Obviously Strategyproof Mechanisms without Money for Scheduling

Obviously Strategyproof Mechanisms without Money for Scheduling*

Maria Kyropoulou† Carmine Ventre‡

January 13, 2025

Abstract

We consider the scheduling problem when no payments are allowed and the machines are bound by their declarations. We are interested in a notion of incentive compatibility, stronger than the (standard) strategyproofness, termed obvious strategyproofness (OSP) and explore its possibilities and its limitations. OSP formalizes the concept of strategyproofness 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 providing information about her true type is at least as good as the best possible outcome after misreporting information about her type. Under the *weaker* constraint of strategyproofness, Koutsoupias [2014] proves a tight approximation ratio of $\frac{n+1}{2}$ for the makespan for one task, under the monitoring paradigm. 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 strategyproofness to OSP, as long as the mechanism designer can implement a particular notion of monitoring. To achieve this, we introduce the notion of max-monitoring and prove that weaker monitoring frameworks do not suffice, 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

*A preliminary version of this article has appeared in AAMAS'19 [25]. The current version significantly differs from the old one as it contains full proofs of all our results and additional discussion on the results and their possible interpretations. Moreover, an entirely new proof is provided for one of our main results (Theorem 1); even though the result is the same, the new proof reveals a lot of insight about the structure of the implementation tree of mechanisms that achieve a bounded approximation ratio.

†University of Essex, Computer Science and Electronic Engineering, Colchester, UK .

‡King's College London, Department of Informatics, London, UK .

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; moreover, they do not care about the impact that this has on the *makespan*, i.e., the completion time of the schedule. We are interested in designing allocation protocols that do not use payments and that return a solution whose makespan is not far from the non-strategic, centrally enforced optimum makespan. The study of the strategic variant of the scheduling problem was initiated by Nisan and Ronen [28] in their seminal paper on algorithmic mechanism design. Since then, scheduling has been considered one of the fundamental and most motivational problems in the area of algorithmic mechanism design as it captures the need to take into account the incentives of the owners controlling the computers in major computing platforms. As Nisan and Ronen state, "with the emergence of the Internet as *the* platform of computation, this assumption (that the participating computers will act as instructed) can no longer be taken for granted". We highlight how in scheduling problems, machines and tasks are abstractions that can model any situation in which some workers have to complete some jobs for the designer.

A primary designer goal that has been extensively studied is that of *strategyproofness*, which informally implies that a player maximizes 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 the machine 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. Very recently, the long standing conjecture of Nisan and Ronen [28] for the general case of n machines and many tasks has been resolved, confirming that that no strategyproof mechanism can achieve an approximation ration better n [6]. But, even for the case of one task, the strategyproofness 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 an ascending-price implementation of the second-price auction [4, 21]. Li [26] provided an explanation for this phenomenon by introducing the notion of Obviously Strategy-Proof (OSP) mechanisms.

Informally, OSP implies that at any step of the execution of the algorithm the divergent agent would prefer even the worst possible outcome corresponding to her true type, to even the best possible outcome corresponding to any alternative type. As mentioned above, different implementations of the same auction can have different properties with respect to OSP. To see this, consider some point

in the execution of a descending auction and focus on a particular agent. If the processing time of the agent is lower than the tentative compensation, then the worst possible outcome from acting truthfully and staying in the auction (corresponding to a utility of 0 for not getting the task or getting the task for a compensation equivalent to the processing time) is at least as good as the best possible outcome of dropping out now. Because a similar constraint holds for any agent and any point in the execution of the algorithm, descending auctions are OSP. On the other hand, if we consider the implementation of a sealed-bid second price auction, getting utility 0 after acting truthfully is not necessarily at least as good as the best possible outcome if the agent reports a type lower than her true one; indeed, in this latter case, the best possible outcome might be that the agent is allocated the task for a compensation that is exceeding her true type. So, sealed-bid second price auctions are not OSP, even thought they are strategy-proof (indeed, given a fixed type profile of the other agents, one's utility is always at least as high when reporting truthfully). Technically, the main difference between OSP and strategy-proofness is that OSP constraints do not consider a fixed type profile of the remaining agents and, instead, take a worst-case approach.

Li [26] characterized the OSP solution concept as the correct one for people with a certain form of limited contingent reasoning skills, thus formalizing the concept of strategyproofness for agents with a certain kind of bounded rationality. Perhaps counter-intuitively to the required detail and the complexity of formally defining the OSP property, it is evident that (computational considerations aside) a participant in an OSP mechanism will not think twice before acting truthfully as there is nothing better she can hope to achieve. Hence, in this very precise sense, OSP mechanisms capture the very important notion of strategic simplicity. From a mechanism design angle, OSP is also linked to mechanisms with partial commitment power [26]. Particularly in the context of machine scheduling, we believe that both strategic simplicity and partial designer commitment power are important aspects to consider as they are key to attract more processing agents that trust the allocation. The monitoring paradigm also allows the designer more flexibility by expanding the range of potential mechanisms.

In many cases, the use of payments might be considered unethical, illegal (e.g., organ donations) or even just impractical (see, e.g., [31]). Yet, strategyproofness remains a very desirable property that can help the mechanism designer know what kind of behavior to expect from the agents and therefore to correctly optimize the target function. For this reason, researchers have started turning their attention to possible ways of achieving strategyproofness without the use of payments. Unfortunately, achieving strategyproofness is not always compatible with maintaining a good objective value in the absence of payments [20, 33]. So, in order to achieve a reasonable (even simply bounded) approximation of the optimal solution, the use of additional machinery might be imperative. Koutsoupias [22] provides a positive result in the context of scheduling without payments using the plausible assumption of *monitoring*, which implies that the machines are bound by their declarations. In other words, the mech-

anism *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 declaration. 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., [29, 18]). A framework that is also very much related to monitoring is the notion of verification where agents are heavily punished when caught lying [16, 13] or, in presence of money, denied their payments [30]; these penalties are, in some cases, in addition to being monitored [28]. Variants of monitoring have been studied extensively in the literature (under slightly different names) [19, 13, 34, 17, 24, 18, 29, 15].

