Obviously Strategyproof Mechanisms for Machine Scheduling

Diodato Ferraioli
Università degli Studi di Salerno, Italy
dferraioli@unisa.it

Adrian Meier
ETH Zurich, Switzerland
meiera@student.ethz.ch

Paolo Penna
ETH Zurich, Switzerland
paolo.penna@inf.ethz.ch

Carmine Ventre
King's College London, UK
carmine.ventre@kcl.ac.uk

Abstract

Catering to the incentives of people with limited rationality is a challenging research direction that requires novel paradigms to design mechanisms and approximation algorithms. Obviously strategyproof (OSP) mechanisms have recently emerged as the concept of interest to this research agenda. However, the majority of the literature in the area has either highlighted the shortcomings of OSP or focused on the "right" definition rather than on the construction of these mechanisms.

We here give the first set of *tight* results on the approximation guarantee of OSP mechanisms for scheduling related machines. By extending the well-known cycle monotonicity technique, we are able to concentrate on the algorithmic component of OSP mechanisms and provide some novel paradigms for their design.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Bounded Rationality, Extensive-form Mechanisms, Approximate Mechanism Design

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.46

Related Version A full version of the paper is available at `https://arxiv.org/abs/1805.04190`.

Funding *Diodato Ferraioli*: This author is partially supported by GNCS-INdAM and by the Italian MIUR PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Markets".

1 Introduction

Mechanism design has been a very active research area that aims to develop algorithms that align the objectives of the designer (e.g., optimality of the solution) with the incentives of self-interested agents (e.g., maximize their own utility). One of the main obstacles to its application in real settings is the assumption of full rationality. Where theory predicts that people should not strategize, lab experiments show that they do (to their own disadvantage): this is, for example, the case for Vickrey's renown second-price auction; proved to be strategyproof and yet bidders lie when submitting sealed bids. Interestingly, however, lies are less frequent when the very same mechanism is implemented via an ascending auction [18].

A vague explanation of this phenomenon is that, from the point of view of a bidder, the strategyproofness of an ascending price auction is *easier to grasp* than the strategyproofness of the second-price sealed bid auction [3]. The key difference is the way these auctions are *implemented*:

- In the second-price sealed-bid auction (direct-revelation implementation), each bidder submits her own bid *once* (either her true valuation or a different value). This mechanism is *strategyproof* meaning that truth-telling is a dominant strategy: *for every report of the other bidders*, the utility when truth-telling is not worse than the utility when bidding untruthfully.
- In the ascending price auction (extensive-form implementation), each bidder is repeatedly offered some price which she can accept (stay in the auction) or reject (leave the auction). In this auction, momentarily *accepting a good price* guarantees a non-negative utility, while *rejecting a good price* or *accepting a bad price* yield non-positive utility. Here good price refers to the private valuation of the bidder and, intuitively, truth-telling in this auction means accepting prices as long as they are not above the true valuation.

Intuitively speaking, in the second type of auction, it is *obvious* for a bidder to decide her strategy, because the utility for the *worst scenario* when truth-telling is at least as good as that of the *best scenario* when cheating. The recent definition of *obviously strategyproof (OSP)* mechanisms [23] formalizes this argument: ascending auctions are OSP mechanisms, while sealed-bid auctions are not. Interestingly, [23] proves that a mechanism is OSP *if and only if* truth-telling is dominant even for bidders who lack contingent reasoning skills.

As being OSP is *stronger* than being strategyproof, it is natural to ask if this has an impact on what can be done by such mechanisms. For instance, the so-called *deferred-acceptance* (DA) auctions [26] are OSP (as they essentially are ascending price auctions), but unfortunately their performance (approximation guarantee) for several optimization problems is quite poor compared to what strategyproof mechanisms can do [9]. Whether this is an inherent limitation of OSP mechanisms or just of this technique is not clear.

One of the reasons behind this open question might be the absence of a general technique for designing OSP mechanisms and the lack of an algorithmic understanding of OSP mechanisms. Specifically, it is well known that strategyproofness is equivalent to certain *monotonicity* conditions of the *algorithm* used by the mechanism for computing the solution (be it an allocation of goods or a path in a network with self-interested agents). Therefore, one can essentially focus on the algorithmic part and study questions regarding the approximation and the complexity. The same type of questions seems much more challenging for OSP mechanisms, as such characterizations are not known. Recent work in the area [5, 27, 24] aims at simplifying the notion of OSP, by looking at versions of the revelation principle for OSP mechanisms. This, for example, allows to think, without loss of generality, at deterministic (rather than randomized) extensive-form mechanisms where each agent moves sequentially (rather than concurrently).

