\mathcal{S}'$ and output out of it the optimal solution for a carefully designed weakly utilitarian cost function. (The computation of $\mathcal{S}'$ relies on the PTAS for the weighted completion time in [CK].) Similar arguments also show that when the number of machines is not constant is it possible to have a $(1+\varepsilon)$-approximate collusion-resistant mechanisms with verification though not running in polynomial-time. (Details can be found in Appendix C.3.)

5 An algorithmic approach

We next present generalize the result in Theorem 9 and prove the following type of result. If the algorithm satisfies a "cost monotonicity" condition, then it can be used to obtain a collusion-resistant mechanism with verification.

Definition 14 *An algorithm A is* cost monotone *if there exists a weakly utilitarian cost function* $\text{cost}^+()$ *such that the following holds. For any* $\mathbf{t}$ *and* $\mathbf{b}$ *such that* $t_i(A(\mathbf{b})) \leq b_i(A(\mathbf{b}))$*, it holds*

$$\text{cost}^+(A(\mathbf{t}), \mathbf{t}) - \text{cost}^+(A(\mathbf{b}), \mathbf{b}) \leq \alpha_i \cdot (t_i(A(\mathbf{b})) - b_i(A(\mathbf{b}))),$$

where α_i is the constant in the definition of weakly utilitarian cost function $\text{cost}^+()$.

Lemma 11 says that every weakly utilitarian algorithm is cost motonote. Interestingly, a proof similar to that of Theorem 9 proves the following (more general) result.

Theorem 15 *Every (polynomial-time) cost monotone algorithm over an arbitrary bounded domain admits a (polynomial-time) collusion-resistant mechanism with verification satisfying the voluntary participation condition.*

Proof. If no agent is caught lying, then by (5) we have

$$\begin{aligned} \text{utility}_i(\mathbf{b}) &= \hbar - \frac{\text{cost}^+(A(\mathbf{b}), \mathbf{b})}{\alpha_i} + b_i(A(\mathbf{b})) - t_i(A(\mathbf{b})) \\ &\leq \hbar - \frac{\text{cost}^+(A(\mathbf{t}), \mathbf{t})}{\alpha_i} \\ &= \text{utility}_i(\mathbf{t}). \end{aligned}$$

where the inequality follows from the fact that A is cost monotone. This reduces the analysis to the case in which there is at least one agent in the coalition under consideration who is caught lying. This case is the same as in Theorem 9. □

We note that we can further extend the result since we do not need the "monotonicity" of cost() when defining the weakly utilitarian cost function $\text{cost}^+()$ (see Definitions 7 and 14).

Acknowledgements. The authors are grateful to Riccardo Silvestri for several useful comments on an earlier version of this work.

References

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

[ADP+] V. Auletta, R. De Prisco, P. Penna, G. Persiano, and C. Ventre. New constructions of mechanisms with verification. In *Proc. of International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 596–607, 2006.

[ADPP] V. Auletta, R. De Prisco, P. Penna, and G. Persiano. The power of verification for one-parameter agents. In *Proc. of the 31st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 3142 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2004.

[AT1] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2001.

[AT2] A. Archer and É. Tardos. Frugal path mechanisms. *ACM Transactions on Algorithms*, 3(1), 2007.

[CK] C. Chekuri and S. Khanna. A ptas for minimizing weighted completion time on uniformly related machines. In *ICALP*, pages 848–861, 2001.

[CKV] G. Christodoulou, E. Koutsoupias, and A. Vidali. A lower bound for scheduling mechanisms. In *Proc. of Annual ACM Symposium on Discrete Algorithms (SODA)*, pages 1163–1170, 2007.

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

[ESS] E. Elkind, A. Sahai, and K. Steiglitz. Frugality in path auctions. In *Proc. of the 15th ACM-SIAM symposium on Discrete algorithms Annual ACM Symposium on Discrete Algorithms (SODA)*, pages 701–709, 2004.

[FPSS] J. Feigenbaum, C. H. Papadimitriou, R. Sami, and S. Shenker. A bgp-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1):61–72, 2005.

[Gam] I. Gamzu. Improved lower bounds for non-utilitarian truthfulness. In *Proceedings of the 5th Workshop on Approximation and Online Algorithms (WAOA)*, Lecture Notes in Computer Science, pages 15–26, 2007.

[GH] A. V. Goldberg and J. D. Hartline. Collusion-resistant mechanisms for single-parameter agents. In *Proc. of the 16th Annual ACM Symposium on Discrete Algorithms (SODA)*, pages 620–629. SIAM, 2005.