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 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 strategyproofness 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., [26, 10]). Indeed, the revelation principle does not hold for obvious strategyproofness [26]. Implementation trees model 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 mechanism 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 that the mechanism will enforce if the execution path reaches that leaf.

[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 [19].

Extending the definition of monitoring to 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 to this question, we introduce the notion of *max*-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; max-monitoring is the only form of monitoring that can overcome impossibility results.[2] In particular, this can also be shown to imply that direct-revelation mechanisms like the one by Koutsoupias are not OSP. Indeed, no implementation of the mechanism by Koutsoupias would allow for the max-monitoring paradigm as the mechanism's outcomes need complete knowledge of the type profile.[3] 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 is subject to a presumptive income taxation – i.e., the tax authority *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.[4]

[2]It is interesting to compare this result with the approach, used in, e.g., [5], to ask an agent to move one last time by revealing her full preferences once the outcome for that agent has been determined. Indeed, our work omits this revelation step and achieves a bounded approximation ratio (under the right monitoring framework).

[3]Observe that its approximation guarantee contradicts our lower bound for specific finite domains.

[4]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. As a form of punishment our notion appears well motivated; if someone is truthful, then even if they do not disclose fully, they would not be punished. If someone is not truthful (and this is discovered), then the worst would be assumed regarding their intended future manipulation and they would be punished accordingly. It seems reasonable to design systems that choose not to listen or believe any information submitted by a person once their credibility has been compromised.

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 [22] for fully rational agents, as a benchmark. As noted above, for the case of scheduling a single task Koutsoupias [22] 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 strategyproofness to OSP, as long as the mechanism designer can implement a particular(ly harsh) 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 ratio of the maximum over the minimum type therein. Therefore, the price to pay to turn strategyproofness into OSP is less than a factor 2 loss in the approximation guarantee, if we are allowed to use a stricter variant of monitoring.

Another contribution of our work is the notion of max-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 max-monitoring is necessary and sufficient to obtain bounded approximation ratios, in the sense that any other (weaker) notion of monitoring[5] leads to a strong lower bound on the approximation guarantee of OSP mechanisms.

Our work uses in fact two somewhat novel techniques to lower bound the approximation guarantee of OSP mechanisms without money, which adapt the ideas in [10] to the setting without money. For the former, we identify a property that implementation trees need to satisfy (regardless of the strategyproofness concept) in order to guarantee a "good" approximation, and only 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, mimics the typical structure of a lower bound to the approximation guarantee of a truthful mechanism, see, e.g., [23], and combines approximation and OSP constraints. Crucially, the incentive-compatibility constraints are defined upon some structural properties of the implementation tree of mechanisms with bounded approximation. We, in fact, characterize the queries that need to be done (including in what order) to marry bounded approximation and OSP. With this approach, we reach the conclusion that one needs a careful definition of monitoring to obtain bounded approximations (Theorem 4.4).

We finally briefly consider the extension to many tasks, the so-called schedul-

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

ing unrelated machines problem. We observe that we can independently allocate each task to the machines preserving OSP, and obtain an $\alpha \cdot n$ approximation. We leave as an open problem to understand whether this is the best possible as opposed to strategyproofness where it is known that n-approximation is possible [19].

Gap between OSP and strategyproofness It is perhaps unsurprising that a *very strong* notion of monitoring is necessary to close the gap between strategyproofness and OSP. As discussed above and witnessed by its few real-world implementations, max-monitoring is a particularly harsh way to punish misbehaving agents and consequently makes the life of the designer much easier. Nevertheless, there is a silver lining to our work, namely demonstrating the existence of a monitoring paradigm (however harsh) that makes OSP feasible with a bounded loss in the approximability compared to plain strategyproofness. This is motivated by the known negative results – discussed below – on the relative power of OSP versus strategyproofness, even when the designer uses some form of control on misbehaving agents (cf., e.g., [14]). Overall, our work provides a clean characterization for the incentive compatibility of boundedly rational agents for the case of a single task.

Related work In addition to the literature on truthful mechanisms for scheduling discussed above, our work contributes to the growing body of work on OSP mechanisms. After its definition in [26], the authors of [2] prove that no mechanism computing stable matchings can be OSP. Some classical results for truthful mechanism design are reconsidered for OSP in [5]. A number of papers look at understanding or relaxing the notion [27, 35, 8] while classic settings without money are studied in [32]. OSP mechanisms with money (and some form of monitoring) are studied in [15] for scheduling related machines and facility location, where some lower bounds on the approximation guarantee of these mechanisms are proved. A general technique based on an extension of cycle monotonicity, and tight approximation bounds for scheduling related machines, are then given in [10]. A follow-up paper uses the cycle monotonicity technique to characterize OSP mechanisms for set system problems [9]. The full versions of these two papers are combined in [11]. A public project problem is studied in [14] for OSP mechanisms with probabilistic verification and no money; in essence, the main result says that either the fines to punish lying agents or the number of agents to verify are very high. More recent results on OSP mechanisms for binary allocation problems and combinatorial auctions are presented in [12, 7], respectively.

2 Preliminaries

In the unrelated machines scheduling problem, we are given m tasks that need to be allocated to n machines. Each machine i is controlled by a selfish agent who has job-dependent types $t_{i,j}$ that correspond to the time machine i needs