The goal of this work is build the foundations to reason about OSP algorithmically. In particular, we advance the state of the art by providing an algorithmic characterization of OSP mechanisms. Among others, our results show why deferred acceptance auctions [26] – essentially the only known technique to design OSP mechanisms with money – do not fully capture the power of a "generic" OSP mechanism, as the latter may exploit some aspects of the implementation (i.e., extensive-form game) in a crucial way.

Our Contribution. To give an algorithmic characterization of OSP mechanisms, we extend the well known cycle-monotonicity (CMON) technique. This approach allows to abstract the truthfulness of an algorithm in terms of non-negative weight cycles on suitably defined graphs. We show that non-negative weight cycles continue to characterize OSP when the graph of

interest is carefully defined. Our main conceptual contribution is a way to accommodate the OSP constraints, which *depend on the particular extensive-form implementation* of the mechanism, in the machinery of CMON, which is designed to focus on the algorithmic output of mechanism. Interestingly, our technique shows the interplay between algorithms (which solution to return) and how this is implemented as an extensive-form game (what we call the *implementation tree*). Roughly speaking, our characterization says which algorithms can be used for *any* choice of the implementation tree. The ability to choose between different implementation trees is what gives extra power to the designer: for example, the construction of OSP mechanisms based on DA auctions [26] uses always *the same* fixed tree for all problems and instances. Though this yields a simple algorithmic condition, it can be wasteful in terms of optimality (approximation guarantee) as we show herein. In fact, for our results, we will use CMON *two ways* to characterize both algorithmic properties (having fixed an implementation tree) and implementation properties (having fixed the approximation guarantee we want to achieve).

Armed with the OSP CMON technique, we are able to give the first *tight* bounds on the approximation ratio of OSP mechanisms. In particular, we consider the problem of scheduling n related machines (for identical jobs). While the lower bound holds regardless of the size of the domain, the mechanisms that we provide are shown to be OSP only for two- and three-value domains, as we prove that these are the only cases in which non-negative two-cycles are necessary and sufficient.

We show that the optimum for machine scheduling can be implemented OSP-ly when the agents' domains have size two. We prove that given a "balanced" optimum (i.e., a greedy allocation of jobs to machines) we can always find an implementation tree for which OSP is guaranteed. The mechanism directly asks the queried agents to reveal their type; given that the domain only contains two values, this is basically a descending/ascending auction. For domains of size three, instead, we give a lower bound of $\sqrt{n}$ and an essentially tight upper bound of $\lceil\sqrt{n}\rceil$. Interestingly, the latter is proved with two different mechanisms – one assuming more than $\lceil\sqrt{n}\rceil^2$ number of jobs and the second under the hypothesis that there are less than that. On the technical level, these results are shown by using our approach of CMON two ways. We prove that any better than $\sqrt{n}$-approximate OSP mechanism must have the following structure: for a number of rounds, the mechanism must (i) separate, in its implementation tree, largest and second largest value in the domain; (ii) assign nothing to agents who have maximum value in the domain. The former property restricts the family of implementation trees we can use, whilst the latter restricts the algorithmic output. Our lower bound shows that there is nothing in this intersection. Our matching upper bounds need to find *both* implementation tree and algorithm satisfying OSP and approximation guarantee.

▶ **Main Theorem** (informal). *The tight approximation guarantee of OSP mechanisms that can be guaranteed over all three-valued domains is $\sqrt{n}$. The OSP mechanisms use a descending auction (to find the $n - \lceil\sqrt{n}\rceil$ slowest machines) followed by an ascending auction (to find the fastest machine(s)).*

While the general idea of the implementation is that of a descending auction followed by an ascending auction independently of the number of jobs, we need to tailor the design of the mechanisms (namely, their ascending phase) according to the number of jobs to achieve OSP and desired approximation simultaneously. This proves two important points. On one hand, the design of OSP mechanisms is challenging yet interesting as one needs to carefully balance algorithms and their implementation. On the other hand, it proves why fixing the implementation, as in DA auctions, might be the wrong choice. It is indeed straightforward to extend and adapt our analysis in order to prove that any ascending and descending (thus including DA) auction has an approximation of n.