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

[HS] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, 1976.

[KKT] A. R. Karlin, D. Kempe, and T. Tamir. Beyond vcg: Frugality of truthful mechanisms. In *FOCS*, pages 615–626, 2005.

[KV] E. Koutsoupias and A. Vidali. A lower bound of $1+\phi$ for truthful scheduling mechanisms. In *Proceedings of the 32nd Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science, pages 454–464, 2007.

[LS] R. Lavi and C. Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pages 252–261, 2007.

[MS] A. Mu'alem and M. Schapira. Setting lower bounds on truthfulness. In *Proc. of Annual ACM Symposium on Discrete Algorithms (SODA)*, pages 1143–1152, 2007.

[NR] N. Nisan and A. Ronen. Algorithmic Mechanism Design. *Games and Economic Behavior*, 35:166–196, 2001. Extended abstract in the Proc. of the 31st Annual ACM Symposium on Theory of Computing (STOC), pages 129–140, 1999.

[PV] P. Penna and C. Ventre. Collusion-resistant mechanisms with verification yielding optimal solutions. In *Proc. of ESA*, pages 708–719, 2008.

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

[Ven] C. Ventre. Mechanisms with verification for any finite domain. In *Proc. of the 2nd Workshop on Internet and Network Economics (WINE)*, volume 4286 of *Lecture Notes in Computer Science*, pages 37–49. Springer, 2006.

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

A Postponed proofs

A.1 Proof of Fact 6

In this section, we prove Fact 6. To this end, we extend the notation $(\alpha, \mathbf{b}_{-i})$ to subsets C of agents in the natural way. Hence $(\mathbf{u}_C, \mathbf{b}_{-C})$ denotes the vector whose entry i is u_i for $i \in C$ and b_i otherwise.

PROOF. Let (A, p) be an arbitrary mechanism (with verification). By Definition 5, this mechanism is c-truthful if and only if the following condition holds:

(c-truthfulness) For every C of size at most c, for every $\mathbf{t}$, and for every $\mathbf{b}$, it holds that

$$\sum_{i \in C} \text{utility}_i((\mathbf{t}_C, \mathbf{b}_{-C})) \quad \geq \quad \sum_{i \in C} \text{utility}_i(\mathbf{b}). \tag{6}$$

The condition in Fact 6 can be rewritten as follows:

(condition in Fact 6) For every C of size at most c, for every $\mathbf{u}$, and for every $\mathbf{v}$ such that $\mathbf{u}$ and $\mathbf{v}$ agree in all entries not in C, that is, $\mathbf{v} = (\mathbf{v}_C, \mathbf{u}_{-C})$, it holds that

$$\sum_{i \in C} \text{utility}_i(\mathbf{u}) \quad \geq \quad \sum_{i \in C} \text{utility}_i(\mathbf{v}). \tag{7}$$

We have to show that the two conditions above are equivalent.

(c-truthfulness)⇒(condition in Fact 6). We define vectors $\mathbf{u}$ and $\mathbf{v}$ by means of $\mathbf{t}$ and $\mathbf{b}$ as in (6). Consider the following vectors: $\mathbf{u} := (\mathbf{t}_C, \mathbf{b}_{-C})$ and $\mathbf{v} := (\mathbf{b}_C, \mathbf{b}_{-C})$. Thus (7) follows from (6).

(c-truthfulness)⇐(condition in Fact 6). We define type vectors $\mathbf{t}$ and $\mathbf{b}$ as function of $\mathbf{u}$ and $\mathbf{v}$ as in (7). Take $\mathbf{t} := \mathbf{u}$ and $\mathbf{b} := (\mathbf{v}_C, \mathbf{u}_{-C})$. Thus (6) follows from (7).

This completes the proof. □

A.2 Payments are bounded by two constants

From the boundness condition of cost() and because the domain is bounded, we show that there exist two constants κ_{inf} and $\kappa_{\text{sup}} \geq \kappa_{\text{inf}}$ such that

$$\hbar - \kappa_{\text{sup}} \leq p_i(\mathbf{b}) \leq \hbar - \kappa_{\text{inf}}$$

for every bid vector $\mathbf{b}$. Let $\alpha_{\min} := \min_i\{\alpha_i\}$ and $\alpha_{\max} := \max_i\{\alpha_i\}$. We define

$$\kappa_{\text{inf}} := \frac{t_{\text{inf}} \cdot \left(\sum_{j \neq i} \alpha_j\right) + \text{inf cost}}{\alpha_{\max}}, \quad \text{and}$$

$$\kappa_{\text{sup}} := \frac{t_{\text{sup}} \cdot \left(\sum_{j \neq i} \alpha_j\right) + \text{sup cost}}{\alpha_{\min}}.$$

The inequality above follows from the definition of the payments (see Equation 4).

