Obviously Strategyproof Mechanisms without Money for Scheduling

Paper #131

ABSTRACT

We consider the scheduling problem when no payments are allowed and the machines are bound by their declarations. We are interested in a stronger notion of truthfulness termed obvious strategyproofness (OSP) and explore its possibilities and its limitations. OSP formalizes the concept of truthfulness for agents/machines with a certain kind of bounded rationality, by making an agent's incentives to act truthfully obvious in some sense: roughly speaking, the worst possible outcome after selecting her true type is at least as good as the best possible outcome after misreporting her type. Under the *weaker* constraint of truthfulness, Koutsoupias [2011] proves a tight approximation ratio of $\frac{n+1}{2}$ for one task. We wish to examine how this guarantee is affected by the strengthening of the incentive compatibility constraint. The main message of our work is that there is essentially *no* worsening of the approximation guarantee corresponding to the significant strengthening of the guarantee of incentive-compatibility from truthfulness to OSP. To achieve this, we introduce the notion of strict monitoring and prove that such a monitoring framework is essential, thus providing a complete picture of OSP with monitoring in the context of scheduling a task without money.

KEYWORDS

Obvious strategyproofness; Extensive-form mechanisms without money; Monitoring; Machine scheduling

1 INTRODUCTION

Consider the very basic situation in which we need to allocate a set of tasks to a set of agents controlling machines, each with different and private processing capabilities, a.k.a. *types*. Agents are assumed to be selfish and strategic, so they might misreport the time they need to execute the tasks in an attempt to minimize their own running time after the allocation. We are interested in designing allocation protocols that do not use payments and the stable outcomes are not far from the non-strategic, centrally enforced optimum *makespan*, i.e., the completion time of the schedule.

A primary designer goal that has been extensively studied is that of *truthfulness*, which informally implies that a player should be able to optimize her own individual utility by reporting truthfully. For the case of a single task, and when payments are allowed, we know that a second-price auction that allocates the task to the machine declaring the lowest execution time and compensates her with a payment equal to the second lowest declaration, incentivizes truth-telling. In other words, under the second-price auction rational agents are expected to *truthfully* reveal their execution times and the task can then be optimally allocated to the machine that minimizes its completion time. However, the truthfulness of the second-price auction is not always easy to grasp and its transparency depends also on its *implementation.* Indeed, lab experiments show that people will lie to a sealed-bid implementation (Vickrey auction) more often than to a descending-price implementation of the second-price auction [4]. Recently, Li [2017] provided an explanation for this phenomenon by introducing the notion of Obviously Strategy-Proof (OSP) mechanisms; descending auctions are OSP mechanisms, while sealed-bid auctions are not. Li characterized this solution concept as the correct one for people with limited contingent reasoning skills, thus formalizing the concept of truthfulness for agents with a certain kind of bounded rationality.

In many cases, though, the use of payments might be considered unethical, illegal (e.g. organ donations) or even just impractical (see, e.g., [22]). Yet, truthfulness remains a very desirable property that can help the mechanism designer know what kind of behavior to expect from the agents, and for this reason, researchers have started turning their attention to possible ways of achieving truthfulness without the use of payments. Unfortunately, achieving truthfulness is not always compatible with maintaining a good objective value in the absence of payments [15, 23]. So, in order to achieve a reasonable (even simply bounded) approximation of the optimal solution, the use of additional machinery might be imperative. Koutsoupias [2011] provides a positive result in the context of scheduling without payments using the plausible assumption of *monitoring*, that is, the machines are bound by their declarations. In other words, the mechanism *monitors* the agents so that their declarations are checked against their behavior at run-time, and can force the machines to work longer than necessary (if they declared higher execution time than what they actually need) by enforcing that their execution time is compatible with their bid. This was influenced by the notion of impositions, according to which a mechanism can restrict the set of reactions available to a player after the outcome is chosen (see e.g. [12, 20]). A framework that is also very much related to monitoring is the notion of verification where agents are heavily punished when caught lying [7, 10] or, in presence of money, denied their payments [21]; these penalties are, in some cases, in addition to being monitored [19]. Variants of monitoring have been studied extensively in the literature (under slightly different names) [7, 8, 11, 13, 14, 17, 20, 24].

Overall, monitoring and similar notions, are paradigms that give the mechanism designer additional power that can help the prevention of lies. The designer can exert this power in a number of real-life situations, e.g., when agents and designer are in the same physical space or when the designer can control the legislative environment. Clearly, it might be trickier to motivate in some other cases. Nonetheless, as from the related literature, *whenever* the

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 2019, Montreal, Canada

https://doi.org/doi

mechanism designer has the power to implement this paradigm, then many impossibility results can be bypassed. In particular, for the case of fully rational machines, Koutsoupias provides a truthful-in-expectation (randomized) mechanism that allocates a single task and achieves an $(n+1)/2$-approximation of the optimum makespan (completion time) while proving that this is best possible.[1] The question that remains is:

> *To what extent can we optimally allocate the task to our set of n selfish agents when they have bounded rationality?*

More specifically, we seek a quantitative answer to the following question: *How much does the approximation guarantee deteriorate when we strengthen the constraint of incentive-compatibility from truthfulness to OSP*? The literature on OSP commonly considers mechanisms that are not direct revelation but rather implemented as an extensive-form game through what is called the mechanism's *implementation tree* (see, e.g., [6]). Such a tree models the steps that the mechanism takes according to the actions of the agents. In particular, an agent is associated with each vertex of the tree (the *divergent* agent) and actions (mapped to the types) are associated with the edges. The mechanisms starts at the root and traverses a path of the tree. At each step the mechanism asks the corresponding divergent agent to act and select one of the outgoing edges; this corresponds to the agent declaring that her type belongs to the set of types associated with the corresponding edge. Each leaf node of the tree is associated with an outcome which the mechanism will enforce if the execution path reaches that leaf.