to execute job j. $t_{i,j} \in D_i$ where D_i denotes the domain of agent i's; when the domain of all agents is the same we denote it by D. Slightly abusing notation, we often use the term machine to refer to the agent controlling it. Types are private knowledge of the machines, and machines will be asked to declare them to a mechanism, who will then decide the allocation of tasks to machines accordingly. This, possibly randomized, allocation is referred to as a schedule, or the outcome of the mechanism. Machines wish to minimise their own (expected) running time in the resulting schedule, which is computed as the sum over all tasks, of the time the corresponding machine will need to execute the corresponding task times the probability of the task being allocated to that machine. We seek to design mechanisms that probabilistically allocate the jobs to minimize the *makespan*, defined as the completion time of the entire schedule, or equivalently the expected maximum running time of any machine. We focus on mechanisms that are OSP, in that they incentivise truthtelling even if the machines are of a certain form of bounded rationality. 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). The domain of f is a collection of subsets, each of which corresponds to a set of possible types for an agent; f 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.[6] Each internal node u of $\mathcal{T}$ is labelled with a player $S(u)$, called the divergent agent at u. Each edge (v, u) is labeled with a collection of sets of types $\{D_i(u)\}_{i=1,\dots,n}$ that are corresponding to the current domain of each agent, as explained below. 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 outgoing edges, and is equivalent to declaring that her true type appears in the corresponding label. The edge labels satisfy the following: Let w denote the root of the tree. The sets corresponding to agent $i \neq S(w)$ in the labels of the outgoing edges of w are exactly D_i, while the sets corresponding to $S(w)$ form a partition of the domain of $S(w)$, i.e., each type in $D_{S(w)}$ appears in exactly one such set. Similarly for an internal node u other than the root, the sets corresponding to agent $i \neq S(u)$ in the labels of the outgoing edges of u are equal to $D_i(u)$, while the sets corresponding to $S(u)$ form a partition of $D_{S(u)}(u)$. We say that $\{D_i(u)\}_{i=1,\dots,n}$ consists of sets of types that are compatible with the history of u. Applying f to the set of bid profiles corresponding to the collection $\{D_i(u)\}_{i=1,\dots,n}$ of a leaf node u defines 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 a bid profile, 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 $(b_{i,j})$) for task j. For simplicity, we use $f(\mathbf{b})$ to denote the outcome of f for the leaf of $\mathcal{T}$

[6] 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 [10] show how this is equivalent to Li's original definition in [26] and the ones used in subsequent work.

to whose associated set of profiles $\mathbf{b}$ 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 $\{H\}$ and $\{M\}$.

We now define OSP and OSP with (max-)monitoring; 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 (even if the other agents' future actions are assumed to depend in an arbitrary way on the choice of the said agent): 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 two compatible profiles $(t, \mathbf{b}_{-i})$ and $(b, \mathbf{b}'_{-i})$ diverge at u if $i = S(u)$ and t and b belong to the 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 at that node.

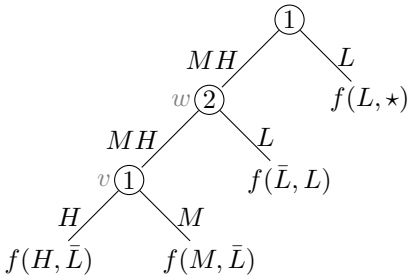

Figure 1: An implementation tree for scheduling a task to two machines with domain $\{L, M, H\}$; $\bar{L}$ denotes the fact that the type of a machine is in $\{M, H\}$ in the corresponding leaf, while $\star$ denotes the fact that the machine can have any type in the domain D in the corresponding leaf.

For every agent i and types $t, b \in D_i$, we let $U^i_{t,b}$ denote the set of vertices u in the implementation tree, 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 may 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)**.** An extensive-form mechanism $\mathcal{M} = (f, \mathcal{T})$ is OSP for scheduling a single task if for all i, $t, b \in D_i$, $u^i_{t,b} \in U^i_{t,b}$, and $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}(u^i_{t,b})$, it holds that $f_i(t, \mathbf{b}_{-i}) \leq f_i(b, \mathbf{b}'_{-i})$.

Definition 2.2 (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, $t, b \in D_i$, $u^i_{t,b} \in U^i_{t,b}$, 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 of OSP with monitoring 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 $\max\{t, b\} \cdot f_i(b, \mathbf{b}'_{-i})$. Note that the definition captures the case where a machine reports a type smaller than her true type, i.e., machines may claim that they can execute the task faster than what they actually can. In that case, if allocated the task, the corresponding machine will be allowed to finish the execution, but her cost will be equal to her true type.

However, there is not an immediate analogue of declaration in extensive-form mechanisms (different from a final revelation step as in, e.g., [5] – see Footnote 2). 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? Max-monitoring implies that we assume the highest compatible type. Although this is a very strict assumption, we show later (Theorem 4.4) that *any weaker monitoring notion* does not suffice, as it would lead to an unbounded approximation of the optimal makespan[7]. 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 2.3 (OSP with max-monitoring)**.** A mechanism $(f, \mathcal{T})$ in extensive-form is OSP with max-monitoring for scheduling a single task if for all i, $t, b \in D_i$, $u^i_{t,b} \in U^i_{t,b}$ and $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}(u^i_{t,b})$, it holds that

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

When max-monitoring is applied, the makespan of the schedule is computed as

$$\mathbb{E}_f \left[\max_{i=1,\dots n} f_i(\mathbf{b}) \cdot \max\{t_i, \max D_i(\ell_{\mathbf{b}})\} \right].$$

[7]The approximation ratio is technically bounded from below by a function of the types in the domain, which can be made arbitrarily high for arbitrarily large values of the largest type.

We note that our definition implicitly assumes that if we allocate a machine the task, then we can identify whether the information declared about her type was truthful (e.g., if machines cannot slow down their true speed), and not monitor the machine in that case (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 max-monitoring to dishonest agents. (This is similar to the Italian presumptive taxation system discussed above.) Our results, positive and negative, continue to hold even in the case where such a type-identification mechanism is not available, and even if the truthful agents are expected to (possibly unfairly) exert effort equal to the highest possible type compatible with the history.

Randomization, monitoring and OSP In our model, an outcome is defined as a lottery over allocations, so OSP effectively requires that each agent weakly prefers the worst-case (randomized) allocation that can be the output of the mechanism after declaring her true type, to the best-case (randomized) allocation that can be the output of the mechanism after declaring any other value. This could be viewed as a sort of OSP in expectation concept. Our definition is in fact an immediate application to scheduling of the definitions introduced by Li in his seminal paper [26]. Randomness in Li's definition can come in two shapes: chance (internal) nodes and lotteries in the leaves. The draw from the random coin flips of internal nodes are used in the definition to determine best/worst outcomes. It has been noted in the literature how this choice translates "universally truthful mechanisms" to OSP (see, for example, footnote 9 in [2]). (Incidentally, a definition of OSP in expectation which also considers the effects of internal chance nodes on bounded rationality is given in [14].) In the context of the randomness of lotteries in the nodes, Li's definition allows to look at lotteries as outcomes (cf. footnote 15 in [26]). We focus on this weaker "in expectation" desiderata since, even for truthfulness, we cannot achieve any positive result that is robust ex post to the randomness of the mechanism, as observed in [22]. OSP with monitoring has been considered already in [15]; however their definition fails to capture the intricacies of extensive-form mechanisms and mainly focuses on direct revelation.