A.3 How to guarantee approximations

Suppose there exists a constant $c \geq 1$ such that following inequalities hold for all x and $\mathbf{t}$:

$$\text{cost}(x, \mathbf{t}) \leq \text{cost}^+(x, \mathbf{t}) \leq c \cdot \text{cost}(x, \mathbf{t}). \tag{8}$$

Let x' and x^* be the optimal solutions for $\text{cost}^+()$ and $\text{cost}()$, respectively. We show that x' is a c-approximation for $\text{cost}()$, that is

$$\text{cost}(x', \mathbf{t}) \leq c \cdot \text{cost}(x^*, \mathbf{t}).$$

By contradiction, assume $\text{cost}(x', \mathbf{t}) > c \cdot \text{cost}(x^*, \mathbf{t})$. Since x' is optimal for $\text{cost}^+()$, we have that

$$\text{cost}^+(x', \mathbf{t}) \leq \text{cost}^+(x^*, \mathbf{t}).$$

Because of (8) for x^* and $\mathbf{t}$ we have

$$c \cdot \text{cost}(x^*, \mathbf{t}) \geq \text{cost}^+(x^*, \mathbf{t}).$$

Then we reach a contradiction with (8) for x' and $\mathbf{t}$:

$$\text{cost}(x', \mathbf{t}) > c \cdot \text{cost}(x^*, \mathbf{t}) \geq \text{cost}^+(x^*, \mathbf{t}) \geq \text{cost}^+(x', \mathbf{t}).$$

This argument shows that assuming (8) the weakly utilitarian algorithm

$$A(\mathbf{t}) \in \arg\min_{x \in \mathcal{S}} \text{cost}^+(x, \mathbf{t})$$

returns a c-approximation for $\text{cost}()$. Suppose further that there exists a subset $\mathcal{S}'$ of feasible solutions with the following property: For any $\mathbf{t}$, the set $\mathcal{S}'$ contains a c'-approximate solution, that is, a solution $x' \in \mathcal{S}'$ such that

$$\text{cost}(x, \mathbf{t}) \leq c' \cdot \min_{x \in \mathcal{S}} \text{cost}(x, \mathbf{t}).$$

Then the weakly utilitarian algorithm

$$A(\mathbf{t}) \in \arg\min_{x \in \mathcal{S}'} \text{cost}^+(x, \mathbf{t})$$

returns a cc'-approximation for $\text{cost}()$.

Min-max approximations. In a min-max problem the goal is to minimize the cost function

$$\text{cost}(x, \mathbf{t}) = \max_i t_i(x).$$

We can guarantee $(1+\varepsilon)$-approximations, for any $\varepsilon > 0$, by considering the weakly utilitarian cost function (5):

$$\text{cost}^+(x, \mathbf{t}) = \max_i t_i(x) + \varepsilon \cdot \frac{\sum_i t_i(x)}{n}.$$

We simply have to show that (8) holds with $c = 1 + \varepsilon$, that is

$$\underbrace{\max_i t_i(x)}_{\text{cost}} \quad \leq \quad \underbrace{\max_i t_i(x) + \varepsilon \cdot \frac{\sum_i t_i(x)}{n}}_{\text{cost}^+} \leq (1+\varepsilon) \cdot \underbrace{\max_i t_i(x)}_{\text{cost}}. \tag{9}$$

These inequalities follows from $0 \leq \sum_i t_i(x) \leq n \cdot \max_i t_i(x)$.

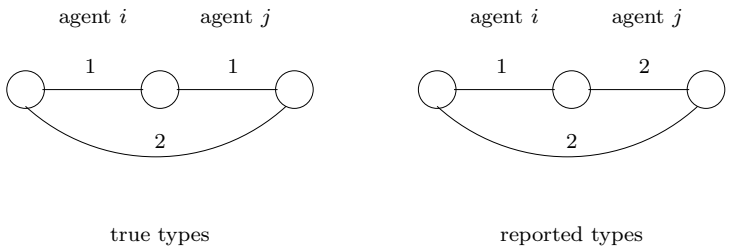

Figure 2: For Problem 1 the technique in [PV] does not provide collusion-resistant mechanisms.

