Collusion-resistant mechanisms with verification yielding optimal solutions[*]

Paolo Penna[†] Carmine Ventre[‡]

October 21, 2009

Abstract

A *truthful mechanism* consists of an algorithm augmented with a suitable payment function which guarantees that the "players" cannot improve their utilities by "cheating". Mechanism design approaches are particularly appealing for designing "protocols" that cannot be manipulated by rational players.

We present new constructions of so called mechanisms with *verification* introduced by Nisan and Ronen [NR01]. We first show how to obtain mechanisms that, for single-parameter domains, are resistant to *coalitions* of colluding agents even if they can exchange compensations. Based on this result we derive a class of *exact* truthful mechanisms with verification for *arbitrary* bounded domains. This class of problems includes most of the problems studied in the algorithmic mechanism design literature and for which exact solutions cannot be obtained with truthful mechanisms *without* verification. This result improves over all known previous constructions of exact mechanisms with verification.

1 Introduction

The emergence of the Internet as *the* platform for distributed computing posed new questions on how to design "good" protocols which take into account the lack of a "central authority" [KP99, NR01, Pap01]. Algorithmic mechanism design [NR01] considers distributed settings where the participants, termed agents, cannot be assumed to follow the protocol but rather their own preferences. The designer must ensure in advance that it is in the agents' interest to behave correctly. The protocol can be regarded to as an algorithm augmented with a suitable payment rule. The algorihm needs to collect (part of) the input from the agents and the desired condition is that agents have no interest in reporting false information. Mechanism design is in fact a beautiful area in Game Theory which studies precisely the boundary conditions under which it is possible (for the designer) to achieve a certain goal. The work by Nisan and Ronen [NR01] points out (at least) two important differences that stem from Computer Science applications: (1) the designer's goal is typically different from the "classical" micro-economic setting and (2) new mechanisms become

[*]An extended abstract of this work appeared in Proc. of ESA'08.

[†]Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, Italy. E-mail: penna@dia.unisa.it. The author is supported by the European Union under IST FET Integrated Project AEOLUS (IST-015964).

[‡]Computer Science Department, University of Liverpool, UK. E-mail: Carmine.Ventre@liverpool.ac.uk. The author is partially supported by the European Union under IST FET Integrated Project AEOLUS (IST-015964).

possible because of the possibility of gaining additional information. These mechanisms, called mechanisms with *verification*, are essentially based on the idea that it is possible to partially verify the agents' reported information. We explain these issues by means of two examples from [NR01].

Perhaps the simplest mechanism design problem motivated by Computer Science applications is the following one:

Example 1 (The links problem) *We want to transmit one message from a source node to a destination node and there are two available links. Each link has a transmission rate, say* t_1 *and* t_2 *for link* 1 *and link* 2*, respectively. This information is only known to the owner of the link who may find it convenient to misreport this information for the following reason: If her link is chosen for transmitting the message, then she will incur a cost specified by the corresponding* t_i *(the amount of time her link is busy). Without compensations (payments) there is a clear incentive for both agents to exaggerate their costs so that their link is not selected. Can we find payments such that,* ***no matter what are the true rates*** t_1 ***and*** t_2*, each agent finds it convenient to report her transmission rate truthfully?*

We can regard to the above question in terms of mechanism design. In fact, we are asking if there is a *truthful* mechanism for the problem above, meaning that both agents (the owners of the links) always maximize their utility (payment minus cost) by reporting their true information (the rate of their own link). In general, the true information of each agent is referred to as the *type* of this agent. The main technique in the field is the so-called Vickrey-Clarke-Groves (VCG) mechanisms [Vic61, Cla71, Gro73] which deal with the case in which the objective is the (weighted) social welfare. That is, truthfulness is guaranteed whenever the underlying algorithm minimizes a global cost functions of the form

$$\alpha_1 \cdot t^1(x) + \cdots + \alpha_n \cdot t^n(x) \tag{1}$$

where each α_i is some nonnegative constant, and $t^i(x)$ is the cost incurred by agent i when solution x is chosen. One one hand, this is very general result because the costs t^i of the agents can be arbitrary functions. On the other hand, it limits to objectives of the form (1). To see why this is a limitation, consider the following generalization of the problem in Example 1:

Example 2 (Scheduling unrelated machines) *We have two machines of type* t^1 *and* t^2 *and a number of jobs to be scheduled. Every allocation of the jobs,* x*, results in a completion time* $t^1(x)$ *and* $t^2(x)$ *given by the sum of the execution time of the jobs that are allocated to each machine: Each job* j *requires* t^1_j *on the first machine and* t^2_j *on the second machine (machines are unrelated and the execution time of a job on a machine can be an arbitrary positive number). The owners of each machine incurs a cost equal to the completion time of her machine, and the type of the machines are only known to the owner. Can we find a truthful mechanism (algorithm + payments) which minimizes the makespan, that is, the global cost function* $\max\{t^1(x), \ldots, t^n(x)\}$?

In general, the construction of a truthful mechanism is a challenging problem since the mechanism must fix the "rules" in advance without knowing the types of the agents. The only available information is that each agent's type belongs to some *domain* which depends on the problem and agents can only report types in that domain. Moreover, one may want to guarantee a stronger condition which is the fact that even coalitions of *colluding* agents cannot manipulate the mechanism:

Example 3 (Collusion in a truthful mechanism) *The Vickrey auction [Vic61], a special case of VCG mechanisms, provides a very elegant solution for the links problem in Example 1: We select the link having the best rate and we pay to its owner an amount equal to the rate of the other link. For instance, if the true/reported rates are $t_1 = 5$ and $t_2 = 2$, then we select link 2 and pay its owner an amount equal 5. Note, for instance, that the owner of the selected link has no incentive to declare a cost equal 3, because in this case her link is still selected and she still receives the same payment. Indeed, it is known that this mechanism is truthful (see e.g. [NR01]). However, such mechanism can be easily manipulated if the two agents collude: the owner of the second link can offer money to the owner of the other link for misreporting her true cost. When they report costs 100 and 2, the second link (the briber) receives 100 as payment and the non-selected one (the bribee) is also better off because receives money from the other.*

Schummer [Sch00] and Goldberg and Hartline [GH05] consider the much stronger solution concept of *c-truthful* mechanism which requires that truthtelling remains a dominant strategy (utility maximizing) for arbitrary coalitions of at most c colluding agents, even when the colluding agents can exchange compensations among themselves.[1]

In this work we present new constructions of mechanisms with verification which guarantee c-truthful mechanisms for certain domains (including Example 1) or exact solutions for a very general class of problems (including Example 2). Before discussing these and prior results in detail, we describe informally the main idea of mechanisms with verification:

Mechanisms with verification. In both Example 1 and Example 2, the cost of an agent corresponds to the amount of time this agent will have to work for "the system" according to the computed solution (the link selected, or the tasks' allocation). Here is it possible to use mechanisms *with verification* meaning that certain lies can be detected by the mechanism. For the scheduling problem, machine i cannot provide the output of its tasks before $t^i(x)$ time steps. Therefore, if agent i reports a type b^i and a solution x is implemented, the mechanism is able to detect that b^i is *not* the true type of machine i if $b^i(x) < t^i(x)$. A similar argument can be applied to the links problem when the selected link reported a better (smaller) transmission rate.

Mechanisms with verification can be applied to the following general framework (see Section 1.2 for a formal definition). For every feasible solution x, an agent of type t^i has a cost $t^i(x)$ associated to this solution. The simplest way to view this cost is to consider it as the time that this agent must spend for implementing solution x. (In general, the verification paradigm can be applied to all "measurable" types such as amounts of traffic, routes availability [LSZ08] and distance.) Artificial delays can be introduced at no cost since an agent can use the idle time for other purposes.[2] Agent i is caught lying if her reported type b^i and the computed solution x are such that $b^i(x) < t^i(x)$. Agents who are caught lying receive no payment. In contrast, the classical approach in mechanism design is to provide always each agent with a payment that depends only on the reported types.

[1]In fact, Schummer [Sch00] considers bribeproof mechanisms which correspond to the case of two colluding agents, one briber and one bribee as in Example 3. He proves impossibility results which clearly apply to any c-truthful mechanism for $c \geq 2$, being the latter notion a stronger one.

[2]Nisan and Ronen [NR01] considered the case in which an agent introducing an artificial delay to her computations will pay *this* augmented cost. This is one of the differences between mechanisms with verification we consider and those in [NR01]. See [PV09, full version] for a discussion.

In order to distinguish these mechanisms from mechanisms with verification, in the sequel we use the term mechanisms *without verification*.