Extending the definition of monitoring to these extensive-form mechanisms is not entirely straightforward. For example, the execution of the mechanism might follow a path that does not involve all the agents. Or, it might be that, in the chosen execution path, a divergent agent selected an edge with more than one types associated with it in the last time that she acted. Both of these cases show that we might reach a leaf of the tree without having requested that all agents declare their exact type. The question that arises is: *to what type compatible with the history should we tie the cost of the monitored agent?* As an answer this question, we introduce the notion of *strict* monitoring, where we assume that monitoring is enforced according to the maximum type of each agent that is compatible with the history. While this is clearly a very rigid notion, two complementary observations can be made to mitigate and motivate its use. Firstly, we give a complete picture of the power of monitoring in this context, as we show that *any other definition* of monitoring for extensive-form games leads to unbounded approximation guarantees; strict monitoring is then essential to overcome impossibility results. Secondly, many countries adopt legislation to deal with incomplete declarations (of income, mainly). For example, in New Zealand higher tax rates are used in the cases in which people do not provide their Inland Revenue Department number when collecting their investment incomes. In Italy, a business declaring an income that is not in line with the sector average are subject to a presumptive income taxation – i.e., the tax man *rounds up* the income to the sector average and issues a tax bill for that amount. The onus is on the business to prove that the income is effectively lower.[2]

[1]When applied to many tasks, this mechanism immediately implies an $n(n+1)/2$ approximation ratio for the makespan objective. A different algorithm achieving an approximation ratio of n was later provided in [14].

1.1 Our contribution

We here study the problem of scheduling with monitoring when payments are not allowed and examine the repercussion of a bounded rationality assumption by considering OSP mechanisms. We consider the results in [16] for fully rational agents, as a benchmark. As noted above, for the case of scheduling a single task Koutsoupias [16] proved that the approximation ratio of any (randomized) truthful mechanism is at least $(n+1)/2$ and gave a mechanism matching this bound, where n is the number of machines. The main message of our work is that there is essentially *no* worsening of the approximation guarantee corresponding to the significant strengthening of the guarantee of incentive-compatibility from truthfulness to OSP, as long as the mechanism designer can implement a particular notion of monitoring.

Specifically, our work provides a complete picture of OSP with monitoring in the context of scheduling a task without money. We design an OSP mechanism that returns approximation $\alpha < n$ when we have n machines and we show that this is *tight*. The actual bound depends on the agents' type *domain* (i.e., the allowed set of types – execution times – that the agents can declare to the mechanism) and gets smaller the smaller the difference between maximum and minimum type therein. Therefore, the price to pay to turn truthfulness into OSP is less than a factor 2 loss in the approximation guarantee.

Another contribution of our work is the notion of strict monitoring which is suited for extensive-form mechanisms and allows us to bypass the inapproximability results in the case of large domains. In particular, we show that strict monitoring is necessary and sufficient to obtain bounded approximation ratios; any other (weaker) notion of monitoring[3] leads to a strong lower bound on the approximation guarantee of OSP mechanisms.

Our work introduces in fact two new techniques to lower bound the approximation guarantee of OSP mechanisms without money. The first adapts the ideas in [6] to the setting without money; specifically, we identify a property that implementation trees need to satisfy in order to guarantee a "good" approximation, and then combine this with OSP constraints. This is used to prove that the approximation of our mechanism is tight (Lemma 3.3). The second technique, instead, reverses the approach and the approximation analysis yields structural properties on the implementation trees of OSP mechanisms with bounded approximation. Specifically, we mimic the typical structure of a lower bound to the approximation guarantee of a truthful mechanism and combine approximation and OSP constraints to reach the conclusion that one needs a careful definition of monitoring to obtain bounded approximations (Lemma 4.1). Interestingly, we then show how this result can be used to

[2]In these examples, it is the case that agents choose to withhold information from the system; in our case, it is the system – mechanism – itself which might prevent full disclosure. However, since agents accept to work for free there must be an imbalance of power between themselves and the designer just like in those examples. Therefore, it is arguably the case that the agents will accept the rules of the system, both in terms of declaration and punishment.

[3]It is folklore that, already for truthful mechanisms, no bounded approximation is possible without monitoring.

characterize the queries that need to be done to marry bounded approximation and OSP.

We finally consider the extension to many tasks, the so-called scheduling unrelated machines problem. We observe that we can sequentially allocate each task to the machines preserving OSP, and obtain an αn approximation. We leave as an open problem to understand whether this is the best possible as opposed to truthfulness where it is known that n-approximation is possible [14].

2 PRELIMINARIES

In the unrelated machines scheduling problem, we are given m tasks that need to be allocated to n machines. Each machine has job-dependent types $t_{i,j} \in D$ that correspond to the time machine i needs to execute job j, D denoting the domain of agents' types. Types are private knowledge of the machines. We seek to design mechanisms that probabilistically allocate the jobs to minimize the *makespan*, defined as the completion time of the schedule. We focus on mechanisms that are OSP, in that they incentivate truthtelling even if the machine are not perfectly rational. Before we can present the OSP notion, we need to give some background.

An extensive-form mechanism $\mathcal{M}$ is a pair $(f, \mathcal{T})$, where f is an algorithm (also termed social choice function) that takes as input bid profiles and returns a probabilistic allocation of tasks to the machines (f is essentially a mapping between bid profiles and outcomes), and $\mathcal{T}$ is an extensive-form game that we call implementation tree.[4] Each internal node u of $\mathcal{T}$ is labelled with a player $S(u)$, called the divergent agent at u, and the outgoing edges from u are labelled with types in the domain of $S(u)$ that are compatible with the history leading to u; the edges' labels denote a partition of the compatible types. We denote by $D_i(u)$ the types in the domain of i that are compatible with the history leading to u. The tree models how $\mathcal{M}$ interacts with the agents: at node u the agent $S(u)$ is queried and asked to choose a strategy that will effectively select one of the type-sets in the partition. The leaves of the tree will then be linked to (a set of) bid profiles and f will return the probability distribution used to allocate the tasks to the machines. (Observe that this means that the domain of f is effectively given by the leaves of $\mathcal{T}$.) We use $\mathbf{b}$ to denote bid profiles, so that $b_{i,j}$ stands for the execution time machine i "declared" (i.e., i played the game $\mathcal{T}$ as if her type were $(b_{i,j})_j$) for task j. For simplicity, we use $f(\mathbf{b})$ to denote the outcome of f for the leaf of $\mathcal{T}$ to which $\mathbf{b}$ belongs, i.e., $f_{i,j}(\mathbf{b})$ denotes the probability that task j is allocated to machine i for declarations/actions according to $\mathbf{b}$ (when we examine the case of a single task we drop the dependence on j for ease of presentation). Figure 1 gives an example of an implementation tree for scheduling a task to two machines with a three-value domain $\{L, M, H\}$. The root partitions the domain of machine 1 into $\{M, H\}$ and L. If we let v denote the internal node that is labelled with 1, then $D_1(v) = \{M, H\}$ as type L is no longer compatible with the history of v. Finally, v partitions $D_1(v)$ into $\{M\}$ and $\{H\}$.