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 shortest 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.) The probability p needs to be small enough to satisfy a

critical OSP constraint, while the larger it is, the better the approximation ratio becomes. So, in order to achieve the best possible approximation ratio under OSP, the mechanism defines p to be as large as possible without violating any OSP constraint. We note that Proposition 3.2 is a special case of Theorem 4.7 and will be proved later in its general form.

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 that 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.

Proposition 3.2. *$\mathcal{M}_2$ is an OSP with monitoring[8] mechanism that for every type domain with two values, L and H, achieves approximation ratio $\frac{Hn}{H+(n-1)L} < n$ for scheduling one task to n machines.*

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 (Inequalities (1) and (2a) below). First note that for the specific allocation probabilities it holds that

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

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

$$\begin{aligned} L \cdot f_i(H^{i-1}, L, \star) \leq H \cdot f_i(H^{i-1}, H, \star), \quad &\iff \quad (1) \\ L \cdot p \leq H \min\left\{\frac{1-p}{n-1}, \frac{1}{n}\right\} \quad &\iff \\ L\frac{H}{H+(n-1)L} \leq H\frac{L}{H+(n-1)L}, \end{aligned}$$

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

$$\begin{aligned} H \cdot f_i(H^{i-1}, H, \star) \leq H \cdot f_i(H^{i-1}, L, \star) \quad &\iff \quad (2a) \\ \max\left\{\frac{1-p}{n-1}, \frac{1}{n}\right\} \leq p \quad &\iff \\ \frac{1}{n} \leq \frac{H}{H+(n-1)L}, \end{aligned}$$

[8]Note that for two-value domains, monitoring and max-monitoring coincide.

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 one machine have type L, the expected completion time of $\mathcal{M}_2$ is at most $pL+\frac{1-p}{n-1}H(n-1) = \frac{Hn}{H+(n-1)L}\cdot L \leq nL$, 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 Proposition 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 that has 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 $\mathcal{M}$ that achieves approximation α for values of the domain that satisfy $H > (2n-1)L$. Assume for a contradiction that $\mathcal{M}$ is OSP (with monitoring) and

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

Then, $\mathcal{M}$ has to satisfy the following property: 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 can be made arbitrarily larger than the optimum (L) for sufficiently large values of H.

Consider the path of the implementation tree that would be followed if the first $n-1$ machines that were queried declared according to type H. Note that by the property above, such a path exists in our implementation tree. Denote by o the outcome that will arise if the remaining machine has true type L (it could be that the remaining machine is asked to diverge in the final step, or the outcome is immediately computed after querying the first $n-1$ machines). In outcome o, the machine with type L should be allocated the task with probability $p_l \geq \frac{H-\alpha L}{H-L} = p$, in order to satisfy the approximation ratio guarantee of α. To see this, note that if $p_l < p$ then with the remaining probability a machine with type H would be allocated the task, for an expected completion time of $p_l L + (1-p_l)H = H - p_l(H-L) > H - p(H-L) = \alpha L$. So, we can assume that $p_l \geq p$.

In outcome o, at least one machine with type H 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:

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

This implies that if $\mathcal{M}$ is OSP, then its approximation ratio α satisfies $\alpha \geq \frac{Hn}{L(n-1)+H}$, which is a contradiction.

□

4 Unrestricted domains

In this section we consider the general case of scheduling with large finite domains. Although monitoring is enough to achieve a bounded approximation ratio for strategyproofness [22], we here prove that it is not enough to get a bounded approximation ratio for OSP, whilst max-monitoring is sufficient (and necessary).

We first present a technical lemma that will be useful in our analysis.

Lemma 4.1. *Let v^+ denote a vector of values such that the minimum value in the vector is equal to v. If for some $a < b < c$ in the type domain it holds that*

$$b \cdot f_1(b, c^+) \leq c \cdot f_1(c, a^+),$$

then the approximation ratio of the corresponding mechanism is unbounded if a, b, and c can be scaled up arbitrarily.

Proof. An approximation guarantee of α for an instance $(t_i, \mathbf{t}_{-i})$, where $t_i < \min_{j\neq i} t_j$ implies

$$\begin{aligned} t_i f_i(t_i, \mathbf{t}_{-i}) + \min_j t_j(1 - f_i(t_i, \mathbf{t}_{-i})) \leq \alpha t_i \iff & \\ f_i(t_i, \mathbf{t}_{-i}) \geq \frac{\min_{j\neq i} t_j - \alpha t_i}{\min_{j\neq i} t_j - t_i}. & \end{aligned}$$

So, for some vector $\mathbf{t}$ such that $t_i = b$ and $\min_{j\neq i} t_j = c > b$, we have

$$f_i(b, c^+) \geq \frac{c - \alpha b}{c - b}. \tag{4}$$

Similarly, the approximation guarantee for an instance $(t_i, \mathbf{t}_{-i})$, with $t_i > \min_j t_j$ implies that

$$t_i f_i(t_i, \mathbf{t}_{-i}) + \min_j t_j (1 - f_i(t_i, \mathbf{t}_{-i})) \leq \alpha \min_j t_j$$
$$f_i(t_i, \mathbf{t}_{-i}) \leq \frac{(\alpha - 1) \min_j t_j}{t_i - \min_j t_j}.$$

So, for some vector $\mathbf{t}$ such that $t_i = c$ and $\min_j t_j = a < c$, we have

$$f_i(c, a^+) \leq \frac{(\alpha - 1)a}{c - a}. \tag{5}$$

Recall that by assumption it holds that $b \cdot f_1(b, c^+) \leq c \cdot f_1(c, a^+)$. So, from Inequality (5) for $i = 1$, we get that

$$\frac{b}{c} \cdot f_1(b, c^+) \leq \frac{(\alpha - 1)a}{c - a} \iff$$
$$f_1(b, c^+) \leq \frac{c}{b} \frac{(\alpha - 1)a}{c - a},$$

while from this and Inequality (4) we get that

$$\frac{c - \alpha b}{c - b} \leq \frac{c}{b} \frac{(\alpha - 1)a}{c - a} \iff$$
$$\alpha \geq \frac{bc(c - a) + ac(c - b)}{b^2(c - a) + ac(c - b)}.$$

The limit of the RHS of the above inequality as c grows large is $\frac{a+b}{a}$, which can be made arbitrarily large, for sufficiently large b. The proof is complete.