We remark that our mechanisms are, to the best of our knowledge, the first examples of OSP mechanisms with money that do not follow a clock or a posted price auction format (other mechanisms that do not follow these formats have been proposed only for setting without money, namely matching and voting [23, 2, 5, 27]). One of the main messages of our work is exactly that it is possible to combine ascending and descending phases for the implementation trees of algorithms with good approximation guarantees and obtain OSP mechanisms.

Related Works. The notion of OSP mechanism has been introduced recently by [23] and has received a lot of attention in the community. As mentioned above, the class of deferred-acceptance auctions [26] yields OSP mechanisms since every such auction can be implemented as a (suitable) ascending price auction. One of the main advantages of DA auctions is that the construction boils down to the problem of defining a suitable *scoring function* for the bidders [26]. [9] studied the approximability of DA auctions for several optimization problems, and showed that in some cases DA auctions must have an approximation guarantee significantly worse than the best strategyproof mechanism; [9, 19] provide a number of positive results where DA auctions are instead optimal. [15] studies also DA auction for the job scheduling problem: they design an approximate mechanism, but for a different objective function, namely the weighted completion time.

Several works have focused on understanding better the notion of OSP mechanism, and studying settings without money, namely matching and voting. In particular [2, 5, 24] mainly attempt to simplify the notion, whilst [27, 31, 14] define, among other results, stronger and weaker versions of OSP. A couple of recent papers related to ours are [12], where among other settings the authors consider OSP mechanisms with money for machine scheduling, and [21], where this problem is studied in the setting without money. In particular, the lower bound for machine scheduling in [12] is *constant* and uses a particular definition of payments, while here we prove a $\sqrt{n}$ lower bound that follows from the CMON characterization of OSP; their upper bound instead uses monitoring, a model wherein agents pay their reported costs whenever they overbid. Monitoring is also used in [21] to prove an encouraging bound for OSP mechanisms without money and a single task; the bound (asymptotically) matches the performances of strategyproof mechanisms.

Research in algorithmic mechanism design [17, 7] has suggested to focus on "simple" mechanisms to deal with bounded rationality. For example, posted-price mechanisms received huge attention very recently and have been applied to many different settings [4, 11, 1, 10, 8]. In these mechanisms one's own bid is immaterial for the price paid to get some goods of interest – this should immediately suggest that trying to play the mechanism is worthless no matter the cognitive abilities of the agents. However, posted price mechanisms do not fully capture the concept of simple mechanisms: e.g., ascending price auctions are not posted price mechanisms and still turn out to be "simple" to play and understand.

CMON is a widely used technique in mechanism design that dates back to [28] – a general treatment is given in [25, 16]. This method has been used quite extensively to prove strategyproofness of mechanisms in the classical setting, cf., e.g., [6, 22] and when some form of verification can be adopted, see [30, 20]. Particularly relevant for our work is the research which shows that in order to establish strategyproofness it is sufficient to study cycles of length 2 as in [29].

2 Preliminaries

A mechanism design setting is defined by a set of n *selfish agents* and a set of allowed *outcomes* $\mathcal{S}$. Each agent i has a *type* $t_i \in D_i$, where D_i is called the *domain* of i. The type t_i is usually assumed to be *private knowledge* of agent i. We will let $t_i(X) \in \mathbb{R}$ denote the *cost* of agent i with type t_i for the outcome $X \in \mathcal{S}$. In our application, we will assume that costs are non-negative; however, our framework and characterization hold in general no matter the sign.

A *mechanism* is a process for selecting an outcome $X \in \mathcal{S}$. To this aim, the mechanism interacts with agents. Specifically, agent i is observed to take *actions* (e.g., saying yes/no) that may depend on her presumed type $b_i \in D_i$ (e.g., saying yes could "signal" that the presumed type has some properties that b_i enjoys). We say that agent i takes *actions compatible with (or according to)* b_i to stress this. We highlight that the presumed type b_i can be different from the real type t_i.