B A simple utilitarian problem not addressed in [PV]

Consider the Problem 1 above that we called the path auction problem. Although the declaration domain of the problem is "binary" the technique in [PV] does not provide collusion-resistant mechanisms with verification for it. Indeed, no optimal algorithm for Problem 1 above satisfies the additional technical properties needed by the technique in [PV]. Their payments are related to the *threshold values*[5]: they pay an agent that is not selected a suitable constant diminished by this threshold value.

Consider the graph in Figure 2(left) where numbers represent the true costs associated to each edge (i.e., the true type of the corresponding agent). The algorithm has to choose the shortest path connecting the leftmost node to the rightmost node. Assume that for the true type vector either agent i and agent j are not selected. According to the payments in [PV] they both receive a payment of $\hbar - 1$ for a suitable constant $\hbar$. (Notice that their threshold values are equal to 1 since the upper path is selected as long as its length is less than 2.) Agent i can get a better payment if this threshold value decreases. In particular, agent j can bid 2 as in Figure 2(right) and still being not selected. In this case she receives the same payment $\hbar - 1$. This however, causes the agent i to get a strictly better utility since now her threshold becomes 0 (she receives a payment $\hbar$ and she is not selected as in the previous case).

Observe that this example shows that the mechanism in [PV] is neither bribeproof or group strategyproof for this problem.

C Applications to various mechanism design problems

In this section, we describe in detail the mechanism design problems considered in the literature and to which we applied our results.

C.1 Scheduling unrelated machines

The problem of scheduling *unrelated machines* is one of the most studied problems in the algorithmic mechanism design literature. Here we are given m tasks that have to be scheduled on n machines, each corresponding to a selfish agent. The type of a machine specifies the time the machine takes

[5]The threshold value distinguishes "winning" from "losing" bids. By declaring a bid lower than the threshold value an agent is selected. On the contrary, by declaring values greater of the threshold value an agent is not selected. Any truthful mechanism with verification admits these threshold values [ADPP].

to complete all tasks that a schedule x assigned it (this time being the cost for the corresponding agent). The objective is to minimize the *makespan*, that is, the cost function $\max_i t_i(x)$. For this min-max problem Corollary 13 implies the following:

Corollary 16 *The scheduling unrelated machines problem admits a collusion-resistant $(1+\varepsilon)$-approximate mechanism with verification, for every $\varepsilon > 0$, if machines execution times are bounded from above by some constant. The mechanism satisfies the voluntary participation condition.*

Nisan and Ronen [NR] introduced this problem and proved that no deterministic mechanism without verification can achieve approximations better than 2 for two or more machines. This lower bound have been recently improved to $1+\sqrt{2}$ in [CKV] for 3 or more machines and to $1+\Phi \approx 2.618$ as n goes to infinity [KV]. A deterministic n-approximation mechanism is given in [NR] and a randomized $(7n/8)$-approximation one is given in [MS].

C.2 Scheduling related machines

Archer and Tardos [AT1] considered the case of *related* machines in which machines' execution times differ only by a multiplicative factor (the speed of the machine). One of the problem versions they considered is that of minimizing the makespan. For this problem [AAS] provides a $(1+\varepsilon)$-approximate polynomial-time (also in $1/\varepsilon$) truthful mechanism without verification for any fixed number of machines. We have the following result.

Theorem 17 *The scheduling related machines problem (so as to minimize the makespan) admits a collusion-resistant FPTAS mechanism with verification for any fixed number of machines, if machines execution times are bounded from above by some constant. The mechanism satisfies the voluntary participation condition.*

Proof. Fix any $\varepsilon > 0$ and consider $0 < \varepsilon', \varepsilon'' < \varepsilon$ such that $(1+\varepsilon') \cdot (1+\varepsilon'') = (1+\varepsilon)$. By using the algorithm of Figure 3 in [AAS] we compute a suitable subset $\mathcal{S}'$ of feasible allocations. This is done on top of an FPTAS (with measure guarantee w.r.t. ε') for the makespan (e.g., the one in [HS]). The set $\mathcal{S}'$ is computed in time polynomial in the input size and in $1/\varepsilon'$ and satisfies the following: the best solution out of $\mathcal{S}'$ has a makespan which is a $(1+\varepsilon')$-approximation of the optimum in $\mathcal{S}$ (see Theorems 5.1 and 5.2 in [AAS]). Therefore, since $\mathcal{S}'$ has size polynomial in the input size and in $1/\varepsilon'$ (and thus size polynomial in $1/\varepsilon$), there exists an FPTAS weakly utilitarian algorithm which finds an optimal solution for the weakly utilitarian cost function

$$\mathrm{cost}^+(x, \mathbf{t}) = \max_i t_i(x) + \varepsilon'' \cdot \frac{\sum_i t_i(x)}{n}.$$