□

The next couple of lemmas essentially characterize the implementation trees of OSP mechanisms with bounded approximation guarantees by giving some structural properties on the queries to perform (Lemma 4.2) and their order (Lemma 4.3). Let us call a *min query*, a divergence that partitions the set of compatible types (at that point) of the corresponding divergent agent into a singleton containing her minimum compatible type, and the set of all the remaining compatible types. [9]

Lemma 4.2. *Any OSP mechanism that achieves a bounded approximation ratio must use a min-query every time the current domains of the agents satisfy the following:*

If the current domain of the divergent agent is $\mathcal{D}$', then

[9] The notion of min query has already appeared as central in several OSP mechanisms, such as ascending auctions [26] and the general form of OSP voting mechanisms [5]. The reasons behind this are discussed in [12, 7] for binary allocation and combinatorial auction domains, respectively, in the context of OSP mechanisms with money.

- *the current domain of at least one other agent is also $\mathcal{D}$', and*
- *the current domain of any other agent includes all elements of $\mathcal{D}$' except possibly the smallest one.*

Proof. It suffices to consider mechanisms that perform at least one query, as otherwise it can be easily seen that the approximation ratio is unbounded. Consider a mechanism $\mathcal{M}$ and a divergence of $\mathcal{M}$ that is not a min-query. Let the divergent agent be agent 1, the current domain of agent 1 be $D' = \{t_1, t_2, \ldots, t_d\}$ where $t_1 < t_2 < \ldots < t_d$ and $d \geq 3$ (with two types, the divergence would be done with a min-query), and the current domain of any other agent is either D' or $D' \setminus \{t_1\}$, as per the statement of the lemma.

Assume that machine 1 is asked to distinguish between $P_1, P_2, \ldots, P_k$ that form a partition of D' and let $t_1 \in P_1$. Since, by assumption the partition is not t_1 and $D' \setminus t_i$, then there must exist $t_1 < t_i < t_j$, such that t_i and t_j are in different partitions. We consider the OSP constraint regarding the deviation of agent 1 from t_i to t_j:

$$t_i \cdot f_1(t_i, t_j^\star) \leq t_j \cdot f_1(t_j, t_1^+),$$

where $t^\star$ denotes the vector where all the remaining agents have type t and t^+, as above, denotes a vector of values where the minimum value is equal to t. Note that by assumption, there exists at least one other agent with type t_1 in her current domain. By Lemma 4.1, we conclude that $\mathcal{M}$ has unbounded approximation ratio. □

Lemma 4.3. *The implementation tree of any OSP mechanism that achieves a bounded approximation ratio satisfies the following property: The path that is compatible with all agents having the maximum type consists of consecutive subpaths in which each agent diverges exactly once with a min-query.*[10]

Proof. Let $D = \{t_1, \ldots, t_d\}$ where $t_1 < t_2 < \ldots < t_d$ denote the original domain, and consider a mechanism $\mathcal{M}$ and the path S in its implementation tree that is compatible with all agents having the maximum type, t_d. We will prove the statement by induction. Assume for a contradiction that in S an agent (call her 1) is asked to diverge for the second time before all other agents diverge, and consider the first time that happens starting from the root of the tree. In particular, S contains nodes u and u' such that agent 1 diverges at both u and u', each other agent diverges at most once (agent 1 diverges exactly once at u) before u' in S, while at least one agent does not diverge in the same interval. Denote by O the set of agents that have diverged (once) in S before u' (including agent 1), and by Z the set of agents that have not diverged (Z is non-empty). By Lemma 4.2, it follows that all the divergences of agents in O were min-queries. Also, if we denote by D'_i the current domain of agent i at u',

[10] Figure 1 shows a small example of such a tree.

it holds that $D'_i = D \setminus \{t_1\}$ for the agents $i \in O$ while $D'_i = D$ for the agents $i \in Z$.

Now assume that the partition used by the mechanism at u' partitions D'_1 in $P_1, \ldots, P_k$. It holds that $t_2 = \min D'_1$ and assume without loss of generality that $t_2 \in P_1$. Also, let t_m be the maximum type in D'_1 that does not belong to P_1; note that $t_m > t_2$. Consider the following OSP constraint regarding the divergence of agent 1 from type t_2 to type t_m

$$t_2 \cdot f_1(t_2, t^\star_m) \leq t_m \cdot f_1(t_m, t^Z_1, t^O_2), \tag{7}$$

where t^Z denotes a vector in which all the agents in Z have type t (similarly for t^O), and, as above, $t^\star$ denotes the vector where all the remaining players play t. Note that by construction the types used are still compatible for the corresponding agents at u'; recall that we are focusing on the history where all the agents who diverged have a type compatible with t_d. By Lemma 4.1, we conclude that $\mathcal{M}$ has unbounded approximation ratio. So, we can conclude that the first n nodes of S perform min-queries to different agents, for $\mathcal{M}$ to have bounded approximation ratio.

Now assume that the first $n \cdot k$ nodes in S satisfy that every consecutive set of n nodes performs a min-query to each agent exactly once. This means that the current domain $\bar{D}$ of each agent exactly before the $(n \cdot k + 1)$-th divergence is the same. By repeating the above arguments after substituting D with $\bar{D}$, we prove that the following n nodes of S will again perform n min-queries, each to a different agent, as desired.

□

We are now ready to show that max-monitoring is necessary.