1.1 Our contribution and related work

We study the existence of truthful (or even c-truthful) mechanisms with verification that guarantee *exact* solutions for problems in which the objective is to minimize some global cost function of interest. Intuitively speaking, our basic question is whether one can augment an optimal algorithm with a suitable payment function in order to guarantee that no agent (or even coalitions of colluding agents) can benefit from misreporting their types (i.e., part of the input of the algorithm). We consider a rather general class of objective functions in which the global cost of a solution depends on the various costs that the agents associate to that solution; Naturally, the overall cost cannot decrease if the cost of one agent increases (see Section 2 for a formal definition). The contribution of this work is twofold:

- We provide a sufficient condition for which an algorithm can be turned into a c-*truthful* mechanism with verification, for any $c \geq 1$. This result applies to the class of *single-parameter* bounded domains (see Section 3 for a formal definition).

- We then show how to obtain optimal truthful mechanisms with verification for the much more general case of *arbitrary* bounded domains, i.e., the mechanism needs only an upper bound on the agents' costs (see Section 1.2). Despite the fact that these domains are extremely rich, we provide *exact* truthful mechanisms with verification for *every* problem in which the global cost function is of the form
$$\text{Cost}(t^1(x), \ldots, t^n(x)) \tag{2}$$
where $t^i(x)$ is the cost that agent i associates to solution x and the above function is naturally nondecreasing in its arguments.

The conditions for obtaining these mechanisms are stated in terms of *algorithmic properties* so that the design of the entire mechanism reduces to the design of an algorithm that fulfills these conditions. All our mechanisms satisfy also the *voluntary participation* condition saying that truthful agents have always a nonnegative utility.

The result on single-parameter bounded domains is the first technique for obtaining c-truthful mechanisms, for $c > 1$, without restricting to a particular class of global cost functions and it might be of some independent interest. For instance, certain non-utilitarian graph problems studied in [PW05] have single-parameter domains and thus are the right candidate for studying exact n-truthful mechanisms based on our constructions (namely Theorems 15 and 19). Interestingly enough, the only way to guarantee c-truthfulness without verification, for $c \geq 2$, is to run a (useless) mechanism which returns always the same *fixed* solution [Sch00, GH05].

The result on arbitrary bounded domains extends significantly the class of problems for which it is possible to have *exact* truthful mechanisms with verification. In particular, [Ven06] shows exact mechanisms for the case of *finite* domains (i.e., there is a finite set of possible types that each agent can report to the mechanism) but does not need to assume functions of the form (2). Exact n-truthful mechanisms with verification for so-called *weakly utilitarian* costs (see Section 2.1) are presented in [PV09]: weakly utilitarian costs do not cover all costs of the form (2) like, for instance, the cost function $\max_i t^i(x)$. These so called min-max problems received a lot of attention in the algorithmic mechanism design literature [NR01, CKV07, KV07, MS07, Gam07]. These works prove that there is no exact or even r-approximate mechanism without verification, for some $r > 1$; These results apply also to finite domains and to mechanisms without verification that run in exponential time and/or use randomization [MS07].

We instead show exact mechanisms with verification for any global cost function of the form (2) without assuming finite domains like in [Ven06] (see Corollary 27). Indeed, we only need to consider an (arbitrarily large) upper bound on the agents' costs, which turns out to be reasonable in practice. These arbitrary bounded domains are, in general, infinite because there are infinitely many types that an agent can report. Since the "cycle-monotonicity" approach adopted in all recent constructions [ADPP09, ADP$^+$06, Ven06] cannot deal with infinite domains, we use a totally different idea which is to turn c-truthful mechanisms for single-parameter domains into truthful mechanisms for arbitrary domains (see Section 4). The result of Corollary 27 is "tight" in the sense that one cannot relax any of the assumptions without introducing additional conditions (see Theorems 28 and 29). Finally, an explicit formula for the payments guarantees that the entire mechanism runs in polynomial-time if the algorithm is polynomial-time and the domain is finite (Corollary 30).

In this work we do not consider frugality issues, that is, how much the mechanism pays the agents. The optimality of the payments is an important issue *in general* since even truthful mechanisms must have large payments for rather simple problems [ESS04]. Our positive results pose another interesting question that is to design *computationally-efficient* algorithms satisfying the conditions required by our methods.

Roadmap. Preliminary definitions are given in Section 1.2. In Section 2 we introduce the class of optimal algorithms leading to (c-)truthful mechanisms. Mechanisms for single-parameter domains are given in Section 3, while those for arbitrary domains are presented in Section 4.

1.2 Preliminaries

We have a finite set $\mathcal{S}$ of feasible *solutions*. We let $\mathfrak{s} := |\mathcal{S}|$ denote the number of feasible solutions. There are n selfish agents, each of them having a so called *type*

$$t^i : \mathcal{S} \to \mathbb{R}^+ \cup \{0\}$$

which associates a monetary *cost* to every feasible solution. If an agent i receives a payment equal to r^i and a solution x is selected, then her *utility* is equal to

$$r^i - t^i(x). \tag{3}$$

Each type t^i belongs to a so called *domain* D^i which consists of all admissible types, that is, a subset of all functions $u : \mathcal{S} \to \mathbb{R}^+ \cup \{0\}$. The type t^i is *private knowledge*, that is, it is known to agent i only. Everything else, including each domain D^i, is *public knowledge*. Hence, each agent i can *misreport* her type to *any* other element b^i in the domain D^i. We sometimes call such b^i the *bid* or *reported type* of agent i. We let D being the cross product of all agents domains, that is, D contains all bid vectors $\mathbf{b} = (b^1, \ldots, b^n)$ with b^i in D^i. An *algorithm* A is a function

$$A : D \to \mathcal{S}$$

which maps all agents (reported) types $\mathbf{b}$ into a feasible solution $x = A(\mathbf{b})$.[3] A *mechanism* is a pair (A, p), where A is an algorithm and $p = (p^1, \ldots, p^n)$ is a vector of suitable *payment functions*, one for each agent, where each payment function

$$p^i : D \to \mathbb{R}$$

associates some amount of money to agent i. We say that D is a *bounded domain* if there exists ℓ such that $b^i(x)$ belongs to the interval $[0, \ell]$, for all solutions x, for all b^i in D^i, and for all agents i. Unless we make further assumptions on the domain D, we have (algorithms over) *arbitrary bounded domains*. Throughout the paper we consider only type vectors $\mathbf{t}$ in the domain D and we denote by t^i the type corresponding to agent i.

We say that an agent i is *truthtelling* if she reports her type, that is, the bid b^i coincides with her type t^i. Given an algorithm A and bids $\mathbf{b} = (b^1, \ldots, b^i, \ldots, b^n)$, we say that agent i is *caught lying by the verification* if the following inequality holds:

$$t^i(A(\mathbf{b})) > b^i(A(\mathbf{b})).$$

A mechanism (A, p) is a *mechanism with verification* if, on input bids $\mathbf{b}$, every agent that is caught lying does not receive any payment, while every other agent i receives her associated payment $p^i(\mathbf{b})$. Hence, the utility of an agent i whose type is t^i is equal to

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

On the contrary, we say that (A, p) is a *mechanism without verification* if every agent receives *always* her associated payment $p^i(\mathbf{b})$.

For any two type vectors $\mathbf{t}$ and $\mathbf{b}$, we say that a coalition C *can misreport* $\mathbf{t}$ *to* $\mathbf{b}$ if the vector $\mathbf{b}$ is obtained by changing the type of some of the agents in C, i.e., $t^i = b^i$ for every agent i *not* in the coalition C. For any two type vectors $\mathbf{t}$ and $\mathbf{b}$, we say that *verification does not catch* $\mathbf{t}$ *misreported to* $\mathbf{b}$ if $t^i(A(\mathbf{b})) \leq b^i(A(\mathbf{b}))$ for every agent i. Conversely, we say that *verification catches* $\mathbf{t}$ *misreported to* $\mathbf{b}$ if $t^i(A(\mathbf{b})) > b^i(A(\mathbf{b}))$ for some agent i.

Mechanisms (with verification) which are resistant to coalitions of $c \geq 1$ colluding agents that can exchange side payments satisfy the following definition.

Definition 4 (c-truthfulness [GH05]) *A mechanism (with verification) is* c-truthful *if, for any coalition of size at most c 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.*

Mechanisms (with verification) satisfying the definition above only for $c = 1$ are called *truthful* mechanisms (with verification).

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. Thus the following known fact holds (for the sake of completeness we give a proof in Appendix A:

Fact 5 *A mechanism (with verification) is c-truthful if and only if, for any coalition C of size at most c and for any two type vectors* $\mathbf{t}$ *and* $\mathbf{b}$ *such that C can misreport* $\mathbf{t}$ *to* $\mathbf{b}$*, the corresponding agents' utilities satisfy*

$$\sum_{i \in C} \text{utility}^i(\mathbf{t}) \geq \sum_{i \in C} \text{utility}^i(\mathbf{b}). \tag{4}$$

[3]In the Game Theory literature A is often referred to as *social choice function*.