We now define OSP with (strict) monitoring, the solution concepts of interest to our work; since our main focus is on the case of single task we will restrict our focus and notation to $m = 1$ in order to simplify the exposition.

OSP informally implies that whenever an agent is asked to diverge, she is better off acting according to her true type in *any* possible future scenario: the worst possible outcome after selecting her true type is at least as good as the best possible outcome after misreporting her type, at that particular point in the implementation tree. For the formal definition, we need to introduce some more notation. We call a bid profile $\mathbf{b}$ *compatible with* u if $\mathbf{b}$ is compatible with the history of u for all agents. We furthermore say that $(t, \mathbf{b}_{-i})$ and $(b, \mathbf{b}'_{-i})$ diverge at u if $i = S(u)$ and t and b are labels of different edges outgoing u. So, for example, (M, H) and (H, L) are compatible at node w on Figure 1 (labelled with 2) and diverge at that node, whilst (M, H) and (H, M) are compatible but do not diverge.

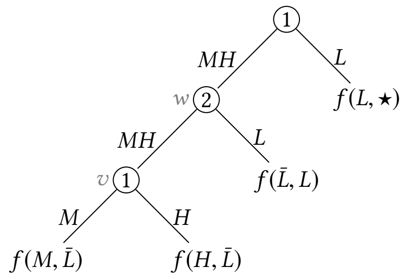

Figure 1: An implementation tree for scheduling a task to two machines with domain $\{L, M, H\}$; $\bar{L}$ denotes the fact that the type of a machine is in $\{M, H\}$ in the corresponding leaf, while $\star$ denotes the fact that the machine can have any type in the domain D in the corresponding leaf.

For every agent i and types $t, b \in D_i$, we let $u^i_{t,b}$ denote a vertex u in the implementation tree, $\mathcal{T}$, such that, for some $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}(u) = \times_{j \neq i} D_j(u)$, we have that $(t, \mathbf{b}_{-i})$ and $(b, \mathbf{b}'_{-i})$ are compatible with u but diverge at u . Note that such a vertex might not be unique as agent i will be asked to separate t from b in different paths from the root (but only once for every such path). We call these vertices of $\mathcal{T}$ *tb-separating* for agent/machine i. We are now ready to define OSP.

Definition 2.1 (OSP with monitoring). An extensive-form mechanism $\mathcal{M} = (f, \mathcal{T})$ is OSP with monitoring for scheduling a single task if for all i, $t, b \in D_i$, $u^i_{t,b} \in \mathcal{T}$, and $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}(u^i_{t,b})$, it holds that

$$t \cdot f_i(t, \mathbf{b}_{-i}) \leq \max\{t, b\} \cdot f_i(b, \mathbf{b}'_{-i}).$$

In the definition above, we can appreciate how monitoring bounds an agent to her declaration: machine i will execute the task for a time that is the maximum between her true type and her reported one. So, the expected cost/workload of machine i of type t declaring b is defined as $\max\{t, b\} \cdot f_i(b, \mathbf{b}'_{-i})$.

However, there is not an immediate analogue of declaration in extensive-form mechanisms. Let us turn our attention to Figure 1. Consider the right child of node w in the tree. Along this path, we have not given machine 1 the chance to separate M from H. What type should we use to monitor machine 1 who claimed her type to be either M or H? Strict monitoring implies that we assume the

[4]The definition of a mechanism as a pair identifies that in OSP, in addition to the social choice function, the design of the extensive-form implementation is essential to define the incentive constraints. The authors of [6] show how this is equivalent to Li's original definition and the ones used in subsequent work.

highest compatible type. Although this is a very strict assumption, we show later (Lemma 4.1) that it is indeed essential, as *any weaker monitoring notion* would lead to an unbounded approximation of the optimal makespan. Formally, if we let $\ell_{\mathbf{b}}$ denote the unique leaf of the tree with which $\mathbf{b}$ is compatible, we have the following.

Definition 2.2 (OSP with strict monitoring). An extensive-form mechanism $\mathcal{M} = (f, \mathcal{T})$ is OSP with strict monitoring for scheduling a single task if for all i, $t, b \in D_i$, $u^i_{t,b} \in \mathcal{T}$ and $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}(u^i_{t,b})$, it holds that

$$t \cdot f_i(t, \mathbf{b}_{-i}) \leq \max\{t, \max D_i(\ell_{(b,\mathbf{b}'_{-i})})\} \cdot f_i(b, \mathbf{b}'_{-i}).$$

We note that our definition implicitly assumes that we can identify the true processing time of a machine if we allocate them the task, and not monitor them if they are being truthful (if their true processing time is compatible with the leaf defining the outcome, but other types are also compatible with it). This way, we can avoid unfairly punishing truthful agents and only apply strict monitoring to dishonest agents. (This is similar to the Italian presumptive taxation system discussed above.) Our results continue to hold even in the case where such a type-identification mechanism is not available, and even the truthful agents are expected to (possibly unfairly) exert effort equal to the highest possible type compatible with the history.

Finally, we note that the above definitions of OSP are an immediate application of the definitions introduced by Li in his seminal paper [18] to scheduling – see discussion in [6]. OSP with monitoring has been considered already in [8]; however their definition fails to capture the intricacies of extensive-form mechanisms and mainly focuses on direct revelation. It is also worth remarking that our use of randomness is compatible with the literature on OSP. Randomness in Li's definition can come in two shapes: chance (internal) nodes and lotteries in the leaves (cf. footnote 15 in [18]). It has been noted in the literature how the former concept translates "universally truthful mechanisms" to OSP (see, for example, footnote 9 in [3]). The latter is our focus herein; the expectation is taken on each leaf and therefore fits with Li's arguments for bounded rationality. (Incidentally, a definition of OSP in expectation which also considers the effects of internal chance nodes on bounded rationality is given in a recent paper [9].)

3 TWO-VALUE DOMAINS

In this section we examine the case of two-value domains, i.e., $D = \{L, H\}$, $L < H$, and demonstrate a tight approximation ratio $\alpha < n$ for OSP mechanisms for scheduling. The mechanism that we design is reminiscent of an ascending-price auction. It looks for the smallest possible type (the fastest execution time) in the domain and as soon as it identifies an agent with that type, it immediately allocates the task with some carefully selected probability p to the corresponding machine and evenly splits, amongst the other agents, the remaining probability mass $1 - p$. (If no fast agent/machine is found, the task is allocated uniformly.) We show that the probabilities used by the mechanism are small enough to satisfy OSP while also large enough to guarantee a good approximation.