Theorem 4.4. *There is no OSP mechanism with bounded approximation ratio for scheduling one task when the type domain has at least three values whose ratios can be arbitrarily large, even with monitoring when the type we use to monitor an agent is not the highest compatible type. In other words, notions of monitoring weaker than max-monitoring do not suffice to achieve a bounded approximation ratio.*

Proof. Consider an OSP mechanism that achieves approximation ratio α. We will prove that α cannot be bounded when the mechanism is not using max-monitoring. We will focus on the path S of the tree that is compatible with all agents having the maximum type. By Lemma 4.3, we know that S consists of consecutive subpaths in which each agent diverges exactly once with a min-query. Assume that the mechanism is monitoring with a value t_m that is less than the maximum, i.e., $t_1 < t_m < t_d$.

Let $D = \{t_1, \ldots, t_d\}$ where $t_1 < t_2 < \ldots < t_d$ denote the original domain. Consider the i-th divergence of S, and let agent i diverge there, for $i = 1, \ldots, n$. By Lemma 4.3 we know that a min-query is performed, so we will consider the OSP constraint of agent i that involves the type vectors $(t_d^{i-1}, t_1, t_d^\star)$ and $\mathbf{t} = (t_d^{i-1}, t_m, t_1, t_d^\star)$, where the first $i-1$ agents and the last $n-(i+1)$ agents

have type t_d in both, and agents i and $i+1$ have type t_1 and t_d respectively in the former vector, and t_m and t_1 respectively in the latter vector. (We let t^i denote a vector of i t's and, as in the proofs above, we let $t^\star$ denote a vector of all t's whose dimension is clear from the context.) If the true type of agent i is t_1, her cost after acting according to t_1 should always be at most her cost when she acts according to a misreport monitored with t_m. We get the following OSP constraint for agent i, where t^+ denotes a vector of values whose minimum value is t:

$$\begin{aligned} t_1 \cdot f_i(t_1, t_d^\star) \leq t_m \cdot f_i(\mathbf{t}) = t_m \cdot f_i(t_m, t_1^+) &\iff \\ f_i(t_m, t_1^+) \geq \frac{t_1}{t_m} f_i(t_1, t_d^\star) &\implies \\ f_i(\mathbf{t}) \geq \frac{t_1}{t_m} \cdot \frac{t_d - \alpha t_1}{t_d - t_1}, & \end{aligned} \tag{8}$$

where we have used Inequality (4) in the proof of Lemma 4.1 that holds for any mechanism with approximation ratio α. We immediately get that

$$f_{i+1}(\mathbf{t}) \leq 1 - \frac{t_1}{t_m} \cdot \frac{t_d - \alpha t_1}{t_d - t_1}. \tag{9}$$

We will now consider the OSP constraint of agent $i+1$ (at the $(i+1)$-th diversion of S, by assumption) regarding her deviation from type t_m to t_1, which involves vectors $(t_d^i, t_m, t_d^\star)$ and $\mathbf{t} = (t_d^{i-1}, t_m, t_1, t_d^\star)$, as above. Note that the type of agent i is not the same in the two profiles. It holds that

$$\begin{aligned} t_m \cdot f_{i+1}(t_m, t_d^\star) \leq t_m \cdot f_{i+1}(\mathbf{t}) &\implies \\ \frac{t_d - \alpha t_m}{t_d - t_m} \leq 1 - \frac{t_1}{t_m} \cdot \frac{t_d - \alpha t_1}{t_d - t_1} &\iff \\ \alpha \geq \frac{t_1 t_d^2 + t_m(t_m - t_1)t_d - t_1 t_m^2}{(t_m^2 + t_1^2)t_d - t_1 t_m (t_m + t_1)}, & \end{aligned}$$

where we have used the Inequality (4) in the proof of Lemma 4.1 that holds for any mechanism with approximation ratio α, and Inequality (9). It suffices to observe that the limit of the RHS of the last inequality as t_d grows large is infinity. Interestingly, the result already holds true for the simple case of two agents with domains of three values.

□

The above theorem can also be used to show that when the implementation tree is complete no bounded approximation guarantee can be achieved. Informally, this holds because when the implementation tree is complete, the domain of any leaf node contains exactly one type, so monitoring can only happen with this type and max-monitoring is equivalent to plain monitoring. A formal proof is presented in the context of Proposition 4.5 below.

Proposition 4.5. *Assume that the domain has size at least three. Then any OSP mechanism with max-monitoring whose implementation tree is complete, i.e., those where each leaf correctly determines the agents' reported types, has an unbounded approximation guarantee.*

Proof. The proof follows from the arguments used in the proof of Theorem 4.4, after we observe that, we can only define plain monitoring (as opposed to max-monitoring) when the implementation tree is complete.

Indeed, consider domain $D = \{t_1, t_2, \ldots, t_d\}$ where $t_1 < t_2 < \ldots < t_d$, $d \geq 3$, and let $t_m = t_2$ (any other type in the domain that is different to t_1 and t_d would work as well). We focus on the path S of the tree that is compatible with all agents having the maximum type and consider the i-th divergence of S; let agent i diverge there, for $i = 1, \ldots, n$. By Lemma 4.3 we know that a min-query is performed, so we will consider the OSP constraint of agent i that involves the type vectors $(t_d^{i-1}, t_1, t_d^\star)$ and $\mathbf{t} = (t_d^{i-1}, t_m, t_1, t_d^\star)$ (the type of agent i is t_1 in the former vector and t_m in the latter), exactly like we did in the proof of Theorem 4.4. Since the implementation tree is complete by assumption, monitoring is indeed happening with type t_m, so, we reach inequalities (8) and (9) here as well. We can then similarly conclude that the approximation guarantee of the mechanism is unbounded. □

We note here that the mechanism in [22] needs, in general, a complete implementation tree as the definition of the output probabilities uses all the types in the strategy profile. As a result, it is not possible to have some OSP implementation of that mechanism achieving an $(n+1)/2$ ratio.

In the following result we bypass the inapproximability result of Theorem 4.4 using the notion of max-monitoring. As from above, max-monitoring implies that whenever a machine is allocated a task, but the machine 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 (assuming that the machine is found to have acted untruthfully). We define a strict generalisation of Mechanism $\mathcal{M}_2$, which we call $\mathcal{M}_d$, where the probabilities are chosen as in $\mathcal{M}_2$, only with t_d in lieu of H and the current t_j in lieu of L.