(This is the weakly utilitarian cost function (5) with ε'' in place of ε.) As observed above this is a $(1+\varepsilon'')$-approximation of the solution with minimal makespan in $\mathcal{S}'$ (see (9) above). This solution is thus a $(1+\varepsilon') \cdot (1+\varepsilon'')$-approximation of the optimum in $\mathcal{S}$. Corollary 12 implies then the existence of an FPTAS collusion-resistant mechanism with verification. □

C.3 Weighted sum of jobs completion times

Archer and Tardos [AT1] for the case of *related* machines also considered the problem of minimizing the weighted sum of all jobs' completion times. That is, for given $w_j \in \mathbb{R}^+$, the objective is to

minimize the cost function

$$\sum_j w_j \cdot C_j(x, \mathbf{t}) \tag{10}$$

where $C_j(x, \mathbf{t})$ is the completion time of job j according to the schedule x and the machines' types $\mathbf{t}$. They provided that no truthful mechanism without verification can have an approximation better than $2/\sqrt{3}$ for two or more related machines [AT1]. In contrast, we can obtain $(1+\varepsilon)$-approximation mechanisms using the same idea of min-max problems.

Theorem 18 *The problem of minimizing the weighted sum of all jobs completion times admits, for every $\varepsilon > 0$, a collusion-resistant $(1+\varepsilon)$-approximation with verification, if machines execution times are bounded from above by some constant. The mechanism satisfies the voluntary participation condition.*

PROOF. Consider the following weakly utilitarian cost function:

$$\sum_j w_j \cdot C_j(x, \mathbf{t}) + \varepsilon w_{\min} \sum_i t_i(x), \tag{11}$$

with $w_{\min} = \min_j w_j$. Observe that, since $C_j(x, \mathbf{t})$ is the completion time of job j, and $t_i(x)$ is the completion time of machine i, we have $t_i(x) = \sum_{j \in x^i} C_j(x, \mathbf{t})$ with x^i being the set of jobs that x assigns to machine i. Therefore

$$w_{\min} \sum_i t_i(x) \leq \sum_j w_j C_j(x, \mathbf{t})$$

thus implying the two inequalities required to prove the $(1+\varepsilon)$-approximation

$$\underbrace{\sum_j w_j C_j(x, \mathbf{t})}_{\text{cost}} \leq \underbrace{\sum_j w_j C_j(x, \mathbf{t}) + \varepsilon w_{\min} \sum_i t_i(x)}_{\text{cost}^+} \leq (1+\varepsilon) \cdot \underbrace{\sum_j w_j C_j(x, \mathbf{t})}_{\text{cost}}. \tag{12}$$

The theorem thus follows from Theorem 9. □

By using the weakly utilitarian cost function defined above for the weighted sum scheduling we obtain a PTAS for any constant number of machines as described in the main part of the paper. This is shown in the next theorem.

Theorem 19 *The weighted sum of all jobs completion times admits a collusion-resistant PTAS mechanism with verification for any fixed number of machines, if machines execution times are bounded from above by some constant. The mechanism satisfies the voluntary participation condition.*

PROOF. Fix any $\varepsilon > 0$ and consider $0 < \varepsilon', \varepsilon'' < \varepsilon$ such that $(1+\varepsilon') \cdot (1+\varepsilon'') = (1+\varepsilon)$. We call the algorithm of Figure 3 in [AAS] AAS. We define the step $2(a)$ of the algorithm AAS as the PTAS (with measure guarantee w.r.t. ε') for the weighted sum scheduling in [CK]. Then by means of the algorithm AAS we compute in polynomial-time a suitable subset $\mathcal{S}'$ of feasible allocations. The set $\mathcal{S}'$ is such that the best solution in it has a weighted completion time which is a $(1+\varepsilon')$-approximation of the optimum in $\mathcal{S}$. Indeed, by defining the step 3 of AAS with respect

to the measure (10), proofs of Theorems 5.1 and 5.2 in [AAS] hold *mutatis mutandis* in this case. Therefore, since $\mathcal{S}'$ has polynomial-size, there exists a PTAS weakly utilitarian algorithm which finds an optimal solution for the weakly utilitarian cost function

$$\sum_j w_j \cdot C_j(x, \mathbf{t}) + \varepsilon'' w_{\min} \sum_i t_i(x), \quad \text{with} \quad w_{\min} = \min_j w_j.$$