Definition 3.1 (Mechanism $\mathcal{M}_2$). $\mathcal{M}_2$ specifies an arbitrary order of the machines and asks the machines one after the other in that order if their type is L or H. Let machine i^* be the first machine who answers L. The mechanism will then stop querying the machines and produce the following allocation: the task is allocated with probability $p = \frac{H}{H+(n-1)L}$ to machine i^* and with probability $\frac{1-p}{n-1} = \frac{L}{H+(n-1)L}$ to each other machine. If no machine replies L, then the task is allocated with probability $1/n$ to each machine.

Theorem 3.2. *$\mathcal{M}_2$ is an OSP with monitoring*[5] *mechanism that achieves approximation ratio $\frac{Hn}{H+(n-1)L} < n$ for scheduling one task to n machines, when their type domain has two values, L and H.*

Proof. Consider the non-trivial case where $L < H$ and mechanism $\mathcal{M}_2$ as defined above (Definition 3.1). We start by proving that $\mathcal{M}_2$ is OSP with monitoring (cf. (1) and (2)). First note that for the specific allocation probabilities it holds that

$$\frac{1-p}{1-n} = \frac{L}{H + (n-1)L} < \frac{1}{n}.$$

Let a bidding vector of the form $(H^k, b, \star)$ denote that the first k machines declared H, the $(k+1)$-th machine declared b, and the machines that were potentially asked afterwards made an arbitrary declaration. We have that

$$L \cdot f_i(H^{i-1}, L) \leq H \cdot f_i(H^{i-1}, H, \star), \qquad \Longleftrightarrow \qquad (1)$$
$$L \cdot p \leq H \min\left\{\frac{1-p}{n-1}, \frac{1}{n}\right\} \qquad \Longleftrightarrow$$
$$L \frac{H}{H + (n-1)L} \leq H \frac{L}{H + (n-1)L},$$

which is true and confirms the OSP constraint for an agent whose true type is L. For an agent whose true type is H, we have

$$H \cdot f_i(H^{i-1}, H, \star) \leq H \cdot f_i(H^{i-1}, L) \qquad \Longleftrightarrow \qquad (2)$$
$$\max\left\{\frac{1-p}{n-1}, \frac{1}{n}\right\} \leq p \qquad \Longleftrightarrow$$
$$\frac{1}{n} \leq \frac{H}{H + (n-1)L},$$

which is again true and confirms the OSP property of the mechanism.

To see that our mechanism achieves an $\frac{Hn}{H+(n-1)L}$ approximation of the optimal makespan we observe that, should at least a machine have type L, the expected completion time of $\mathcal{M}_2$ is at most $pL + \frac{1-p}{n-1}H(n-1) = \frac{Hn}{H+(n-1)L} \cdot L$, while the optimal makespan is at least L. (The mechanism is optimal if no machine has type L.) □

The following result shows that the approximation ratio achieved in Theorem 3.2 is actually the best possible for two-value domains.

Lemma 3.3. *There is no OSP mechanism for scheduling one task to n machines with approximation ratio better than $\frac{Hn}{H+(n-1)L}$ even with monitoring, when their type domain has two values, L and H.*

Proof. Consider a mechanism $\mathcal{M}$ with approximation ratio α; let us consider a domain where $H > (2\alpha - 1)L$. We will prove that if $\mathcal{M}$ is OSP (with monitoring) then

$$\alpha \geq \frac{Hn}{H + (n-1)L}.$$

[5]Note that for two-value domains, monitoring and strict monitoring coincide.

$\mathcal{M}$ has to satisfy the following to have approximation ratio α: At any path of the implementation tree from the root to a leaf, the mechanism should have either discovered some machine with type L, or have asked at least $n-1$ machines to diverge between L and H. Assume to the contrary that there exists a path where all the divergent agents have played according to H and there are at least two machines who have not been asked to diverge. Let exactly one of these two machines have true type L. The best solution that $\mathcal{M}$ can adopt is to return the uniform distribution over these machines that has makespan at least $(L+H)/2$, which is more than α times the optimum (L) since $H > (2\alpha - 1)L$, a contradiction.

Consider a path of the implementation tree with the property above. Consider the point where the last machine was asked to diverge between L and H on that path. If she says L then she should get the task with probability $p_l \geq \frac{H-\alpha L}{H-L} = p$ to satisfy the approximation ratio guarantee of α. In that subtree/outcome, at least one other machine will get the task with probability at most $\frac{1-p}{n-1} = \frac{(\alpha-1)L}{(n-1)(H-L)}$, let's call that machine i^*. Now consider the point in the previous path where machine i^* was asked to diverge between L and H. In the subtree where i^* declared L there exists the leaf/outcome where everyone else has type H so i^* should get at least $\frac{H-\alpha L}{H-L} = p$ to guarantee approximation α.

Now consider the OSP constraint for i^* diverging at the point mentioned above:

$$\begin{aligned} Lp &\leq H\frac{1-p}{n-1} && \iff \\ L\frac{H-\alpha L}{H-L} &\leq H\frac{1}{n-1}\cdot\frac{(\alpha-1)L}{H-L} && \iff \\ \alpha &\geq \frac{Hn}{L(n-1)+H}, \end{aligned}$$

as desired. □

4 UNRESTRICTED DOMAINS

In this section we consider the general case of scheduling with large finite domains.[6] Although monitoring is enough to achieve a bounded approximation ratio for truthfulness [16], we here prove that it is not enough to get a bounded approximation ratio for OSP, while strict monitoring is sufficient.

Lemma 4.1. *There is no OSP mechanism with bounded approximation ratio for scheduling one task when the type domain has at least three values, even with monitoring when the type we use to monitor an agent is different from the highest compatible type. In other words, we cannot get a bounded approximation ratio without strict monitoring.*

Proof. Consider the case of two machines and type domain $D = \{t_1, t_2, \ldots, t_d\}$ where $t_1 < t_2 < \ldots < t_d$.

Let t_l, t_h be any two types of the domain such that $t_l < t_h$. The approximation guarantee for an instance (t_l, t_h) implies that

$$\begin{aligned} t_l \cdot f_1(t_l,t_h) + t_h(1-f_1(t_l,t_h)) \leq \alpha t_l &\iff \\ f_1(t_l,t_h) \geq \frac{t_h - \alpha t_l}{t_h - t_l}, & \qquad (3) \end{aligned}$$

[6]The assumption here is that the domain is finite; this appears to be an inherent limitation of OSP and mechanisms in extensive-form (and more in general extensive-form games); see, e.g., the discussion in [1] and the references therein.