Definition 4.6 (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 that 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 4.7. *$\mathcal{M}_d$ is an OSP with max-monitoring mechanism that achieves approximation ratio $\frac{t_d \cdot n}{t_d + t_1(n-1)} < n$ for scheduling one task to n machines with finite domain, where t_1 and t_d are the smallest and largest value in the domain, respectively.*

Proof. The proof follows along the same lines as the proof of Proposition 3.2. Extending that mechanism to the case of multiple types introduces some additional opportunities for misreporting to the agents/machines which have to be considered as well. 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 4.6). We first prove that $\mathcal{M}_d$ is OSP with max-monitoring. We will seemingly ignore the case where the task is allocated to each machine with probability $1/n$ (if no machine ever replies yes to the mechanism), but we note that in fact this is covered in the analysis below by recalling that $p_d = 1/n$. We start by showing that a machine will not misreport a type t_j if her true type is higher, i.e., t_{j+k}, for some k. Reporting according to her true type would result in a cost equal to $t_{j+k} \cdot p_{j+k}$. However, when the machine is considering diverging she needs to consider the possibility that another machine would answer "yes" to the mechanism for having type $t_{j'}$, before she would get a chance to report t_{j+k}, that is $t_j \leq t_{j'} \leq t_{j+k}$; in that case the machine would have a cost equal to $t_{j+k} \cdot \frac{1-p_{j'}}{n-1} = t_{j+k} \cdot \frac{t_{j'}}{t_d + t_{j'}(n-1)} \leq t_{j+k} \cdot \frac{t_{j+k}}{t_d + t_{j+k}(n-1)} \leq t_{j+k} \cdot \frac{t_d}{t_d + t_{j+k}(n-1)} = t_{j+k} \cdot p_{j+k}$. So, it suffices to show that $t_{j+k} \cdot p_{j+k}$ is at most equal to the cost the machine would get by misreporting t_j when her true type is t_{j+k}. Indeed, it holds that

$$\begin{aligned} t_{j+k} \cdot p_{j+k} \leq t_{j+k} \cdot p_j &\iff \\ p_{j+k} \leq p_j &\iff \\ \frac{t_d}{t_d + t_{j+k}(n-1)} \leq \frac{t_d}{t_d + t_j(n-1)} &\iff \\ t_j \leq t_{j+k}, & \end{aligned}$$

which is true. Next, we show that a machine with true type t_j will not try to misreport a higher type, i.e., t_{j+k}, for any k. We will first look at the case where the machine successfully misreports t_{j+k} (that is later in the run she reports t_{j+k}) and then look at the case where the mechanism halts before she gets a chance to report t_{j+k}. In the former case, we have that

$$\begin{aligned} t_j \cdot p_j \leq t_{j+k} \cdot p_{j+k} &\iff \\ t_j \cdot \frac{t_d}{t_d + t_j(n-1)} \leq t_{j+k} \cdot \frac{t_d}{t_d + t_{j+k}(n-1)}, & \end{aligned}$$

which is true since $\frac{d}{dx}\left(\frac{x t_d}{t_d + x(n-1)}\right) = \frac{t_d^2}{(t_d + x(n-1))^2} \geq 0$, 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 max-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 and in fact, this is the reason p_j was defined as it was.

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 at most

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

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

$$\frac{t_d \cdot n}{t_d + t_1(n-1)} < n,$$

where t_1 and t_d are the smallest and largest value in the domain, respectively. The proof is complete.

□

As noted above, max-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 4.7 above. Of course this can be observed by the fact that Mechanism $\mathcal{M}_d$ (Definition 4.6) is a generalization of $\mathcal{M}_2$ (Definition 3.1).

Overall, 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

In this section we consider the general case of scheduling many tasks to machines that have task-dependent types belonging to 2-value domains, and explore how our results for the single task case might extend to this case. 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. Here we are considering a *parallel* composition rather than a *sequential* composition (which is known not to be OSP in general , e.g. this is true for sequential ascending price auctions with additive bidders [5])[11]. This would model agents who would "forget" the history regarding previous jobs when facing the mechanism for the allocation of a new job.[12] 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.

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

Proof. Consider the non-trivial case where $H > 2L$.[13] 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.[14] $\mathcal{M}$ 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 $f_{1,2} = 1$,

[11] A sequential composition would imply that there is a single mechanism/implementation tree, where the execution considers the allocation of tasks sequentially, and an outcome corresponds to the allocation of all tasks. In such a mechanism, each time a machine acts, she reasons about the overall allocation. In contrast, a parallel composition considers the allocation of each task by a separate mechanism/implementation tree whose outcome corresponds to the allocation of the corresponding task. In each of these mechanisms, each time a machine acts, she reasons about the allocation of the respective task alone. The overall schedule is computed by combining the individual outcomes.

[12] The extent to which this notion of OSP with parallel composition is close to the bounded rationality modeled by the original notion and the sequential composition of OSP mechanisms is an open problem.

[13] 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.

[14] This is an indicative mechanism. In fact the order of questions does not make a difference as long as $\mathcal{M}$ decides the outcome after machine 1 has fully disclosed her type.

- 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. To see that mechanism $\mathcal{M}$ is OSP it suffices to observe that the utility from truthtelling is always $2L$, while the utility from misreporting is computed as the maximum between the utility had the misreport been truthful and the utility from misreporting had there been no monitoring, which can only 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 $t_{1,2} = H$, 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 $\mathcal{M} = \frac{L}{H}(L+H) + \left(1 - \frac{L}{H}\right) H = \frac{L^2}{H} + H \leq 2H = 2\mathcal{O}$.

The case where $t_{1,1} = H$ and $t_{1,2} = L$ is dropped due to symmetry; the analysis follows by identical arguments if we distinguish cases regarding type $t_{2,1}$. 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 $\mathcal{O}$=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.