(This is the weakly utilitarian cost function (11) above with ε'' in place of ε.) As observed above this is a $(1+\varepsilon'')$-approximation of the solution with minimal makespan in $\mathcal{S}'$ (see (12) above). This solution is thus a $(1+\varepsilon') \cdot (1+\varepsilon'')$-approximation of the optimum in $\mathcal{S}$. Corollary 12 implies the existence of a PTAS collusion-resistant mechanism with verification. □

C.4 Inter-domain routing

We are given an undirected graph $G = (N, L)$ where each node represents an Autonomous System (AS) of which the Internet is comprised. The set of nodes N consists of a destination node d and n source nodes. The set of edges L is the set of links between nodes in N. The intensity of the traffic (number of packets) originating from source i for destination d is denoted by τ_i. Let $neighbours(i)$ be the set of nodes in N linked to i. Each source node is a selfish agent having a *private* cost function that specifies the per-packet cost $cost\_per\_packet_i(j)$ incurred by this node for carrying traffic to node $j \in neighbours(i)$. This cost models the load imposed to the AS internal routing for sending a packet to an adjacent AS.

The goal is to assign all source nodes routes to d and, since nodes are not allowed to transfer traffic to two adjacent nodes, the route allocation should form a confluent tree to the destination d. The set of solutions $\mathcal{S}$ is then the set of all trees of G rooted at d. Each tree x in $\mathcal{S}$ represents the fact that a node i routes traffic to its parent $parent_i(x)$. Therefore, in addition to its own traffic, node i must route the traffic generated at all nodes below it in the tree x, for a total of

$$traffic_i(x) = \tau_i + \sum_{k \text{ is below } i \text{ in } x} \tau_k$$

packets to be forwarded to node $parent_i(x)$. Hence the cost for agent i given a solution x is equal to

$$t_i(x) = cost\_per\_packet_i(parent_i(x)) \cdot traffic_i(x).$$

Note that this problem is *not* one-parameter since the agent has to declare a per-packet cost for each of her neighbors.[6] The per-packet cost is determined by a number of parameters "internal" to the AS and it is essentially the latency experienced by a packet. This latency can be observed and thus verification is possible in this scenario.

Lowest total cost (utilitarian inter-domain routing). The *Lowest Total Cost* problem consists in finding the tree minimizing the total cost of routing all packets [FPSS], that is, the cost function

$$\sum_{i \in N} t_i(x).$$

[6] Observe that in our setting nodes should declare a cumulative cost $t_i()$. However, in this case it suffices that agent i declare the cost function $cost\_per\_packet_i()$ as it defines the cumulative cost we need.

This can be done efficiently by minimizing, for every $i \in N$, the cost of the path between i and d. This algorithm is a polynomial-time (weakly) utilitarian algorithm and thus Corollary 12 implies the following:

Corollary 20 *The lowest total cost problem in inter-domain routing admits an exact* polynomial-time *collusion-resistant mechanism with verification, whenever nodes costs are bounded from above by some constant. The mechanism satisfies the voluntary participation condition.*

Workload minimization (min-max inter-domain routing) In a different version we seek routing trees in which the workload imposed on the busiest node is minimized, that is, the cost function

$$\max_{i \in N} t_i(x)$$

is minimized. We refer to this problem as *Workload Minimization* in inter-domain routing. This problem has been defined in [MS] where the authors showed a lower bound of about 1.618 for truthful mechanisms without verification, later improved to 2 [Gam]. Since this is a min-max problem from Corollary 13 we immediately obtain the following:

Corollary 21 *The workload minimization problem in inter-domain routing admits a collusion-resistant* $(1+\varepsilon)$*-approximation mechanism with verification, for every* $\varepsilon > 0$*, if nodes costs are bounded from above by some constant. The mechanism satisfies the voluntary participation condition.*

D The Nisan-Ronen model of verification

In this section we compare two different models of mechanisms with verification: the original model defined in [NR] that we call the *full monitoring* model and the model we consider here (studied in [ADPP, ADP$^+$, Ven]) that we call the *non-limited* model.