Similarly, the approximation guarantee for an instance (t_h, t_l) implies that

$$\begin{aligned} t_h \cdot f_1(t_h,t_l) + t_l(1-f_1(t_h,t_l)) \leq \alpha t_l &\iff \\ f_1(t_h,t_l) \leq \frac{(\alpha-1)t_l}{t_h - t_l}. & \qquad (4) \end{aligned}$$

Consider the first divergent machine; without loss of generality let this be agent 1. Such a divergence must happen in order to have a bounded approximation ratio. Assume that the machine is asked to distinguish between P_1 and P_2 that form a partition of the type domain and let the smallest type in the domain belong to the first part, i.e., $t_1 \in P_1$. We denote by t_m the minimum type that appears in P_2 and distinguish cases depending on the cardinality of P_2.

We first consider the case where $|P_2| = 1$, i.e., $P_2 = \{t_m\}$, and assume for now that $t_m \neq t_2$. Consider the OSP constraint regarding the deviation from the divergent machine from true type $t_2 \in P_1$ to type t_m. We get

$$t_2 \cdot f_1(t_2, t_m) \leq t_m \cdot f_1(t_m, t_1).$$

From this and Inequality (4) we get that

$$\begin{aligned} \frac{t_2}{t_m}\cdot f_1(t_2,t_m) &\leq \frac{(\alpha-1)t_1}{t_m - t_1} && \iff \\ f_1(t_2,t_m) &\leq \frac{t_m}{t_2}\frac{(\alpha-1)t_1}{t_m-t_1}, \end{aligned}$$

while from this and Inequality (3) we get that

$$\begin{aligned} \frac{t_m - \alpha t_2}{t_m - t_2} &\leq \frac{t_m}{t_2}\frac{(\alpha-1)t_1}{t_m-t_1} && \iff \\ \alpha &\geq \frac{t_m t_2(t_m - t_1) + t_m t_1(t_m - t_2)}{t_2^2(t_m - t_1) + t_m t_1 (t_m - t_2)}. \end{aligned}$$

Regarding the RHS of the above inequality we note that

$$\begin{aligned} \alpha &\geq \frac{t_m t_2(t_m - t_1) + t_m t_1(t_m - t_2)}{t_2^2(t_m - t_1) + t_m t_1 (t_m - t_2)} \\ &\geq \frac{t_m^2 t_2 + t_m^2 t_1}{t_2^2 t_m + t_m^2 t_1} = \frac{t_m(t_2+t_1)}{t_m t_1 + t_2^2} \end{aligned}$$

which is increasing in t_m and hence unbounded.

The case where $P_2 = \{t_2\}$ leads to an equivalent argument if we consider the following OSP constraint regarding the deviation from the divergent machine from true type $t_2 \in P_2$ to type $t_d \in P_1$:

$$t_2 \cdot f_1(t_2, t_d) \leq t_d \cdot f_1(t_d, t_1).$$

This is true because in our arguments we have only used the relative order of t_1, t_2, and t_m, which is maintained if we substitute t_m with t_d.

We now consider the case where $|P_2| = 2$. Note that since by our assumption $t_1 \in P_1$, then we know that both elements of P_2 are larger than t_1. Let us denote them by t_m and t_h, such that $t_m < t_h$. We can prove the unboundedness of the approximation ratio by considering the following OSP constraint regarding the deviation from the divergent machine from true type $t_1 \in P_1$ to type $t_m \in P_2$:

$$t_1 \cdot f_1(t_1, t_h) \leq t_m \cdot f_1(t_m, t_1).$$

Note that the only monitoring paradigm that we can use here that is not strict monitoring, is the classical monitoring where the machine will be expected to execute the task for the declared time/type t_m. Similarly to the case above, using Inequality (4), we get that

$$\frac{t_1}{t_m} \cdot f_1(t_1, t_h) \leq \frac{(\alpha - 1)t_1}{t_m - t_1},$$

while by combining this and Inequality (3) we get

$$\frac{t_h - \alpha t_1}{t_h - t_1} \leq \frac{t_m}{t_1} \frac{(\alpha - 1)t_1}{t_m - t_1} \iff$$

$$\alpha \geq \frac{t_h(t_m - t_1) + t_m(t_h - t_1)}{t_1(t_m - t_1) + t_m(t_h - t_1)}.$$

The function at the RHS of the above inequality, which corresponds to the best approximation guarantee that we can achieve, is increasing in t_h, which means that it is unbounded.

It remains to consider the case where $|P_2| \geq 3$. Let $\{t_m, t_j, t_h\} \subseteq P_2$, where t_m and t_h are the minimum and maximum type in P_2. This is the most interesting case, as we can now observe how a monitoring paradigm that is not classical nor strict monitoring would work and prove that for any such paradigm we will still get an unbounded approximation ratio. Consider the OSP constraint regarding the deviation from the divergent machine from true type $t_1 \in P_1$ to type $t_m \in P_2$, but assume that monitoring will happen according to the type t_j[7]. We have

$$t_1 \cdot f_1(t_1, t_h) \leq t_j \cdot f_1(t_m, t_1).$$

Similarly to the cases above, we get that

$$\frac{t_1}{t_j} \cdot f_1(t_1, t_h) \leq \frac{(\alpha - 1)t_1}{t_m - t_1},$$

and that

$$\frac{t_h - \alpha t_1}{t_h - t_1} \leq \frac{t_j}{t_1} \frac{(\alpha - 1)t_1}{t_m - t_1} \iff$$

$$\alpha \geq \frac{t_h(t_m - t_1) + t_j(t_h - t_1)}{t_1(t_m - t_1) + t_j(t_h - t_1)},$$

which is again increasing in t_h, hence unbounded. The proof is now complete. □

In the following result we bypass the above inapproximability result using the notion of strict monitoring. As from above, strict monitoring implies that whenever a machine is allocated a task, but she has not been asked to pinpoint her exact type, she will be required to process the task for the maximum amount of time in her type domain that she has not yet excluded.