Proposition 5.2. *A mechanism that only queries one machine cannot provide a bounded approximation ratio for type domains with at least three values whose ratios can be arbitrarily large.*

Proof. 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 the makespan does not exceed $c2L$. In particular,

it needs to hold that $2Mp_{1,1}p_{1,2} + Mp_{1,1}(1-p_{1,2}) + Mp_{1,2}(1-p_{1,1}) \leq c2L$ or equivalently that $M(p_{1,1}+p_{1,2}) \leq c2L$. 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 the makespan does not exceed $c2M$. In particular it needs to hold that $Hp_{1,1}(1-p_{1,2}) + Hp_{1,2}(1-p_{1,1}) + 2H(1-p_{1,1})(1-p_{1,2}) \leq c2M$ or equivalently that $H(2-p_{1,1}-p_{1,1}) \leq c2M$. 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 $L \ll M \ll H$.

□

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 [28, 1, 22, 19]. As proved by Li [26], OSP mechanisms are the only ones that are identifiable as truthful by agents that have a certain form of limited contingent reasoning skills. Differently from much of the literature on OSP [2, 14], 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 (as long as the mechanism designer can implement a particular notion of monitoring). In a sense, our work extends to the realm of mechanisms without money the main finding of [15]; whilst they prove that monitoring allows to turn any algorithm into an OSP mechanism 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 max-monitoring cannot be implemented in the setting at hand, our results show that the designer needs to look at alternative means (i.e., other paradigms that restrict the lying power of the agents, such as, type-based verification [3] or monitoring that does not necessarily use the types in the domain) 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 [14].

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. The results in [7] might represent a good starting point to at least study special cases of unrelated machines.

References

[1] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *FOCS*, pages 482–491, 2001.

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

[3] Vincenzo Auletta, Paolo Penna, Giuseppe Persiano, and Carmine Ventre. Alternatives to truthfulness are hard to recognize. *Autonomous Agents and Multi-Agent Systems*, 22(1):200–216, 2011.

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

[5] Sophie Bade and Yannai A. Gonczarowski. Gibbard-satterthwaite success stories and obvious strategyproofness. In *EC*, page 565, 2017.

[6] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. A proof of the nisan-ronen conjecture. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 672?685, 2023.

[7] Bart de Keijzer, Maria Kyropoulou, and Carmine Ventre. Obviously strategyproof single-minded combinatorial auctions. In *ICALP*, 2020.

[8] D. Ferraioli and C. Ventre. Obvious strategyproofness, bounded rationality and approximation. *Theory Comput Syst*, 66:696 – 720, 2022.

[9] Diodato Ferraioli, Adrian Meier, Paolo Penna, and Carmine Ventre. Automated optimal OSP mechanisms for set systems: The case of small domains. In *WINE*, 2019.

[10] Diodato Ferraioli, Adrian Meier, Paolo Penna, and Carmine Ventre. Obviously strategyproof mechanisms for machine scheduling. In *ESA*, pages 46:1–46:15, 2019.

[11] Diodato Ferraioli, Adrian Meier, Paolo Penna, and Carmine Ventre. New constructions of obviously strategyproof mechanisms. *Mathematics of Operations Research*, 48(1):332–362, 2022.

[12] Diodato Ferraioli, Paolo Penna, and Carmine Ventre. Two-way greedy: Algorithms for imperfect rationality. In *Proceedings of WINE*, 2021.

[13] Diodato Ferraioli, Paolo Serafino, and Carmine Ventre. What to verify for optimal truthful mechanisms without money. In *AAMAS 2016*, pages 68–76, 2016.

[14] Diodato Ferraioli and Carmine Ventre. Probabilistic verification for obviously strategyproof mechanisms. In *Proceedings of IJCAI*, 2018.

[15] Diodato Ferraioli and Carmine Ventre. Approximation guarantee of OSP mechanisms: The case of machine scheduling and facility location. *Algorithmica*, 83(2):695–725, 2021.

[16] D. Fotakis, P. Krysta, and C. Ventre. Combinatorial auctions without money. *Algorithmica*, 77:756–785, 2017.

[17] Dimitris Fotakis, Piotr Krysta, and Carmine Ventre. Efficient truthful scheduling and resource allocation through monitoring. In *AAAI*, pages 5423–5431, 2021.

[18] Dimitris Fotakis and Christos Tzamos. Winner-imposing strategyproof mechanisms for multiple facility location games. *Theor. Comput. Sci.*, 472:90–103, 2013.

[19] Y. Giannakopoulos, E. Koutsoupias, and M. Kyropoulou. The anarchy of scheduling without money. *Theoretical Computer Science*, 778:19–32, 2019.

[20] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

[21] John H. Kagel, Ronald M. Harstad, and Dan Levin. Information impact and allocation rules in auctions with affiliated private values: A laboratory study. *Econometrica*, 55(6):1275–1304, 1987.

[22] Elias Koutsoupias. Scheduling without payments. *Theory of Computing Systems*, 54(3):375–387, Apr 2014.

[23] Elias Koutsoupias and Angelina Vidali. A lower bound of $1+\varphi$ for truthful scheduling mechanisms. *Algorithmica*, 66(1):211–223, 2013.

[24] Annamária Kovács, Ulrich Meyer, and Carmine Ventre. Mechanisms with monitoring for truthful RAM allocation. In *WINE*, pages 398–412, 2015.

[25] Maria Kyropoulou and Carmine Ventre. Obviously strategyproof mechanisms without money for scheduling. In *Proceedings of AAMAS*, pages 1574–1581, 2019.

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

[27] Andrew Mackenzie. A revelation principle for obviously strategy-proof implementation. *Games and Economic Behavior*, 124:512–533, 2020.

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

[29] Kobbi Nissim, Rann Smorodinsky, and Moshe Tennenholtz. Approximately optimal mechanism design via differential privacy. In *Proceedings of ITCS*, pages 203–213, 2012.

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

[31] Ariel D. Procaccia and Moshe Tennenholtz. Approximate mechanism design without money. *ACM Trans. Economics and Comput.*, 1(4):18:1–18:26, 2013.

[32] Marek Pycia and Peter Troyan. Obvious dominance and random priority. In *Proceedings of EC*, page 1, 2019.

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

[34] Paolo Serafino, Carmine Ventre, and Angelina Vidali. Truthfulness on a budget: trading money for approximation through monitoring. *Auton. Agents Multi Agent Syst.*, 34(1):5, 2020.

[35] Luyao Zhang and Dan Levin. Partition obvious preference and mistrust in mechanism design: Theory and experiment. *SSRN*, 2017.