Notation. Given a type vector $\mathbf{v} = (v^1, \dots, v^n)$, we let $\mathbf{v}^{-i} := (v^1, \dots, v^{i-1}, v^{i+1}, \dots, v^n)$ that is the vector of length $n-1$ obtained by removing v^i from $\mathbf{v}$. Finally, we let $(w, \mathbf{v}^{-i}) := (v^1, \dots, v^{i-1}, w, v^{i+1}, \dots, v^n)$ denote the vector obtained by replacing the i-th entry of $\mathbf{v}$ with w.

2 Optimization problems and exact algorithms

We focus on algorithms which minimize some global cost function of interest. Our ultimate goal is to derive a general technique to augment these algorithms with a suitable payment function so that the resulting mechanism with verification is truthful or even n-truthful.

2.1 Cost functions

We will consider problems where the goal is to minimize some cost function

$$\mathrm{Cost}(x, \mathbf{t}) = \mathrm{Cost}(t^1(x), \dots, t^n(x)) \tag{5}$$

which is monotone *non-decreasing* in each $t^i(x)$.

Example 6 (utilitarian problems) *Utilitarian problems are those corresponding to the case*

$$\mathrm{Cost}(x, \mathbf{t}) = \sum_i \alpha_i \cdot t^i(x)$$

where $\alpha_i \geq 0$ are constants. VCG mechanisms [Vic61, Cla71, Gro73] are truthful and guarantee exact solutions for any such problem, without using any verification. For arbitrary domains, the only truthful mechanisms are those that minimize some utilitarian cost function [Rob79].[4]

Example 7 (min-max problems) *These problems deal with cost functions*

$$\mathrm{Cost}(x, \mathbf{t}) = \max_i t^i(x)$$

and no truthful mechanism without verification can achieve exact or even approximate solutions [NR01, CKV07, KV07, MS07, Gam07, LS07]. These problems admit $(1+\varepsilon)$-approximate collusion-resistant mechanisms with verification for arbitrary bounded domains [PV09], for any $\varepsilon > 0$.

Example 8 (weakly utilitarian) *Weakly utilitarian cost functions "mix" utilitarian costs with arbitrary costs:*

$$\mathrm{Cost}(x, \mathbf{t}) = \sum_i \alpha_i \cdot t^i(x) + \mathrm{AnyCost}(x, \mathbf{t})$$

where $\alpha_i > 0$ are constants and $\mathrm{AnyCost}()$ is an arbitrary monotone non-decreasing cost function. These problems admit exact collusion-resistant mechanisms with verification for arbitrary bounded domains [PV09].

[4]Robert's theorem [Rob79] states that, if we impose no restriction on the agents' domains, then every truthful mechanism is an *affine maximizer* which, is our terminology, means that there exists a subset $S \subseteq \mathcal{S}$ of solutions, constants $\alpha_i \geq 0$, and constants $\{\beta_x\}_{x \in S}$ such that the algorithm guarantees $A(\mathbf{t}) \in \arg\min_{x \in S}\{\beta_x + \sum_i \alpha_i \cdot t^i(x)\}$ for all $\mathbf{t}$ in the domain. By adding a dummy player to incorporate the constants β_x, the condition says that the algorithm minimizes some utilitarian cost function over a fixed subset of solutions.

1. Fix a subset S of the feasible solutions and some order over these solutions (this step is independent of the bids in input);
2. According to the fixed order above, return the first solution minimizing $\mathrm{Cost}(x, \mathbf{b})$ over all x in S, where $\mathbf{b}$ are the bids in input.

Figure 1: A class of exact algorithms.

Our goal is to show that exact truthful mechanisms with verification are possible for any cost cost function of the form (5). This includes utilitarian and min-max problems as special cases, and a subclass of weakly utilitarian ones in which AnyCost() is of the form (5).

Remark 9 One might consider *arbitrary* monotone nondecreasing costs that need not obey the form in (5). In this case, truthful mechanisms with verification can be obtained for *finite domains* [Ven06] or by restricting to weakly utilitarian costs [PV09]. The latter mechanisms are extremely powerful in that they guarantee collusion resistance, a much stronger version of truthfulness, but they can only achieve *approximate* solutions for several problems (for instance min-max or others that do not obey the form (5)). It is then natural to ask if there is a *general* result saying that for all costs of the form (5) it is possible to have *exact* truthful mechanisms with verification.

2.2 Algorithms

To obtain truthful mechanisms with verification, we need to impose some restriction on the algorithm. In Figure 1 we show a class of algorithms that optimize some cost function Cost(). Intuitively speaking, these algorithms enjoy two important properties: they optimize the cost function (over some set of solutions) and they break ties in a fixed manner. These two properties will suffices for obtaining truthful mechanisms with verification. Instead of restricting to the algorithms in Figure 1, we consider all algorithms that satisfy these two conditions:

Definition 10 (exact algorithm) *An algorithm is exact if it is optimal in the range of solutions that it possibly returns and it breaks ties in a fixed manner. That is, for any* $\mathbf{b}$ *and* $\mathbf{b}'$ *in the domain,* $\mathrm{Cost}(A(\mathbf{b}), \mathbf{b}) \leq \mathrm{Cost}(A(\mathbf{b}'), \mathbf{b})$*. Moreover, there exists a total order* $\preceq$ *over the solutions such that, if for* $\mathbf{b}$ *and* $\mathbf{b}'$ *it holds that* $\mathrm{Cost}(A(\mathbf{b}), \mathbf{b}) = \mathrm{Cost}(A(\mathbf{b}'), \mathbf{b})$ *then* $A(\mathbf{b}') \preceq A(\mathbf{b})$*.*

In Section 4, we shall prove that any such algorithm admits a truthful mechanism with verification for arbitrary bounded domains. Mechanisms that return exact solutions for any such problem correspond to the case $S = \mathcal{S}$ in Figure 1. Note also that the class of exact algorithms is very reach: it contains many of the algorithms that have been proposed in the literature for obtaining truthful mechanisms without verification [NR07, BKV05, AT01, AAS05, MS07] and with verification [ADP+06, Ven06].

3 Collusion-resistant mechanisms for single-parameter agents

In this section we consider the case of *single-parameter* agents (see e.g. [GH05]). Here, each solution partitions the agents into two sets: those that are selected and those that are not selected. The value $t^i(x)$ depends uniquely on the fact that i is selected in x or not and it is completely specified by a *parameter* t_i, which is a real number such that

$$t^i(x) = \begin{cases} t_i & \text{if } i \text{ selected in } x, \\ 0 & \text{if } i \text{ not selected in } x. \end{cases} \tag{6}$$

Whether i is selected in x is publicly known, for every solution x, and thus each agent can only specify (and misreport) the parameter t_i. We assume *single-parameter bounded domains*, that is, each parameter t_i belongs to the interval $[0, \ell]$. From (6) we immediately get the following:

Fact 11 *For single-parameter agents, it holds that verification does not catch* $\mathbf{t}$ *misreported to* $\mathbf{b}$ *if and only if* $t_i \leq b_i$ *for every* i *selected in* $A(\mathbf{b})$*.*

In the sequel we will provide sufficient conditions for the existence of c-truthful mechanisms, for any given $c \leq n$.

3.1 A general sufficient condition for c-truthfulness

We begin with a *necessary* condition. Observe that in order to have truthful mechanisms for single-parameter agents the algorithm must select agents "monotonically", even when using verification [ADPP09]:

Definition 12 (monotone) *We say that algorithm* A *is* monotone *if the following holds. Having fixed the bids of all agents but* i*, agent* i *is selected if bidding a cost less than a threshold value* $b_i^\oplus$*, and is not selected if bidding a cost more than a threshold value* $b_i^\oplus$*. In particular, for every* $\mathbf{b} \in D$ *and for every* i*, there exists a value* $b_i^\oplus$ *which depends only on* $\mathbf{b}^{-i}$ *such that*

1. *i is selected in $A(b^i, \mathbf{b}^{-i})$ for all $b^i < b_i^\oplus$*
2. *i is not selected in $A(b^i, \mathbf{b}^{-i})$ for all $b^i > b_i^\oplus$.*

From Definition 12 we can easily obtain the following:

Fact 13 *If* A *is monotone and* i *is selected in* $A(\mathbf{b})$*, then* $b_i \leq b_i^\oplus$*. Moreover, if* i *is not selected in* $A(\mathbf{b})$ *then* $b_i \geq b_i^\oplus$*. Hence, for bounded domains the threshold values of Definition 12 are in the interval* $[0, \ell]$*.*

We next give a general *sufficient* condition for c-truthfulness on single-parameter bounded domains. In the next subsection, we show how this leads to a simpler condition for n-truthfulness in the case of exact algorithms.