For a mechanism $\mathcal{M}$, we let $\mathcal{M}(\mathbf{b})$ denote the outcome returned by the mechanism when agents take actions according to their presumed types $\mathbf{b} = (b_1, \ldots, b_n)$. In our context, this outcome is given by a pair $(f, \mathbf{p})$, where $f = f(\mathbf{b})$ (termed *social choice function* or, simply, algorithm) maps the actions taken by the agents according to $\mathbf{b}$ (i.e., each agent i takes actions compatible with b_i) to a feasible solution in $\mathcal{S}$, and $\mathbf{p} = \mathbf{p}(\mathbf{b}) = (p_1(\mathbf{b}), \ldots, p_n(\mathbf{b})) \in \mathbb{R}^n$ maps the actions taken by the agents according to $\mathbf{b}$ to *payments* from the mechanism to the agents.

Each selfish agent i is equipped with a *utility function* $u_i \colon D_i \times \mathcal{S} \to \mathbb{R}$. For $t_i \in D_i$ and for an outcome $X \in \mathcal{S}$ returned by a mechanism $\mathcal{M}$, $u_i(t_i, X)$ is the utility that agent i has for outcome X when her type is t_i. We define utility as a quasi-linear combination of payments and costs, i.e., $u_i(t_i, \mathcal{M}(b_i, \mathbf{b}_{-i})) = p_i(b_i, \mathbf{b}_{-i}) - t_i(f(b_i, \mathbf{b}_{-i}))$.

A mechanism $\mathcal{M}$ is *strategy-proof* if, for each i, the utility of player i is maximized by playing the extensive-form implementation of $\mathcal{M}$ according to her true type t_i. That is, in a strategy-proof mechanism the actions taken according to the true type are dominant for each agent.

For our application, we will be focusing on *single-parameter* settings, that is, the case in which the private information of each bidder i is a single real number t_i and $t_i(X)$ can be expressed as $t_i w_i(X)$ for some publicly known function w_i. To simplify the notation, we will write $t_i f_i(\mathbf{b})$ when we want to express the cost of a single-parameter agent i of type t_i for the output of social choice function f on input the actions corresponding to a bid vector $\mathbf{b}$.

Obvious Strategyproofness. We now formally define the concept of obviously strategy-proof deterministic mechanisms. This concept has been introduced in [23]. However, our definition is built on the more accessible ones given by [2] and [12]. As shown in [5, 24], our definition is equivalent to Li's.[1]

Let us first formally model how a mechanism works. An *extensive-form mechanism* $\mathcal{M}$ is defined by a directed tree $\mathcal{T} = (V, E)$, called the *implementation tree*, such that:

- Every leaf ℓ of the tree is labeled with a possible outcome $X(\ell) \in \mathcal{S}$ of the mechanism;
- Every internal vertex $u \in V$ is labeled with an agent $S(u) \in [n]$;

[1] More in detail, our definition of implementation tree is equivalent to the concept of round-table mechanism in [24]. Consequently, our definition of OSP is equivalent to the concept of SP-implementation through a round table mechanism, that is proved to be equivalent to the original definition of OSP for deterministic mechanisms. For a discussion of randomization for OSP mechanisms, we kindly refer the reader to [2, 13].

- Every edge $e = (u, v) \in E$ is labeled with a subset $T(e) \subseteq D = \times_i D_i$ of type profiles such that:
  - The subsets of profiles that label the edges outgoing from the same vertex u are disjoint, i.e., for every triple of vertices u, v, v' such that $(u, v) \in E$ and $(u, v') \in E$, we have that $T(u, v) \cap T(u, v') = \emptyset$;
  - The union of the subsets of profiles that label the edges outgoing from a non-root vertex u is equal to the subset of profiles that label the edge going in u, i.e., $\bigcup_{v:\ (u,v)\in E} T(u, v) = T(\phi(u), u)$, where $\phi(u)$ is the parent of u in $\mathcal{T}$;
  - The union of the subsets of profiles that label the edges outgoing from the root vertex r is equal to the set of all profiles, i.e., $\bigcup_{v:\ (r,v)\in E} T(r, v) = D$;
  - For every u, v such that $(u, v) \in E$, where u is not the root, and for every two profiles $\mathbf{b}, \mathbf{b}' \in T(\phi(u), u)$ such that $b_i = b'_i$, $i = S(u)$, if $\mathbf{b}$ belongs to $T(u, v)$, then $\mathbf{b}'$ must belong to $T(u, v)$ also.