Mechanisms with verification have been introduced in [NR] in the context of scheduling unrelated machines. If a task allocation x is chosen, then an agent i can execute her task(s) in any amount of time $e_i(x)$ which cannot be smaller than true amount of time $t_i(x)$ required by her true type. The main limitation of their model is that it assumes the cost for an agent being always her execution time

$$e_i(x)$$

and not the actual time $t_i(x)$ she spent on her task(s). In other words, it is like a worker, who did her job in one hour, feels like being working the whole day because this is what she said to her boss (or equivalently that a boss fully monitors the her worker takes the declared time to do the job). In the model considered here, instead, the worker feels like being working for one hour as the rest of the time she did nothing or she used her time for her own.

Another difference is that in the fully monitoring model of [NR] the mechanism makes no ties between the declaration b_i (declarations are just used to compute the output) and the execution time, only observes $e_i()$ (that as noticed can only be greater than the actual execution time) and awards payments that depend on both b_i and $e_i()$. This implies that in principle in the fully monitoring model an agent being caught is paid by the mechanism. While in the non-limited model we impose a payment of 0 for an agent caught lying by the mechanism. Below, we show that

this more general aspect of the fully monitoring model does not give extra power with respect to the non-limited one. It is also crucial for constructions to work in the model of [NR] that the mechanism observes the execution time of every single job (see below their definition of *Compensation-and-Bonus* mechanisms). This assumption is unneeded in the the non-limited model.

Note that the non-limited model is more general than the one in [NR]. Indeed, in the next section we show that if a mechanism with verification works in our model, then it also works in the model in [NR], while the converse is not true in general.

D.1 Comparing the two models

To see that every truthful mechanism in our non-limited model is also truthful in the fully monitoring model, observe that truthfulness in their model means that

$$p_i((t_i, \mathbf{b}_{-i}), \mathbf{e}) - t_i(x) \geq p_i(\mathbf{b}, \mathbf{e}) - \max\{t_i(x'), e_i(x')\}$$

where $p_i((t_i, \mathbf{b}_{-i}), \mathbf{e})$ and $p_i(\mathbf{b}, \mathbf{e})$ is the money received by i when truthtelling and when reporting arbitrary type b_i, respectively; solution x is computed on the true type, while solution x' is computed when reporting b_i. Indeed, while the agent is being caught lying when reporting b_i, that is $t_i(x') > e_i(x')$, then her cost will be the true execution time $t_i(x')$ (results cannot be given by i before than actual execution time). On the contrary when her lie is not detected by the mechanism, in the fully monitoring model, the cost suffered by the agent is the declared execution time $e_i(x')$.

On the other hand truthfulness in our model means that

$$p_i(\mathbf{t}) - t_i(x) \geq \begin{cases} p_i(\mathbf{b}) - t_i(x') & \text{if } i \text{ is not caught} \\ 0 - t_i(x') & \text{otherwise} \end{cases} \tag{13}$$

where $\mathbf{t}$ is a shorthand for $(t_i, \mathbf{b}_{-i})$ and where, as above, solution x is computed on the true type, while solution x' is computed when reporting b_i.

Therefore, given a truthful mechanism in the non-limited model we can set

$$p_i(\mathbf{b}, \mathbf{e}) := \begin{cases} p_i(\mathbf{b}) & \text{if } i \text{ is not caught lying} \\ 0 & \text{otherwise} \end{cases}$$

and thus, if i is not caught lying, given that the execution time must satisfy $e_i(x') \geq t_i(x')$, (13) implies

$$p_i((t_i, \mathbf{b}_{-i}), \mathbf{e}) - t_i(x) \geq p_i(\mathbf{b}, \mathbf{e}) - e_i(x')$$

that is truthfulness in the fully monitoring model when i is not caught lying. If however i is caught lying then (13) implies

$$p_i((t_i, \mathbf{b}_{-i}), \mathbf{e}) - t_i(x) \geq p_i(\mathbf{b}, \mathbf{e}) - t_i(x')$$

that is truthfulness in the fully monitoring model when i is caught lying. The same argument applies also to collusion-resistant mechanisms: instead of a single agent, we consider an arbitrary coalition. This shows the following.

Fact 22 *Every truthful (resp. collusion-resistant) mechanism with verification in the non-limited model is truthful (resp. collusion-resistant) in the model of [NR].*

In the sequel, we show that *Compensation-and-Bonus* mechanisms in [NR] are *not* truthful in the non-limited model.