Definition 14 (c-resistant) *We say that* $\mathbf{b}$ *is* c-different *from* $\mathbf{t}$ *if these two type vectors differ for at most* c *agents' types. A monotone algorithm* A *is* c-resistant *if, for every* $\mathbf{b}$ *which is* c*-different from* $\mathbf{t}$ *and such that verification does not catch* $\mathbf{t}$ *misreported to* $\mathbf{b}$*, it holds that* $t_i^\oplus \leq b_i^\oplus$ *for all* i *that are not selected in* $A(\mathbf{b})$*.*

Thinking of $\mathbf{t}$ as the true types and $\mathbf{b}$ as the bids, the c-resistant condition says that, if the solution at $\mathbf{b}$ does not select any of the underbidding agents, then the thresholds of all non-selected agents cannot decrease when moving from $\mathbf{t}$ to $\mathbf{b}$.

Theorem 15 *Every c-resistant algorithm A admits a c-truthful mechanism with verification for single-parameter bounded domains.*

Proof. We define the payment functions as follows:

$$p^i(\mathbf{b}) := \begin{cases} \hbar - b_i^{\oplus} & \text{if } i \text{ not selected in } A(\mathbf{b}) \\ \hbar & \text{otherwise} \end{cases} \tag{7}$$

where $\hbar := c \cdot \ell$.

Let us consider an arbitrary coalition C of size at most c and any two type vectors $\mathbf{t}$ and $\mathbf{b}$ such that C can misreport $\mathbf{t}$ to $\mathbf{b}$. Because of Fact 5, it suffices to prove (4). Either verification does not catch $\mathbf{t}$ misreported to $\mathbf{b}$ or verification catches $\mathbf{t}$ misreported to $\mathbf{b}$. We consider the two cases separately.

If verification catches $\mathbf{t}$ misreported to $\mathbf{b}$, then we have at least one agent $j \in C$ which does not receive any payment for $\mathbf{b}$. Moreover, the payment received by every other agent i in the coalition is at most $\hbar$. Hence, we have

$$\sum_{i \in C} \text{utility}^i(\mathbf{b}) \leq (c-1)\hbar = c\hbar - \hbar.$$

We next show that the utility of every truthtelling agent is at least $\hbar - \ell$. Indeed, the definition of $p^i()$ implies that $\text{utility}^i(\mathbf{t})$ is either $\hbar - t_i^{\oplus}$ if i not selected in $A(\mathbf{t})$, or $\hbar - t_i$ if i selected in $A(\mathbf{t})$. Fact 13 says that $t_i^{\oplus} \leq \ell$ and, if i selected in $A(\mathbf{t})$, then $t_i \leq t_i^{\oplus}$. Hence, $\text{utility}^i(\mathbf{t}) \geq \hbar - \ell$. From this and from our choice of $\hbar$, we obtain

$$\sum_{i \in C} \text{utility}^i(\mathbf{t}) \geq c(\hbar - \ell) = c\hbar - c\ell = c\hbar - \hbar.$$

The two inequalities above clearly imply (4).

If verification does not catch $\mathbf{t}$ misreported to $\mathbf{b}$ then we can show that for any $i \in C$ it holds

$$\text{utility}^i(\mathbf{t}) \geq \text{utility}^i(\mathbf{b}),$$

which clearly implies (4). There are four possible cases:

Case 1 (i selected in $A(\mathbf{t})$ and i selected in $A(\mathbf{b})$)**.** In this case nothing changes for i. Indeed, by the definition of $p^i()$, we have $\text{utility}^i(\mathbf{t}) = \hbar - t_i = \text{utility}^i(\mathbf{b})$.

Case 2 (i not selected in $A(\mathbf{t})$ and i selected in $A(\mathbf{b})$)**.** Fact 13 implies that $t_i^{\oplus} \leq t_i$. This and the definition of $p^i()$ imply $\text{utility}^i(\mathbf{t}) = \hbar - t_i^{\oplus} \geq \hbar - t_i = \text{utility}^i(\mathbf{b})$.

Case 3 (i not selected in $A(\mathbf{t})$ and i not selected in $A(\mathbf{b})$)**.** Since A is c-resistant, we have that $t_i^{\oplus} \leq b_i^{\oplus}$. This and the definition of $p^i()$ imply $\text{utility}^i(\mathbf{t}) = \hbar - t_i^{\oplus} \geq \hbar - b_i^{\oplus} = \text{utility}^i(\mathbf{b})$.

Case 4 (i selected in $A(\mathbf{t})$ and i not selected in $A(\mathbf{b})$)**.** Since i selected in $A(\mathbf{t})$, Fact 13 implies $t_i \leq t_i^{\oplus}$. Since i not selected in $A(\mathbf{b})$ and as A is c-resistant, we have that $t_i^{\oplus} \leq b_i^{\oplus}$, thus implying $t_i \leq b_i^{\oplus}$. This and the definition of $p^i()$ imply $\text{utility}^i(\mathbf{t}) = \hbar - t_i \geq \hbar - b_i^{\oplus} = \text{utility}^i(\mathbf{b})$.

This concludes the proof. □

3.2 A simpler sufficient condition for exact algorithms

In this section we show that, for exact algorithms, a "threshold-monotonicity" condition suffices for obtaining n-truthful mechanisms. The advantage is that this condition is simpler to exhibit because we only need to see what happens when increasing exactly one bid:

Definition 16 (threshold-monotone) *A monotone algorithm A is* threshold-monotone *if, for every $\mathbf{t}$ and every $\mathbf{b}$ obtained by increasing one agent entry of $\mathbf{t}$, the inequality $t_i^{\oplus} \leq b_i^{\oplus}$ holds for all i, where $t_i^{\oplus}$ and $b_i^{\oplus}$ are the threshold values of Definition 12.*

Our strategy is to show that threshold-monotonicity implies n-resistance and then apply the result in the previous section (Theorem 15). We begin with the following technical lemma:

Lemma 17 *Let A be an exact algorithm for single-parameter agents. Consider any two vectors $\mathbf{b}$ and $\tilde{\mathbf{b}}$ which differ only in the agent i's entry. If it holds that i is not selected in $A(\mathbf{b})$ and i is not selected in $A(\tilde{\mathbf{b}})$, then $A(\mathbf{b}) = A(\tilde{\mathbf{b}})$.*

Proof. From i not selected in $A(\mathbf{b})$ and i not selected in $A(\tilde{\mathbf{b}})$ and since the two vectors agree in every coordinate other than i, we have the following two identities:

$$\mathrm{Cost}(A(\mathbf{b}), \mathbf{b}) = \mathrm{Cost}(A(\mathbf{b}), \tilde{\mathbf{b}}) \quad \text{and} \quad \mathrm{Cost}(A(\tilde{\mathbf{b}}), \mathbf{b}) = \mathrm{Cost}(A(\tilde{\mathbf{b}}), \tilde{\mathbf{b}}).$$

From this and since A is an exact algorithm we obtain

$$\begin{aligned} \mathrm{Cost}(A(\mathbf{b}), \mathbf{b}) &\leq \mathrm{Cost}(A(\tilde{\mathbf{b}}), \mathbf{b}) && (8)\\ &= \mathrm{Cost}(A(\tilde{\mathbf{b}}), \tilde{\mathbf{b}}) \\ &\leq \mathrm{Cost}(A(\mathbf{b}), \tilde{\mathbf{b}}) && (9)\\ &= \mathrm{Cost}(A(\mathbf{b}), \mathbf{b}). \end{aligned}$$

Hence, the two inequalities must hold with '='. Since A uses a fixed tie breaking rule, from equalities (8) and (9) we have $A(\mathbf{b}) \preceq A(\tilde{\mathbf{b}})$ and $A(\tilde{\mathbf{b}}) \preceq A(\mathbf{b})$, respectively. Therefore, it holds $A(\mathbf{b}) = A(\tilde{\mathbf{b}})$.
□

The following result will be also useful for constructing mechanisms for arbitrary domains in Section 4.