Roughly speaking, the tree represents the steps of the execution of the mechanism. As long as the current visited vertex u is not a leaf, the mechanism interacts with the agent $S(u)$. Different edges outgoing from vertex u are used for modeling the different actions that the agent $S(u)$ can take during this interaction with the mechanism. In particular, each possible action is assigned to an edge outgoing from u. As suggested above, the action that agent i takes may depend on her presumed type $b_i \in D_i$. That is, different presumed types may correspond to taking different actions, and thus to different edges. The label $T(e)$ on edge $e = (u, v)$ then lists the type profiles that enable the agent $S(u)$ to take those actions that have been assigned to e. In other words, when the agent takes the actions assigned to edge e, then the mechanism (and the other agents) can infer that the type profile must be contained in $T(e)$. The constraints on the edges' label can be then explained as follows: first we can safely assume that different actions must correspond to different type profiles (indeed, if two different actions are enabled by the same profiles we can consider them as a single action); second, we can safely assume that each action must correspond to at least one type profile that has not been excluded yet by actions taken before node u was visited (otherwise, we could have excluded this type profile earlier); third, we have that the action taken by agent $S(u)$ can only inform about her types and not about the type of the remaining agents. The execution ends when we reach a leaf ℓ of the tree. In this case, the mechanism returns the outcome that labels ℓ.

Observe that, according to the definition above, for every profile $\mathbf{b}$ there is only one leaf $\ell = \ell(\mathbf{b})$ such that $\mathbf{b}$ belongs to $T(\phi(\ell), \ell)$. Similarly, to each leaf ℓ there is at least a profile $\mathbf{b}$ that belongs to $T(\phi(\ell), \ell)$. For this reason we say that $\mathcal{M}(\mathbf{b}) = X(\ell)$. Moreover, for every type profile $\mathbf{b}$ and every node $u \in V$, we say that $\mathbf{b}$ is *compatible* with u if $\mathbf{b} \in T(\phi(u), u)$. Finally, two profiles $\mathbf{b}$, $\mathbf{b}'$ are said to *diverge* at vertex u if there are two vertices v, v' such that $(u, v) \in E$, $(u, v') \in E$ and $\mathbf{b} \in T(u, v)$, whereas $\mathbf{b}' \in T(u, v')$.

For every node u in a mechanism $\mathcal{M}$ such that there are two profiles $\mathbf{b}, \mathbf{b}'$ that diverge at u, we say that u is a *divergent node*, and $i = S(u)$ the corresponding *divergent agent*. For each agent i, we define the *current domain* at node u, denoted $D_i(u)$, such that $D_i(r) = D_i$ for the root r and $D_i(u) = \cup_{\mathbf{b}\in T(\phi(u),u)} b_i$. In words, this is the set of types of i that are compatible with the actions that i took during the execution of the mechanism until node u is reached. Indeed, according to the definition, at each node u in which i diverges, $\mathcal{M}$ partitions $D_i(u)$ in k subsets, where k is the number of children of u, and where for every child v of u, $D_i(v) \subset D_i(u)$ contains the types of bidder i compatible with the action that she takes when interacting with the mechanism at node u.

We are now ready to define obvious strategyproofness. An extensive-form mechanism $\mathcal{M}$ is *obviously strategy-proof (OSP)* if for every agent i with real type t_i, for every vertex u such that $i = S(u)$, for every $\mathbf{b}_{-i}, \mathbf{b}'_{-i}$ (with $\mathbf{b}'_{-i}$ not necessarily different from $\mathbf{b}_{-i}$), and for every $b_i \in D_i$, with $b_i \neq t_i$, such that $(t_i, \mathbf{b}_{-i})$ and $(b_i, \mathbf{b}'_{-i})$ are compatible with u, but diverge at u, it holds that $u_i(t_i, \mathcal{M}(t_i, \mathbf{b}_{-i})) \geq u_i(t_i, \mathcal{M}(b_i, \mathbf{b}'_{-i}))$. Roughly speaking, an obviously strategy-proof mechanism requires that, at each time step agent i is asked to take a decision that depends on her type, the worst utility that she can get if she behaves according to her true type is at least the best utility achievable by behaving differently. We stress that our definition does not restrict the alternative behavior to be consistent with a fixed type. Indeed, as noted above, each leaf of the tree $\mathcal{T}_u$ rooted in u corresponds to a profile $\mathbf{b} = (b_i, \mathbf{b}'_{-i})$ compatible with u: then, our definition implies that the utility of i in the leaves where she plays truthfully is at least as much as the utility in every other leaf of $\mathcal{T}_u$. Hence, if a mechanism is obviously strategy-proof, then it is also strategy-proof.