Definition 4.2 (Mechanism $\mathcal{M}_d$). $\mathcal{M}_d$ specifies an arbitrary order of the machines and considers all except the largest possible values of the domain in an increasing order. For $j = 1, \ldots d-1$ the mechanism asks the machines one after the other in the prespecified order if their type is t_j or not. Let machine i_j be the first machine who answers yes, i.e. that her type is t_j. The mechanism will then stop querying the machines and produce the following allocation: the task is allocated with probability $p_j = \frac{t_d}{t_d + t_j(n-1)}$ to machine i_j and with probability $\frac{1-p_j}{n-1} = \frac{t_j}{t_d + t_j(n-1)}$ to each other machine. If after the mechanism has asked all machines for all values at most t_{d-1} no machine has replied yes (which would imply that all machines have type t_d) then the task is allocated with probability $1/n$ to each machine.

Theorem 4.3. *$\mathcal{M}_d$ is an OSP with strict monitoring mechanism that achieves approximation ratio $\alpha < n$ for scheduling one task to n machines with finite domain.*

Proof. Consider domain $D = \{t_1, t_2, \ldots, t_d\}$ where $t_1 < t_2 < \ldots < t_d$, and mechanism $\mathcal{M}_d$ as defined above (Definition 4.2). We first prove that $\mathcal{M}_d$ is OSP with strict monitoring. The following inequality shows that a machine will not misreport a type t_j if her true type is higher, i.e. t_{j+k}, for some k.

$$t_{j+k} \cdot p_{j+k} \leq t_{j+k} \cdot p_j \iff$$

$$p_{j+k} \leq p_j \iff$$

$$\frac{t_d}{t_d + t_{j+k}(n-1)} \leq \frac{t_d}{t_d + t_j(n-1)} \iff$$

$$t_j \leq t_{j+k}$$

which is true. Next, we show that a machine with true type t_j will not misreport a higher type, i.e. t_{j+k}, for any k. We have that

$$t_j \cdot p_j \leq t_{j+k} \cdot p_{j+k} \iff$$

$$t_j \cdot \frac{t_d}{t_d + t_j(n-1)} \leq t_{j+k} \cdot \frac{t_d}{t_d + t_{j+k}(n-1)}$$

which is true since $\frac{d}{dx}\left(\frac{x t_d}{t_d + x(n-1)}\right) \geq 0$ and $t_j \leq t_{j+k}$.

It remains to consider the case where the machine with true type t_j unsuccessfully attempts to misreport a higher type, i.e. t_{j+k}, and prove that the OSP constraint holds in this case as well. In other words, let a machine i reply no when asked from the mechanism whether her type is t_j, and before she is asked to diverge at t_{j+k}, another machine i' replies yes to the mechanism (that she, i', has type $t_{j+k'}$ for $k' \leq k$). In this case, the machine i will be allocated the task with probability $\frac{1-p_{j+k'}}{n-1}$. Note that strict monitoring implies that machine i, if allocated the task, will be asked to execute it for time t_d since this is the highest type in the domain and it is compatible with the outcome (the machine has not excluded the possibility that her type is t_d yet). We need to prove that

$$t_j \cdot p_j \leq t_d \cdot \frac{1 - p_{j+k'}}{n-1}.$$

Indeed, notice that $p_1 \geq p_2 \geq \ldots \geq p_d \geq 1/n$. So, it suffices to prove that

$$t_j \cdot p_j \leq t_d \cdot \frac{1 - p_j}{n-1} \iff$$

$$p_j \leq \frac{t_d}{t_d + t_j(n-1)}$$

which is true by definition.

We now argue regarding the approximation ratio of $\mathcal{M}_d$. If the lowest type among the machines is t_j, the expected completion time under $\mathcal{M}_d$ is

$$t_j \frac{t_d}{t_d + t_j(n-1)} + t_d \frac{t_j(n-1)}{t_d + t_j(n-1)}$$

$$= \frac{t_d n}{t_d + t_j(n-1)} t_j,$$

[7] Monitoring according to the declaration t_m would be equivalent to classical monitoring, while monitoring according to the highest compatible type t_h would be equivalent to strict monitoring.

while the optimal approximation ratio is clearly t_j. We can conclude that the approximation ratio of $\mathcal{M}_d$ is at most

$$\alpha = \frac{H \cdot n}{H + L(n-1)} < n$$

where $t_1 = L$ and $t_d = H$ are the smallest and largest value in the domain, respectively. □

As noted above, strict monitoring is equivalent to monitoring in the case of two-value domains. Hence, Lemma 3.3 provides a corresponding lower bound for the case of two types in either monitoring assumption, which proves the tightness of Theorem 4.3 above. Of course this can be observed by the fact that Mechanism $\mathcal{M}_d$ (Definition 4.2) is a generalization of $\mathcal{M}_2$ (Definition 3.1).

We conclude this section by discussing how to use the inapproximability result in Lemma 4.1 to (essentially) characterize the implementation tree of OSP mechanisms with bounded approximation. A first easy observation is that "complete" extensive-form implementation trees, i.e., those where each leaf correctly determines the agents' declarations, lead to unbounded approximation as it would not be possible to define strict monitoring (but only monitoring). Our next lemma shows how to generalize this intuition by characterizing the type of queries that can be used to obtain a bounded approximation ratio.

Lemma 4.4. *Any mechanism that achieves a bounded approximation ratio must use queries like the ones used in Mechanism $\mathcal{M}_d$. In other words, each divergence has to partition the set of compatible types to a singleton containing the minimum compatible type, and all the remaining compatible types of the corresponding divergent machine.*

Proof. Consider a mechanism $\mathcal{M}$ whose implementation tree includes a divergence different than the one in the statement of the lemma. In particular, assume that the partition, P_1 containing the minimum type, t_1 is not a singleton and let t_h denote the largest type in P_1, $t_h > t_1$. We now distinguish between two cases:

First assume that the other partition, P_2, contains at least one element t_m such that $t_m < t_h$. Using similar arguments to the proof of Theorem 4.1, we can prove that we will get an unbounded approximation ratio by considering the OSP constraint regarding the deviation from the divergent machine i from true type $t_m \in P_2$ to type $t_h \in P_1$:

$$t_m \cdot f_i(t_m, t_h) \leq t_h \cdot f_1(t_h, t_1).$$

Otherwise, assume all elements of P_1 are smaller than all elements of P_2, but that the second smallest type $t_2 \in P_1$. Again, in this case, using similar arguments, we can prove that we will get an unbounded approximation ratio by considering the OSP constraint regarding the deviation from the divergent machine i from true type $t_2 \in P_1$ to the highest type $t_d \in P_2$:

$$t_2 \cdot f_i(t_2, t_d) \leq t_d \cdot f_1(t_d, t_1).$$

This concludes the proof. □

The results in this section give a complete picture of OSP with monitoring in the context of scheduling a task without money.