Theorem 18 *Every threshold-monotone exact algorithm is n-resistant.*

Proof. By contradiction, assume that A is not n-resistant. That is, there exists a vector $\mathbf{b}$ which is n-different from $\mathbf{t}$ and such that verification does not catch $\mathbf{t}$ misreported to $\mathbf{b}$ with $t_i^{\oplus} > b_i^{\oplus}$, for some i not selected in $A(\mathbf{b})$. Consider the following two vectors obtained from $\mathbf{t}$ and $\mathbf{b}$, respectively, by replacing their i-th entry with the same value:

$$\begin{aligned} \tilde{\mathbf{t}} &:= (t_1, \ldots, t_{i-1}, \tilde{t}_i, t_{i+1}, \ldots, t_n) \\ \tilde{\mathbf{b}} &:= (b_1, \ldots, b_{i-1}, \tilde{b}_i, b_{i+1}, \ldots, b_n) \end{aligned}$$

with $\tilde{t}_i = \tilde{b}_i$ satisfying

$$b_i^{\oplus} < \tilde{b}_i = \tilde{t}_i < t_i^{\oplus}.$$

Observe that i is not selected in $A(\tilde{\mathbf{b}})$ and by Lemma 17 we have that the solution does not change, that is, $A(\mathbf{b}) = A(\tilde{\mathbf{b}})$. Next, we increase some of the entries of $\tilde{\mathbf{t}}$ in order to obtain a new vector $\hat{\mathbf{t}}$ such that $\hat{t}_j \geq \tilde{b}_j$ for all j. This implies the following inequality that will be used later in the proof:

$$\mathrm{Cost}(A(\hat{\mathbf{t}}), \tilde{\mathbf{b}}) \leq \mathrm{Cost}(A(\hat{\mathbf{t}}), \hat{\mathbf{t}}). \tag{10}$$

We define the vector $\hat{\mathbf{t}}$ as follows:

$$\hat{t}_j := \begin{cases} \tilde{b}_j & \text{if } \tilde{t}_j < \tilde{b}_j, \\ \tilde{t}_j & \text{otherwise.} \end{cases}$$

By definition of $\hat{\mathbf{t}}$ and since A is threshold-monotone, we have $\tilde{t}_i^{\oplus} \leq \hat{t}_i^{\oplus}$. Since $\tilde{\mathbf{t}}$ and $\tilde{\mathbf{b}}$ differ respectively from $\mathbf{t}$ and $\mathbf{b}$ only in the i-th entry, we have $\tilde{t}_i^{\oplus} = t_i^{\oplus}$ and $\tilde{b}_i^{\oplus} = b_i^{\oplus}$. In particular, since $\tilde{t}_i = \tilde{b}_i$, the construction of $\hat{\mathbf{t}}$ implies that $\hat{t}_i = \tilde{t}_i$. Putting all these things together we have

$$\tilde{b}_i^{\oplus} = b_i^{\oplus} < \tilde{b}_i = \tilde{t}_i = \hat{t}_i < t_i^{\oplus} = \tilde{t}_i^{\oplus} \leq \hat{t}_i^{\oplus}.$$

This implies that i is selected in $A(\hat{\mathbf{t}})$ and it is not selected in $A(\mathbf{b})$. Therefore these two solutions are different, that is, $A(\mathbf{b}) \neq A(\hat{\mathbf{t}})$.

We conclude the proof by showing that $A(\mathbf{b}) = A(\hat{\mathbf{t}})$, thus contradicting this fact. Towards this end, we prove another inequality to be used together with (10):

$$\mathrm{Cost}(A(\tilde{\mathbf{b}}), \hat{\mathbf{t}}) \leq \mathrm{Cost}(A(\tilde{\mathbf{b}}), \tilde{\mathbf{b}}). \tag{11}$$

Since verification does not catch $\mathbf{t}$ misreported to $\mathbf{b}$ we have that $t_j \leq b_j$ for every j selected in $A(\mathbf{b})$. This and the definition of $\hat{\mathbf{t}}$ imply that also $\hat{t}_j \leq b_j$ for every j selected in $A(\mathbf{b})$. Finally, since $A(\mathbf{b}) = A(\tilde{\mathbf{b}})$ and since i is not selected in $A(\mathbf{b})$, we can conclude that $\hat{t}_j \leq b_j = \tilde{b}_j$ for every j selected in $A(\tilde{\mathbf{b}})$. This inequality and the monotonicity of the cost function imply (11).

Then we have the following inequalities:

$$\begin{aligned}
\mathrm{Cost}(A(\hat{\mathbf{t}}), \hat{\mathbf{t}}) &\leq && \text{(since } A \text{ is an exact algorithm)} && (12)\\
\mathrm{Cost}(A(\tilde{\mathbf{b}}), \hat{\mathbf{t}}) &\leq && \text{(by Equation 11)} &&\\
\mathrm{Cost}(A(\tilde{\mathbf{b}}), \tilde{\mathbf{b}}) &\leq && \text{(since } A \text{ is an exact algorithm)} &&\\
\mathrm{Cost}(A(\hat{\mathbf{t}}), \tilde{\mathbf{b}}) &\leq && \text{(by Equation 10)} && (13)\\
\mathrm{Cost}(A(\hat{\mathbf{t}}), \hat{\mathbf{t}}). &&&&&
\end{aligned}$$

Hence, these inequalities must hold with '='. Since A uses a fixed tie breaking rule, from equalities (12) and (13) we obtain $A(\hat{\mathbf{t}}) \preceq A(\tilde{\mathbf{b}})$ and $A(\tilde{\mathbf{b}}) \preceq A(\hat{\mathbf{t}})$, respectively. This implies $A(\hat{\mathbf{t}}) = A(\tilde{\mathbf{b}})$. Since $A(\tilde{\mathbf{b}}) = A(\mathbf{b})$ and $A(\mathbf{b}) \neq A(\hat{\mathbf{t}})$ we reach a contradiction: $A(\hat{\mathbf{t}}) = A(\tilde{\mathbf{b}}) = A(\mathbf{b}) \neq A(\hat{\mathbf{t}})$. □

This theorem combined with Theorem 15 implies:

Corollary 19 *Every threshold-monotone exact algorithm admits an n-truthful mechanism with verification for single-parameter bounded domains.*

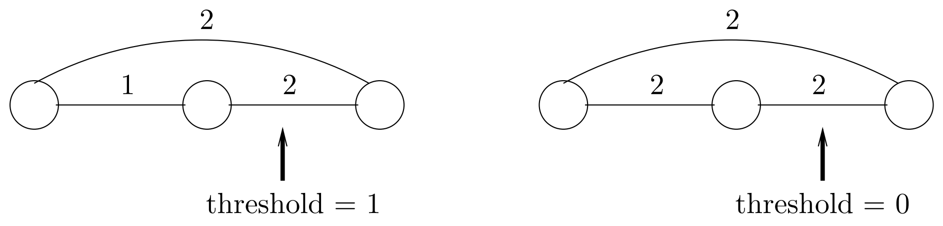

Figure 2: Path auctions are not threshold-monotone.

3.3 Two examples

Consider a procurement auction in which the mechanism wants two procure exactly k items. All items are identical and each agent can procure exactly one item. Selecting an agent means that she will have to procure the item. In a *k-items auction* the algorithm selects the agents corresponding to the k lowest bids, breaking ties arbitrarily. We consider only the case in which the number of agents is larger than the number of items we want to procure ($k < n$) because otherwise the problem is trivial.

Theorem 20 *Every k-items (procurement) auction is threshold-monotone.*

PROOF. By definition, an agent i is selected if her bid is below the k-th smallest value in $\mathbf{b}^{-i}$, and is not selected if it is larger than this value. Let $val_k(\mathbf{v})$ denote the k-th smallest value in a vector $\mathbf{v}$. For any $\mathbf{t}$ and $\mathbf{b}$ obtained from $\mathbf{t}$ by increasing one of its entries, we have $val_k(\mathbf{t}^{-i}) \leq val_k(\mathbf{b}^{-i})$. The proof thus follows from the observation that $t_i^{\oplus} = val_k(\mathbf{t}^{-i})$ and $b_i^{\oplus} = val_k(\mathbf{b}^{-i})$. □

The next example is the *path auction* that is a procurement auction in which agents correspond to the edges of communication network. The mechanism wants to procure the cheapest path connecting two nodes, where the cost of an edge is the type of the corresponding agent [NR01].

Theorem 21 *Path auctions are, in general, not threshold-monotone and the mechanism with verification using the payments of Theorem 15 is not even 2-truthful.*

PROOF. Figure 2 shows an instance for which the path auction is not threshold-monotone. We focus on the threshold of the edge (agent) connecting the middle node to the rightmost node. In the left instance, the threshold is 1: when the edge in question costs less than 1, the lower path is cheaper than the upper one and thus the agent in question is selected; conversely, when the cost of the edge is larger than 1 the upper path is cheaper and the agent in question is not selected. In the right instance, the threshold is 0 because the lower path is never better than the upper one. Hence, the path auction is not threshold-monotone because a smaller threshold has been obtained by increasing one type (see Definition 16)

We conclude by observing that a mechanism based on the payments of Theorem 15 would not guarantee 2-truthfulness: the coalition formed by the two lower agents can improve the utility of one of its members, while the other member has the same utility. Consider the types on the left as the true types and those on the right as the bids. Then the utility of the agent we have considered before improves from $\hbar - 1$ to $\hbar$, while the utility of the other agent remains $\hbar$. (Note that these agents are not selected in either scenario.) □

In the above proof we have shown that a coalition of two agent can improve the utility of one of its members, while the other member has the same utility. So, the mechanism is not bribeproof [Sch00]. We remark that path auctions are utilitarian and thus they admit n-truthful mechanisms via the technique in [PV09].