We say that an extensive-form mechanism is *trivial* if for every vertex $u \in V$ and for every two type profiles $\mathbf{b}, \mathbf{b}'$, it holds that $\mathbf{b}$ and $\mathbf{b}'$ do *not* diverge at u. That is, a mechanism is trivial if it never requires agents to take actions that depend on their type. If a mechanism is not trivial, then there is at least one divergent node. On the other hand, every execution of a mechanism (i.e., every path from the root to a leaf in the mechanism implementation tree) may go through at most $\sum_i (|D_i| - 1)$ divergent nodes, the upper bound being the case in which at each divergent node u, the agent $i = S(u)$ separates $D_i(u)$ in $D_i(u) \setminus \{b\}$ and $\{b\}$ for some $b \in D_i(u)$.

Machine Scheduling. Here, we are given a set of m identical jobs to execute and the n agents control related machines. That is, agent i has a job-independent processing time t_i per unit of job (equivalently, an execution speed $1/t_i$ that is independent from the actual jobs). The social choice function f must choose a possible schedule $f(\mathbf{b}) = (f_1(\mathbf{b}), \ldots, f_n(\mathbf{b}))$ of jobs to the machines, where $f_i(\mathbf{b})$ denotes the job load assigned to machine i when agents take actions according to $\mathbf{b}$. The cost that agent i faces for the schedule $f(\mathbf{b})$ is $t_i(f(\mathbf{b})) = t_i \cdot f_i(\mathbf{b})$. We focus on social choice functions f^* minimizing the *makespan*, i.e., $f^*(\mathbf{b}) \in \arg\min_{\mathbf{x}} \max_{i=1}^{n} b_i(\mathbf{x})$. We say that f is ρ-approximate if it returns a solution whose cost is at most ρ times the optimum.

3 Cycle-monotonicity for OSP Mechanisms

We now show how to generalize the cycle-monotonicity technique to design OSP mechanisms.

Let us consider an extensive-form mechanism $\mathcal{M} = (f, \mathbf{p})$ with implementation tree $\mathcal{T}$.

▶ **Definition 1** (separating vertices). *A vertex u in the implementation tree $\mathcal{T}$ is $\alpha\beta$-*separating *for agent i if the following holds: Node u is labelled with i, i.e., $i = S(u)$; there are two profiles $(\alpha, \mathbf{a}_{-i})$ and $(\beta, \mathbf{b}_{-i})$ which are compatible with u but diverge at u, where $\mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u) = \times_{j \neq i} D_j(u)$.*

Note that there might exist several $\alpha\beta$-separating vertices for agent i as the agent may be asked to separate α from β in different paths from the root to a leaf (but only once for every such path).

The algorithmic characterization of OSP we provide herein is based on the following observation.

▶ **Observation 2.** *An extensive-form mechanism $\mathcal{M} = (f, \mathbf{p})$ with implementation tree $\mathcal{T}$ is OSP if and only if for all i, for all $\alpha, \beta \in D_i$, $\alpha \neq \beta$, for all vertices u that are $\alpha\beta$-separating for i:*

$$p_i(\beta, \mathbf{b}_{-i}) - p_i(\alpha, \mathbf{a}_{-i}) \leq \alpha(f(\beta, \mathbf{b}_{-i})) - \alpha(f(\alpha, \mathbf{a}_{-i})) \quad \text{for all } \mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u) \ . \quad (1)$$

We next restate these conditions in terms of suitable weighted graphs and their cycles.

▶ **Definition 3** (OSP-graph). *Let f be a social choice function and $\mathcal{T}$ be an implementation tree. We define for every agent i, the* OSP-graph $OSP_i^{(f,\mathcal{T})}$ *as follows: There is a node for each type profile in D, and a directed edge $e = ((\alpha, \mathbf{a}_{-i}), (\beta, \mathbf{b}_{-i}))$ for every $\alpha, \beta \in D_i$, $\alpha \neq \beta$, and $\mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u)$, where u is an $\alpha\beta$-separating vertex of $\mathcal{T}$. The weight of the edge is $w(e) = \alpha(f(\beta, \mathbf{b}_{-i})) - \alpha(f(\alpha, \mathbf{a}_{-i}))$.*

▶ **Definition 4** (OSP CMON). *We say that the OSP cycle monotonicity (OSP CMON) property holds if, for all i, the graph $OSP_i^{(f,\mathcal{T})}$ does not have negative weight cycles. Moreover, we say that the OSP two-cycle monotonicity (OSP 2CMON) holds if the same is true when considering cycles of length two only, i.e., cycles with two edges only.*

▶ **Theorem 5.** *A mechanism with implementation tree $\mathcal{T}$ is an OSP mechanism for a social function f on finite domains if and only if OSP CMON holds.*

The proof of the theorem follow standard arguments used for the classical definition of strategyproofness. For our application, it is useful to recast the OSP CMON and OSP 2CMON for the case of single-parameter agents.