5 EXTENSION TO MANY TASKS

Our mechanisms can be applied independently to each task for the case of many tasks and guarantee an approximation of $n \cdot \frac{Hn}{H+L(n-1)} < n^2$ of the optimal makespan for the corresponding cases. Whether or not this is the best that can be achieved is an open question. We provide a preliminary result that gives a better bound for a very special case, almost matching the lower bound with monitoring provided in the previous section ($\frac{Hn}{H+L(n-1)} \approx n$ for carefully chosen type domain). Clearly, all inapproximability bounds presented in the previous sections hold for the case of many tasks as well, so proving a bounded approximation ratio for domains with at least 3 types would require stricter notions of monitoring.

Lemma 5.1. *There is an OSP mechanism with monitoring that achieves approximation ratio $\alpha \leq 2$ for scheduling two tasks to 2 machines when their type domain has two values, L and H.*

Proof. Consider the non-trivial case where $H > 2L$.[8] Consider mechanism $\mathcal{M}$ that picks an arbitrary machine, let that be machine 1, and only asks her to act. In particular, $\mathcal{M}$ first asks machine 1 to distinguish between types L and H for the first task, and then asks her the same about the second task. $\mathcal{M}$ allocates the tasks as follows according to the declarations of machine 1:

- If $t_{1,1} = t_{1,2} = L$, then $f_{1,1} = f_{1,2} = 1$, i.e. machine 1 gets both tasks,
- If $t_{1,1} = L$ and $t_{1,2} = H$, then $f_{1,1} = 1$, and $f_{1,2} = \frac{L}{H}$,
- If $t_{1,1} = H$ and $t_{1,2} = L$, then $f_{1,1} = \frac{L}{H}$, and $f_{1,2} = 1$,
- If $t_{1,1} = t_{1,2} = H$, then $f_{1,1} = f_{1,2} = \frac{L}{H}$.

Note that in each of the above cases, the probabilities yield the same total cost, $2L$ to machine 1.

We first argue that mechanism $\mathcal{M}$ is OSP. Indeed, we consider the following cases: if $t_{1,1} = L$ then if machine 1 acts truthfully, she will get at most cost $2L$, while by deviating to $t'_{1,1} = H$ she will get at least that, regardless of what choices she makes in the next step. If $t_{1,1} = H$ then the worst outcome yields cost $2L$, while by deviating to $t'_{1,1} = L$ machine 1 will get the first task with probability 1 for a cost H, which is higher by our original assumption. If $t_{1,1} = L$ and $t_{1,2} = L$ then deviating to $t'_{1,2} = H$ machine 1 will get cost $L + \frac{L}{H}H = 2L$, same as her cost for reporting truthfully. If $t_{1,1} = L$ and $t_{1,2} = H$ then deviating to $t'_{1,2} = L$ machine 1 will get both tasks with probability 1 for a cost $L + H > 2L$. If $t_{1,1} = H$ and $t_{1,2} = L$ then deviating to $t'_{1,2} = H$ will yield cost $2L$ to machine 1 (since by monitoring she will be expected to take time H to execute each task she is allocated) same as her cost for reporting truthfully. Finally, if $t_{1,1} = H$ and $t_{1,2} = H$ then by deviating to $t'_{1,2} = L$, machine 1 will be allocated each task with at least the same probability compared to reporting truthfully, hence her cost will be higher.

We now prove that mechanism $\mathcal{M}$ has approximation ratio 2. We need to argue about each case independently, and take into account the possible true types of machine 2 as well. We abuse notation and denote by $\mathcal{M}$ and $\mathcal{O}$, the expected makespan from mechanism $\mathcal{M}$, and the optimal expected makespan, respectively, in each of the cases, as the true types in each case will be clear from the context. In particular, if $t_{1,1} = t_{1,2} = L$, then $\mathcal{M} = 2L \leq 2\mathcal{O}$. If $t_{1,1} = L$ and

[8]Otherwise, if $H \leq 2L$, the mechanism that allocates task 1 to machine 1 and task 2 to machine 2 always has approximation ratio at most 2 trivially.

$t_{1,2} = H$, then we distinguish between two cases regarding type $t_{2,2}$:

- If $t_{2,2} = L$, then $\mathcal{O} = L$, while $\mathcal{M} = \frac{L}{H}(L+H) + \left(1 - \frac{L}{H}\right) L = 2L = 2\mathcal{O}$.
- If $t_{2,2} = H$, then $\mathcal{O} = H$ while $\mathcal{M} = \frac{L}{H}(L+H) + \left(1 - \frac{L}{H}\right) H = \frac{L^2}{H} + H \leq 2H = 2\mathcal{O}$.

If $t_{1,1} = H$ and $t_{1,2} = L$, we can distinguish cases regarding type $t_{2,1}$ and the analysis will be equivalent to the one above. Finally if $t_{1,1} = t_{1,2} = H$, then we distinguish between the following cases:

- If $t_{2,1} = L$ and $t_{2,2} = L$, then $\mathcal{O} = 2L$ and $\mathcal{M} = \frac{L^2}{H^2}2H + 2\left(1 - \frac{L}{H}\right)\frac{L}{H}H + \left(1 - \frac{L}{H}\right)^2 2L = 4L - \frac{4L^2}{H} + \frac{2L^3}{H^2} \leq 4L = 2\mathcal{O}$.
- If $t_{2,1} = H$ or $t_{2,2} = H$, then $\mathcal{O} \geq H$ and $\mathcal{M} \leq \frac{L^2}{H^2}2H + 2\left(1 - \frac{L}{H}\right)\frac{L}{H}H + \left(1 - \frac{L}{H}\right)^2 2H = 2H - 2L + \frac{2L^2}{H} \leq 2H = 2\mathcal{O}$.

The proof is now complete. □

Unfortunately, as the next lemma shows, the mechanism that is presented in the proof of the above result cannot be extended to achieve a bounded approximation ratio for a larger type domain.

Lemma 5.2. *A mechanism that only queries one machine cannot provide a bounded approximation ratio for type domains of size at least* 3.