4 Truthful mechanisms for arbitrary bounded domains

In this section we derive truthful mechanisms for any exact algorithm over arbitrary bounded domains. The main idea is to regard each agent as a *coalition* of (virtual) single-parameter agents.

4.1 Arbitrary domains as coalitions of single-parameter agents

We call every agent whose domain is an arbitrary bounded domain a *multidimensional* agent. Since there are $\mathfrak{s} = |\mathcal{S}|$ feasible solutions, any type t^i in the domain of the multidimensional agent i can be seen as a vector

$$\mathbf{t}^i := (t_{i1}, \ldots, t_{i\mathfrak{s}}),$$

with $t_{ix} = t^i(x)$ for every feasible solution x. With this mapping in mind, we can regard each agent i as a coalition

$$C_i := \{i1, \ldots, i\mathfrak{s}\}$$

of (virtual) single-parameter agents with agent ij having type t_{ij}. In this "new game" we have the same set $\mathcal{S}$ of $\mathfrak{s}$ feasible solutions and, whenever solution x is chosen, in each coalition C_i only the (virtual) single-parameter agent ix is selected. This means that each coalition C_i has globally a cost identical to the cost, $t^i(x)$, of multidimensional agent i.

Consider an exact algorithm B over the multidimensional agents, and fix the bids $\mathbf{b}^{-i}$ of all agents but i. Then the resulting *single player function* $B(b^i, \mathbf{b}^{-i})$ can be seen as another exact algorithm $A(\mathbf{b}^i)$ whose domain (input) is restricted to the domains of the $\mathfrak{s}$ single-parameter agents in C_i.

4.2 The mechanism and its analysis

It turns out that every single player function $B(b^i, \mathbf{b}^{-i})$ as above is $\mathfrak{s}$-resistant. Based on this fact, we can apply the techniques developed for single-parameter agents and define the following class of mechanisms:

Definition 22 (threshold-based mechanism) *For any exact algorithm B we consider its* single player function, *depending on $\mathbf{b}^{-i}$, as $A(\mathbf{b}^i) := B(b^i, \mathbf{b}^{-i})$. In this case, we say that the single player function A has C_i as the set of* virtual *single-parameter agents. We define payment functions $q^i(b^i, \mathbf{b}^{-i}) := \sum_{j \in C_i} p^j(\mathbf{b}^i)$ where each $p^j()$ is the payment function of Theorem 15 when applied to A above and to the single-parameter agents in C_i. The resulting mechanism with verification (B, q) is called* threshold-based mechanism*.*

In the sequel we prove that every threshold-based mechanism is truthful *for multidimensional agents*. In order to prove this result, we first observe that the threshold-based mechanism needs only be resistant to the "known" coalitions defined above (recall that we have one virtual single-parameter agent per solution and thus coalitions are of size at most $\mathfrak{s}$):

Lemma 23 *If every single player function A of B is $\mathfrak{s}$-resistant with respect to its virtual single-parameter agents, then the threshold-based mechanism is truthful for the multidimensional agents.*

PROOF. We observe that the utility of a multidimensional agent i is the sum of the utilities of all single-parameter agents in the corresponding coalition C_i. Therefore, if (B, q) was not truthful, then the mechanism (A, p) would not be $\mathfrak{s}$-truthful. Since A is $\mathfrak{s}$-resistant, this would contradict Theorem 15. □

Lemma 24 *Every single player function A of an exact algorithm B is threshold-monotone.*

PROOF. Fix an agent i and an arbitrary bid vector $\mathbf{b}^{-i}$ for all other agents. Consider the corresponding single player function $A(\mathbf{b}^i) := B(b^i, \mathbf{b}^{-i})$, with b^i in the domain of the multidimensional agent i. We shall prove that A is threshold-monotone with respect to its virtual single-parameter agents.

We first show that A is monotone. By way of contradiction, suppose that a virtual single-parameter agent ij is not selected in $A(\mathbf{b}^i)$ but she is selected in $A(\tilde{\mathbf{b}}^i)$, where $\tilde{\mathbf{b}}^i$ obtained from $\mathbf{b}^i$ by increasing b_{ij} to $\tilde{b}_{ij} > b_{ij}$. For $x := A(\mathbf{b}^i) = B(b^i, \mathbf{b}^{-i})$ and $\tilde{x} := A(\tilde{\mathbf{b}}^i) = B(\tilde{b}^i, \mathbf{b}^{-i})$ we have that

$$\begin{array}{lll}
\text{Cost}(\tilde{x}, \tilde{\mathbf{b}}) & \geq & \text{(since } \tilde{b}_{ij} > b_{ij} \text{ and the cost function is monotone)} \\
\text{Cost}(\tilde{x}, \mathbf{b}) & \geq & \text{(since } B \text{ is an exact algorithm)} \\
\text{Cost}(x, \mathbf{b}) & = & \text{(since } ij \text{ is not selected in } x \text{ its cost is independent of } ij\text{'s entry)} \\
\text{Cost}(x, \tilde{\mathbf{b}}) & \geq & \text{(since } B \text{ is an exact algorithm)} \\
\text{Cost}(\tilde{x}, \tilde{\mathbf{b}}) & &
\end{array}$$

Since B uses a fixed ties breaking rule, these equalities imply that $x = \tilde{x}$, thus contradicting the fact that ij is selected only in one of these two solutions.

We next show that A is threshold-monotone. We proceed by way of contradiction and assume that there exist two vectors $\mathbf{t}^i$ and $\mathbf{b}^i$ such that the following happens. The vector $\mathbf{b}^i$ is obtained from $\mathbf{t}^i$ by increasing only the value of t_{ij} to some $b_{ij} > t_{ij}$

$$\begin{aligned}
\mathbf{t}^i &:= (t_{i1}, \ldots, t_{ij-1}, t_{ij}, t_{ij+1}, \ldots, t_{ia}) \\
\mathbf{b}^i &:= (t_{i1}, \ldots, t_{ij-1}, b_{ij}, t_{ij+1}, \ldots, t_{ia})
\end{aligned}$$

and there is a (virtual) single-parameter agent ik such that $t^{\oplus}_{ik} > b^{\oplus}_{ik}$. Observe that it must be $k \neq j$ since $\mathbf{t}^i$ and $\mathbf{b}^i$ are identical apart from the j-th entry and thus $b^{\oplus}_{ij} = t^{\oplus}_{ij}$. Consider the following two vectors obtained from $\mathbf{t}^i$ and $\mathbf{b}^i$, respectively, by replacing the k-th entry with the same value $\tilde{t}_{ik} = \tilde{b}_{ik}$ satisfying

$$b^{\oplus}_{ik} < \tilde{t}_{ik} = \tilde{b}_{ik} < t^{\oplus}_{ik}.$$

Since A is monotone, we have that ik is selected in $A(\tilde{\mathbf{t}}^i)$, ik is not selected in $A(\tilde{\mathbf{b}}^i)$, and $A(\tilde{\mathbf{b}}^i) = A(\mathbf{b}^i) = y$ because of Lemma 17. Since ik is not selected in y we have $k \neq y$ and thus $b_{iy} = \tilde{b}_{iy}$. That is, $b^i(y) = \tilde{b}^i(y)$ which implies the following identity:

$$\begin{aligned}
\text{Cost}(B(\tilde{b}^i, \mathbf{b}^{-i}), (\tilde{b}^i, \mathbf{b}^{-i})) &= \text{Cost}(b^1(y), \ldots, b^{i-1}(y), \tilde{b}^i(y), b^{i+1}(y), \ldots, b^n(y)) \\
&= \text{Cost}(b^1(y), \ldots, b^{i-1}(y), b^i(y), b^{i+1}(y), \ldots, b^n(y)) \\
&= \text{Cost}(B(\tilde{b}^i, \mathbf{b}^{-i}), (b^i, \mathbf{b}^{-i})). \qquad (14)
\end{aligned}$$