To introduce these mechanisms, we use the following notation. Every solution x is a task allocation which assigns a subset x_i of tasks to machine i. Given a vector $\mathbf{b} = (b_1, \ldots, b_n)$, we sometimes write b_i^j to denote the declaration for job j on machine i according to b_i. The so called Compensation-and-Bonus mechanisms outputs an *optimal* schedule and provide agent i a payment

$$p_i(\mathbf{b}, \mathbf{e}) := comp_i(\mathbf{b}, \mathbf{e}) + bonus_i(\mathbf{b}, \mathbf{e})$$

with $comp_i()$ being the following compensation function

$$comp_i(\mathbf{b}, \mathbf{e}) := \sum_{j \in x_i, x = A(\mathbf{b})} e_i^j = e_i(A(\mathbf{b})),$$

and $bonus_i()$ being a suitable bonus function. The key property (which does not hold in the non-limited model) is that, being the agent's cost equal to $e_i(A(\mathbf{b}))$, the utility of an agent in the model of [NR] is always equal to her bonus:

$$p_i(\mathbf{b}, \mathbf{e}) - e_i(A(\mathbf{b})) = bonus_i(\mathbf{b}, \mathbf{e}).$$

To show that this mechanism is *not* truthful in the non-limited model, we first give the necessary details about the bonus function. The *corrected time vector* is obtained by correcting the reported execution times in b_i^j into the execution times e_i^j. Note that this is possible only for those i and j such that the computed solution x assigns j to machine i. Formally

$$corr_i^j(x, \mathbf{b}, \mathbf{e}) := \begin{cases} e_i^j & \text{if } j \in x_i \\ b_i^j & \text{if } j \in x_l \text{ and } l \neq i \end{cases}$$

and the bonus function is as follows:

$$bonus_i(\mathbf{b}, \mathbf{e}) := -\text{cost}(A(\mathbf{b}), \mathbf{corr}(A(\mathbf{b}), \mathbf{b}, \mathbf{e}))$$

where **corr** denotes the corrected time vector. Now observe that the corrected time vector **corr** is identical to the bid vector $\mathbf{b}$ whenever $\mathbf{e} = \mathbf{b}$, that is, each agent i executes a task j in time $e_i^j = b_i^j$. In this case, since $\mathbf{corr} = \mathbf{b}$, the bonus function is equal to

$$bonus_i(\mathbf{b}, \mathbf{e}) = -\text{cost}(A(\mathbf{b}), \mathbf{b}). \tag{14}$$

To see that the mechanism is not truthful, we consider the instance with two tasks and two machines whose true types are as follows. The first task takes time 1 on both machines, while the second task takes time 2 both machines. Suppose without loss of generality, that the optimal algorithm assigns the first task to the first machine, and the second task to the second machine. That is, we are in the following case

$$\mathbf{t} = \begin{pmatrix} \boxed{1} & 2 \\ 1 & \boxed{2} \end{pmatrix}$$

with framed boxes representing the allocation of the optimal algorithm. Since in the truthtelling scenario the execution times are equal to the true types then by (14) the utility in the non-limited model of agent 1 is

$$p_1(\mathbf{t}, \mathbf{t}) - t_i(A(\mathbf{t})) = bonus_1(\mathbf{t}, \mathbf{t}) = -\text{cost}(A(\mathbf{t}), \mathbf{t}) = -2.$$

Now observe that agent 1 can get a better utility by declaring $b_1 = (2, 3)$. The optimal algorithm assigns task as before

$$\mathbf{b} = \begin{pmatrix} \boxed{2} & 3 \\ 1 & \boxed{2} \end{pmatrix}.$$

Moreover, since in the non-limited model we have that

$$b_i(A(\mathbf{b})) = e_i(A(\mathbf{b}))$$

and as only one job is assigned to the first machine we have that agent 1 executes her assigned task in time $e_1^1 = b_1^1 = 2$. This makes the corrected time vector identical to the bid vector $\mathbf{b}$ (the other agent is truthtelling and executes her task accordingly). Then by (14) the utility of agent 1 in the non-limited model becomes

$$\begin{aligned} p_1(\mathbf{b}, \mathbf{e}) - t_1(A(\mathbf{b})) &= comp_1(\mathbf{b}, \mathbf{e}) + bonus_1(\mathbf{b}, \mathbf{e}) - t_1(A(\mathbf{b})) \\ &= e_1(A(\mathbf{b})) - \text{cost}(A(\mathbf{b}), \mathbf{b}) - t_1(A(\mathbf{b})) \\ &= b_1^1 - \max\{b_1^1, b_2^2\} - t_1^1 \\ &= 2 - 2 - 1 = -1. \end{aligned}$$

This shows the following.

Fact 23 *The Compensation-and-Bonus mechanism [NR] is not truthful in the non-limited model.*