Proof. Regarding the first statement, consider the case of 2 tasks, 2 machines and 3 type domains (L, M, H). Let machine 1 be the one that is asked to act and assume that she has type (M, M). For a bounded approximation c, in case machine 2 has type (L, L), machine 1 should be allocated task 1 with probability $p_{1,1}$ and task 2 with probability $p_{1,2}$, such that $2Mp_{1,1}p_{1,2} + Mp_{1,1}(1 - p_{1,2}) + Mp_{1,2}(1 - p_{1,1}) = M(p_{1,1} + p_{1,2}) \leq c2L$. However, in the case where machine 2 has type (H, H), then machine 1 should be allocated task 1 with probability $p_{1,1}$ and task 2 with probability $p_{1,2}$, such that $Hp_{1,1}(1 - p_{1,2}) + Hp_{1,2}(1 - p_{1,1}) + 2H(1 - p_{1,1})(1 - p_{1,2}) = H(2 - p_{1,1} - p_{1,1}) \leq c2M$. The above would imply that $2 - \frac{c2M}{H} \leq p_{1,1} + p_{1,2} \leq \frac{c2L}{M}$, which is not feasible for very high values of H. □

6 CONCLUSIONS

We advance the state of the art in the design of OSP mechanisms, in the specific and well-studied domain of machine scheduling [2, 14, 16, 19]. As proved by Li [2017], OSP mechanisms are the only ones that are truthful for agents that have limited contingent reasoning skills. Differently from much of the recent literature on OSP [3, 5, 9], which shows that very little can be done with this solution concept, we here give the strong positive message that the approximation guarantee of OSP mechanisms can be basically as good as that of truthful mechanisms. In a sense, our work extends to the realm of mechanisms without money the main finding of [8]; whilst they prove that monitoring allows to turn *any* algorithm into an OSP mechanisms with payments (via direct revelation), we here show that a careful implementation tree and the right notion of monitoring can be used to match the approximation guarantee of truthful mechanisms. Whenever our notion of strict monitoring cannot be implemented in the setting at hand, our results show that the designer needs to look at alternative means to guarantee even a bounded approximation of the optimum; notably, even harsher forms of punishments have been shown to be essentially ineffective for OSP mechanisms without money [9].

The main technical open problem left by our work is to establish the correct approximation guarantee of OSP mechanisms for scheduling unrelated machines (i.e. the case of many tasks). This would appear to require significant advances in the understanding of OSP for multi-dimensional agents.

REFERENCES

[1] Mohammad Akbarpour and Shengwu Li. 2018. Credible Mechanisms. In *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018.* 371.

[2] Aaron Archer and Éva Tardos. 2001. Truthful Mechanisms for One-Parameter Agents. In *FOCS 2001.* 482–491.

[3] Itai Ashlagi and Yannai A Gonczarowski. 2018. Stable matching mechanisms are not obviously strategy-proof. *Journal of Economic Theory* (2018).

[4] Lawrence M Ausubel. 2004. An efficient ascending-bid auction for multiple objects. *American Economic Review* 94, 5 (2004), 1452–1475.

[5] Sophie Bade and Yannai A. Gonczarowski. 2017. Gibbard-Satterthwaite Success Stories and Obvious Strategyproofness. In *EC 2017.* 565.

[6] D. Ferraioli, A. Meier, P. Penna, and C. Ventre. 2018. On the approximation guarantee of obviously strategyproof mechanisms. *CoRR* abs/1805.04190 (2018).

[7] Diodato Ferraioli, Paolo Serafino, and Carmine Ventre. 2016. What to Verify for Optimal Truthful Mechanisms without Money. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016.* 68–76.

[8] Diodato Ferraioli and Carmine Ventre. 2017. Obvious Strategyproofness Needs Monitoring for Good Approximations. In *AAAI 2017.* 516–522.

[9] Diodato Ferraioli and Carmine Ventre. 2018. Probabilistic Verification for Obviously Strategyproof Mechanisms. In *IJCAI 2018.* Available at 'https://arxiv.org/abs/1804.10512'.

[10] D. Fotakis, P. Krysta, and C. Ventre. 2014. Combinatorial auctions without money. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014.* 1029–1036.

[11] D. Fotakis, P. Krysta, and C. Ventre. 2018. Equal-Cost Mechanism Design with Monitoring. (2018). Submitted.

[12] Dimitris Fotakis and Christos Tzamos. 2010. Winner-Imposing Strategyproof Mechanisms for Multiple Facility Location Games. In *WINE.* 234–245.

[13] Dimitris Fotakis and Christos Tzamos. 2013. Winner-imposing strategyproof mechanisms for multiple Facility Location games. *Theor. Comput. Sci.* 472 (2013), 90–103. https://doi.org/10.1016/j.tcs.2012.11.036

[14] Y. Giannakopoulos, E. Koutsoupias, and M. Kyropoulou. 2016. The Anarchy of Scheduling Without Money. In *SAGT.* 302–314.

[15] Allan Gibbard. 1973. Manipulation of Voting Schemes: A General Result. *Econometrica* 41, 4 (1973), 587–601.

[16] E. Koutsoupias. 2011. Scheduling without Payments. In *Proc. of SAGT (LNCS)*, Vol. 6982. 143–153.

[17] Annamária Kovács, Ulrich Meyer, and Carmine Ventre. 2015. Mechanisms with Monitoring for Truthful RAM Allocation. In *WINE (Lecture Notes in Computer Science)*, Vol. 9470. Springer, 398–412.

[18] Shengwu Li. 2017. Obviously strategy-proof mechanisms. *American Economic Review* 107, 11 (2017), 3257–87. Available at 'http://ssrn.com/abstract=2560028'.

[19] Noam Nisan and Amir Ronen. 2001. Algorithmic Mechanism Design. *Games and Economic Behavior* 35 (2001), 166–196.

[20] Kobbi Nissim, Rann Smorodinsky, and Moshe Tennenholtz. 2012. Approximately optimal mechanism design via differential privacy. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012.* 203–213.

[21] Paolo Penna and Carmine Ventre. 2014. Optimal collusion-resistant mechanisms with verification. *Games and Economic Behavior* 86 (2014), 491 – 509.

[22] Ariel D. Procaccia and Moshe Tennenholtz. 2013. Approximate Mechanism Design without Money. *ACM Trans. Economics and Comput.* 1, 4 (2013), 18:1–18:26.

[23] Mark Allen Satterthwaite. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory* 10, 2 (1975), 187 – 217.

[24] Paolo Serafino, Angelina Vidali, and Carmine Ventre. 2018. Truthfulness on a Budget: Trading Money for Approximation through Monitoring. (2018). Submitted.