Putting things together we have

$$
\begin{array}{rcll}
\mathrm{Cost}(B(\tilde{t}^i, \mathbf{b}^{-i}), (\tilde{t}^i, \mathbf{b}^{-i})) & \leq & \text{(since } B \text{ is an exact algorithm)} & (15)\\
\mathrm{Cost}(B(t^i, \mathbf{b}^{-i}), (\tilde{t}^i, \mathbf{b}^{-i})) & = & \text{(since } j \neq k \text{ and thus } \tilde{t}_{ik} = t_{ik}) & \\
\mathrm{Cost}(B(t^i, \mathbf{b}^{-i}), (t^i, \mathbf{b}^{-i})) & \leq & \text{(since } B \text{ is an exact algorithm)} & (16)\\
\mathrm{Cost}(B(\tilde{b}^i, \mathbf{b}^{-i}), (t^i, \mathbf{b}^{-i})) & \leq & \text{(since } t_{il} \leq b_{il} \text{ for all } l) & \\
\mathrm{Cost}(B(\tilde{b}^i, \mathbf{b}^{-i}), (b^i, \mathbf{b}^{-i})) & = & \text{(from Equation 14)} & \\
\mathrm{Cost}(B(\tilde{b}^i, \mathbf{b}^{-i}), (\tilde{b}^i, \mathbf{b}^{-i})) & \leq & \text{(since } B \text{ is an exact algorithm)} & (17)\\
\mathrm{Cost}(B(\tilde{t}^i, \mathbf{b}^{-i}), (\tilde{b}^i, \mathbf{b}^{-i})) & = & \text{(since } k = A(\tilde{t}^i) = B(\tilde{t}^i, \mathbf{b}^{-i}) \text{ and } \tilde{b}_{ik} = \tilde{t}_{ik}) & \\
\mathrm{Cost}(B(\tilde{t}^i, \mathbf{b}^{-i}), (\tilde{t}^i, \mathbf{b}^{-i})). & & &
\end{array}
$$

Hence, these inequalities must hold with '='. Since B uses a fixed tie breaking rule, we have the following implications:

$$
\begin{array}{rcl}
(15) & \Longrightarrow & B(\tilde{t}^i, \mathbf{b}^{-i}) \preceq B(t^i, \mathbf{b}^{-i})\\
(16) & \Longrightarrow & B(t^i, \mathbf{b}^{-i}) \preceq B(\tilde{b}^i, \mathbf{b}^{-i})\\
(17) & \Longrightarrow & B(\tilde{b}^i, \mathbf{b}^{-i}) \preceq B(\tilde{t}^i, \mathbf{b}^{-i})
\end{array}
$$

This implies $B(\tilde{t}^i, \mathbf{b}^{-i}) = B(\tilde{b}^i, \mathbf{b}^{-i})$. This is a contradiction to the fact that ik is selected only in one of these two solutions: recall that agent ik is selected in $A(\tilde{\mathbf{t}}^i) = B(\tilde{t}^i, \mathbf{b}^{-i})$ and is not selected in $A(\tilde{\mathbf{b}}^i) = B(\tilde{b}^i, \mathbf{b}^{-i})$. We conclude that A is threshold-monotone. □

Theorem 25 *Every threshold-based mechanism is a truthful mechanism with verification if the domain is bounded.*

Proof. Since every single player function A is threshold-monotone (Lemma 24) it is also $\mathfrak{s}$-resistant (Theorem 18). The theorem thus follows from Lemma 23. □

Corollary 26 *Every exact algorithm admits a truthful mechanism with verification over any arbitrary bounded domain.*

In particular, by taking the exact algorithm in Figure 1 with $S = \mathcal{S}$ we obtain:

Corollary 27 *Every problem in which the domain is bounded and the cost function is monotone nondecreasing of the form (5) admits an exact truthful mechanism with verification.*

4.3 Extensions

In this section we discuss the optimality of our positive results and conclude by showing how to efficiently compute the payments (and the whole mechanism). Observe that exact truthful mechanisms can be obtained under the following two hypothesis:

1. The cost function is monotone non-decreasing in its arguments (Section 2.1);

2. The algorithm uses a fixed tie breaking rule (Section 2.2).

We next show that both hypothesis are necessary as relaxing any of them would make it impossible to have truthful mechanisms with verification.

Theorem 28 *For any cost function that is* not *monotone nondecreasing there exists a bounded domain such that no algorithm that minimizes such a cost function admits a truthful mechanism with verification.*

Proof. We show that, for any cost function which is *not* monotone nondecreasing, it is possible to construct a bounded domain such that no algorithm minimizing this cost function admits a truthful mechanism. The result applies to any set of solutions with at least two elements.[5]

Consider an arbitrary cost function Cost() which is not monotone nondecreasing meaning that there exist n nonnegative values $b_1, \ldots, b_n$ such that increasing the i-th value reduces the cost, that is,

$$\mathrm{Cost}(b_1, \ldots, b_i, \ldots, b_n) > \mathrm{Cost}(b_1, \ldots, c_i, \ldots, b_n) \quad \text{for some } c_i > b_i.$$

We think of each value b_j as the "constant type" which maps all solutions into the same fixed value b_j. The domain of each agent j contains this constant type b_j. Moreover, the domain of agent i contains the following two types. Fix a solution y and let us consider types

$$\alpha(x) := \begin{cases} b_i & \text{if } x = y \\ c_i & \text{otherwise,} \end{cases} \qquad \bar{\alpha}(x) := \begin{cases} b_i & \text{if } x \neq y \\ c_i & \text{otherwise.} \end{cases}$$

Consider the situation in which every agent j, excluding i, reports her constant type b_j. We first observe that, because of the inequality on the cost function, any exact algorithm must choose the solution that costs $c_i > b_i$ to agent i. That is, it must give the following output:

types	optimal solution
$(b_1, \ldots, \alpha, \ldots, b_n)$	$x \neq y$
$(b_1, \ldots, \bar{\alpha}, \ldots, b_n)$	y

We claim that, if there exists an exact truthful mechanism with verification (A, p), then the payments $p_\alpha := p^i(b_1, \ldots, \alpha, \ldots, b_n)$ and $p_{\bar{\alpha}} := p^i(b_1, \ldots, \bar{\alpha}, \ldots, b_n)$ should satisfy the following two inequalities:

$$p_\alpha - c_i \geq p_{\bar{\alpha}} - b_i \tag{18}$$
$$p_{\bar{\alpha}} - c_i \geq p_\alpha - b_i \tag{19}$$

By summing them up, we obtain $c_i \leq b_i$, which contradicts our initial hypothesis that $c_i > b_i$.

We conclude the proof by proving that the above inequalities are in fact implied by the truthfulness of (A, p). Essentially, they correspond to the case in which the true type of i is α and $\bar{\alpha}$, respectively, and and that reporting the other type does not improve i's utility. More in detail, suppose first that $t^i = \alpha$ and $b^i = \bar{\alpha}$ are the true and the reported types of i, respectively. On input $b^i = \bar{\alpha}$, the exact algorithm A outputs solution y. The important thing to observe is that agent i is not caught lying when reporting b^i because

$$t^i(A(\mathbf{b})) = t^i(y) = \alpha(y) = b_i < c_i = \bar{\alpha}(y) = b^i(A(\mathbf{b}))$$

[5]Obviously the case in which there is only one solution is of no interest.

where $\mathbf{b} = (b_1, \ldots, b^i, \ldots, b_n)$. Thus (18) is simply equivalent to the condition

$$\text{utility}^i(\mathbf{t}) \geq \text{utility}^i(\mathbf{b}) \tag{20}$$

for type vectors $\mathbf{t} = (b_1, \ldots, t^i, \ldots, b_n)$ and $\mathbf{b} = (b_1, \ldots, b^i, \ldots, b_n)$, when $t^i = \alpha$ and $b^i = \bar{\alpha}$. Similarly, we can show that (19) is equivalent to (20) when $t^i = \bar{\alpha}$ and $b^i = \alpha$. All we need to prove is that, also in this case, agent i is not caught lying when reporting b^i. Since the output of the exact algorithm on input $b^i = \alpha$ is some $x \neq y$, we have

$$t^i(A(\mathbf{b})) = t^i(x) = \bar{\alpha}(x) = b_i < c_i = \alpha(x) = b^i(A(\mathbf{b})).$$

We have thus shown that both (18) and (19) must hold, if the mechanism (A, p) is truthful, because truthfulness implies that (20) holds for all $\mathbf{t} = (b_1, \ldots, t^i, \ldots, b_n)$ and $\mathbf{b} = (b_1, \ldots, b^i, \ldots, b_n)$. (See Fact. 5 with coalition C consisting of agent i only.) Since (18) and (19) contradict $c_i > b_i$, we conclude that no exact truthful mechanism for this cost function exists if the domain D^i of agent i contains the types α and $\bar{\alpha}$, and the domain D^j of every other agent j contains the constant type b_j. In particular, the theorem holds for the finite domain containing only these types. $\square$

If we remove the "fixed tie breaking rule" assumption from the definition of exact algorithm, then it is no longer possible to obtain truthful mechanisms:

Theorem 29 *There exists a bounded domain and a monotone cost function such that the following holds. There exists an algorithm that minimizes such cost function (not using a fixed tie breaking rule) which does not admit any truthful mechanism with verification.*

Proof. The proof of the theorem is basically the same observation made by Archer and Tardos [AT01] that not all optimal mechanisms for minimizing the makespan on related machines are monotone. We give an algorithm and an instance for which each machine is allocated either one particular job or no jobs at all. So, the resulting problem can be seen as a single-parameter problem.

Consider the following algorithm for allocating two tasks on three related machines. Tasks have weights, and the algorithm allocates the task with larger weight to the fastest machine. This step is optimal for the makespan. Further, if it is possible to allocate the second task to the slowest machine without exceeding the current completion time (of the machine that gets the first task), then the second task is allocated to that machine. Otherwise, the second task is allocated so to minimize the makespan.

It is easy to see that this algorithm for two jobs minimizes the makespan. However, the algorithm does not use a fixed tie breaking rule and, because of this, it turns out to be not monotone according to Definition 12. This is shown by the following example in which jobs have weights 4 and 1, and the algorithm produces the following two allocations on input these types:

	$machine_1$	$machine_2$	$machine_3$
types	1	2	3
allocation	4	–	1
types	1	4	3
allocation	4	1	–

Following the terminology of Definition 12, we say that a machine (agent) is selected if it is allocated the second job.[6] To see that these agents are indeed single-parameter, we observe that the cost for

[6]Assuming the first machine has type 1 or smaller and that the other two machines have types 2 or larger, the first job goes always to the first machine. Notice that this gives a bounded domain.

a machine that is selected is equal to its type, and it is equal zero if not selected.[7] Then observe that $machine_2$ is not selected for type 2, while it is selected for type 4. That is, the algorithm is not monotone, though it minimizes the makespan which is clearly a monotone cost function. □

We conclude this section by observing that the mechanisms presented here have a further advantage of giving an explicit formula for the payments (see Equation 7 and Definition 22). In particular, this improves over the construction in [Ven06] since it gives efficient mechanisms for the case of arbitrary finite domains. The idea is to perform a binary search to determine the threshold values of Definition 12. For threshold-based mechanisms the running time is polynomial in the size of the input $\mathbf{t}$, where each t^i is a vector of $\mathfrak{s}$ values, one for each solution. Such an "explicit" representation of the input is in general necessary, as implied by communication complexity lower bounds for certain instances of combinatorial auction [NS06] which fall into the class of finite domains.

Corollary 30 *Every polynomial-time exact algorithm over an arbitrary finite domain admits a polynomial-time truthful mechanism with verification. For finite single-parameter domains, every polynomial-time c-resistant exact algorithm admits a polynomial-time c-truthful mechanism with verification.*

Acknowledgements. We wish to thank Riccardo Silvestri for several useful comments on an earlier version of this work.

References

[AAS05] Nir Andelman, Yossi Azar, and Motti Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *Proc. of STACS*, pages 69–82, 2005. 8

[ADP$^+$06] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, Giuseppe Persiano, and Carmine Ventre. New constructions of mechanisms with verification. In *Proc. of ICALP*, pages 596–607, 2006. 5, 8

[ADPP09] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. The power of verification for one-parameter agents. *Journal of Computer and System Sciences*, 75(3):190–211, 2009. 5, 9

[AT01] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of FOCS*, pages 482–491, 2001. 8, 18, 19

[BKV05] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *Proc. of STOC*, pages 39–48, 2005. 8

[CKV07] George Christodoulou, Elias Koutsoupias, and Agelina Vidali. A lower bound for scheduling mechanisms. In *Proc. of SODA*, pages 1163–1170, 2007. 4, 7

[7] In the general scenario considered by Archer and Tardos [AT01], the cost for an agent is the completion time of her machine, that is, the product of the work assigned to that machines times her type. In our instance, $machine_2$ and $machine_3$ receive exactly one unit of work and thus they are single-parameter agents. Also $machine_1$ can be regarded as a single-parameter agent by taking the type of this agent being equal to the cost for executing *four* units of work.

[Cla71] Edward H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971. 2, 7

[ESS04] Edith Elkind, Amit Sahai, and Ken Steiglitz. Frugality in path auctions. In *Proc. of SODA*, pages 701–709, 2004. 5

[Gam07] Iftah Gamzu. Improved lower bounds for non-utilitarian truthfulness. In *Proc. of WAOA*, volume 4927, pages 15–26, 2007. 4, 7

[GH05] Andrew V. Goldberg and Jason D. Hartline. Collusion-resistant mechanisms for single-parameter agents. In *Proc. of SODA*, pages 620–629, 2005. 3, 4, 6, 8

[Gro73] Theodore Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973. 2, 7

[KP99] Elias Koutsoupias and Christos H. Papadimitriou. Worst-case equilibria. In *Proc. of STACS*, pages 404–413, 1999. 1

[KV07] Elias Koutsoupias and Angelina Vidali. A lower bound of $1 + \phi$ for truthful scheduling mechanisms. In *Proc. of MFCS*, pages 454–464, 2007. 4, 7

[LS07] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *Proc. of EC*, pages 252–261, 2007. To appear in *Games and Economic Behavior*. 7

[LSZ08] Hagay Levin, Michael Schapira, and Aviv Zohar. Interdomain routing and games. In *Proc. of STOC*, pages 57–66, 2008. 3

[MS07] Ahuva Mu'alem and Michael Schapira. Setting lower bounds on truthfulness. In *Proc. of SODA*, pages 1143–1152, 2007. 4, 5, 7, 8

[NR01] Noam Nisan and Amir Ronen. Algorithmic Mechanism Design. *Games and Economic Behavior*, 35:166–196, 2001. Extended abstract in STOC'99. 1, 2, 3, 4, 7, 13

[NR07] Noam Nisan and Amir Ronen. Computationally Feasible VCG Mechanisms. *Journal of Artificial Intelligence Research*, 29:19–47, 2007. Extended abstract in EC'00. 8

[NS06] Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 129(1):192–224, 2006. 19

[Pap01] Christos H. Papadimitriou. Algorithms, Games, and the Internet. In *Proc. of STOC*, pages 749–753, 2001. 1

[PV09] Paolo Penna and Carmine Ventre. Optimal collusion-resistant mechanisms with verification. In *Proc. of EC*, pages 147–156, 2009. Full version available at http://www.dia.unisa.it/~penna/papers/collusion.pdf. 3, 4, 7, 8, 14

[PW05] Guido Proietti and Peter Widmayer. A truthful mechanism for the non-utilitarian minimum radius spanning tree problem. In *Proc. of SPAA*, pages 195–202, 2005. 4

[Rob79] Kevin Roberts. The characterization of implementable choice rules. *Aggregation and Revelation of Preferences*, pages 321–348, 1979. 7

[Sch00] James Schummer. Manipulation through bribes. *Journal of Economic Theory*, 91(3):180–198, 2000. 3, 4, 14

[Ven06] Carmine Ventre. Mechanisms with verification for any finite domain. In *Proc. of WINE*, volume 4286, pages 37–49, 2006. 4, 5, 8, 19

[Vic61] William Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961. 2, 3, 7

A Proof of Fact 5

In order to prove Fact 5, 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 4, 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

$$\sum_{i \in C} \text{utility}^i((\mathbf{t}^C, \mathbf{b}^{-C})) \geq \sum_{i \in C} \text{utility}^i(\mathbf{b}). \tag{21}$$

The condition in Fact 5 can be rewritten as follows:

(condition in Fact 5) 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, C can misreport $\mathbf{u}$ to $\mathbf{v}$ and thus $\mathbf{v} = (\mathbf{v}^C, \mathbf{u}^{-C})$, it holds that

$$\sum_{i \in C} \text{utility}^i(\mathbf{u}) \geq \sum_{i \in C} \text{utility}^i(\mathbf{v}). \tag{22}$$

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

(c-truthfulness)$\Rightarrow$(condition in Fact 5). We define vectors $\mathbf{u}$ and $\mathbf{v}$ by means of $\mathbf{t}$ and $\mathbf{b}$ as in (21). Consider the following vectors: $\mathbf{u} := (\mathbf{t}^C, \mathbf{b}^{-C})$ and $\mathbf{v} := (\mathbf{b}^C, \mathbf{b}^{-C})$. Thus (22) follows from (21).

(c-truthfulness)$\Leftarrow$(condition in Fact 5). We define type vectors $\mathbf{t}$ and $\mathbf{b}$ as function of $\mathbf{u}$ and $\mathbf{v}$ as in (22). Take $\mathbf{t} := \mathbf{u}$ and $\mathbf{b} := (\mathbf{v}^C, \mathbf{u}^{-C})$. Thus (21) follows from (22).

This completes the proof. $\square$