▶ **Proposition 6.** *For single-parameter settings, OSP 2CMON is equivalent to the following condition. For every i, for any $\alpha, \beta \in D_i$ with $\alpha < \beta$, for any $\alpha\beta$-separating node u of $\mathcal{T}$, with $i = S(u)$, it holds*

$$f_i(\alpha, \mathbf{a}_{-i}) \geq f_i(\beta, \mathbf{b}_{-i}) \qquad \text{for all } \mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u)\ . \tag{2}$$

Warm-up: Using OSP CMON to Bound Approximation Guarantee. We next give a simple lower bound for the machine scheduling problem. This simple result gives a taster of the power of OSP CMON as a tool to answer algorithmic questions about OSP.

▶ **Proposition 7.** *For the machine scheduling problem, no OSP mechanism can be better than 2-approximate, even for two jobs and two agents with three-value domains $D_i = \{L, M, H\}$, where $L < M < H$, with $M > 3L$ and $H > 3M$.*

Proof. Assume by contradiction that there is an OSP mechanism $\mathcal{M}$ that is better than 2-approximate, and let $\mathcal{T}$ be its implementation tree. Since $M > 3L$ and $H > 3M$, every trivial OSP mechanism must have approximation guarantee at least 2. Hence $\mathcal{M}$ must be non trivial. Let i be the first divergent agent of $\mathcal{M}$ implemented with $\mathcal{T}$, and let u be the node where this agent diverges (such an agent exists because the mechanism is not trivial). We show that this mechanism cannot satisfy OSP 2CMON, thus a contradiction.

If i diverges at u on M and H, then consider $\mathbf{b} = (\beta, \mathbf{b}_{-i}) = (H, H)$ and $\mathbf{a} = (\alpha, \mathbf{a}_{-i}) = (M, L)$. Since the mechanism is better than 2-approximate, it must satisfy $f_i(\beta, \mathbf{b}_{-i}) = 1$ and $f_i(\alpha, \mathbf{a}_{-i}) = 0$. Note that this violates the OSP 2CMON condition (Equation 2 in Proposition 6): Since i is the first divergent agent, and u is the corresponding node, the set $D_{-i}(u)$ consists of all types in the domain of the other agent, and therefore $H, L \in D_{-i}(u)$ as required to invoke (2) with our choice $\mathbf{b}_{-i} = H$ and $\mathbf{a}_{-i} = L$. If i diverges at u on L and M, then consider $\mathbf{a} = (\alpha, \mathbf{a}_{-i}) = (L, L)$ and $\mathbf{b} = (\beta, \mathbf{b}_{-i}) = (M, H)$. Since the mechanism is better than 2-approximate, it must satisfy $f_i(\alpha, \mathbf{a}_{-i}) = 1$ and $f_i(\beta, \mathbf{b}_{-i}) = 2$. Similarly to the previous case, this violates the OSP 2CMON condition (2). ◀

Note that for this bound we require the domain to have at least three different values; we will in fact prove in Section 4 that we can design an optimal OSP mechanism for scheduling related machines when $D_i = \{L_i, H_i\}$ for every i. We will also show how to use a more involved argument to prove a substantially higher (and tight) bound of $\sqrt{n}$.

Two-cycles are Sufficient for Single-parameter Domains of Size at most Three. Two-cycle monotonicity is a property easier to work with than CMON. We will now observe that, for single parameter settings, these properties turns out to be equivalent if and only if $D_i = \{L_i, M_i, H_i\}$ for each i, with $L_i \leq M_i \leq H_i$.

▶ **Theorem 8.** *Consider a single-parameter setting where $|D_i| \leq 3$ for each agent i. A mechanism with implementation tree $\mathcal{T}$ and social choice function f is OSP iff OSP 2CMON holds.*

We next show that this result is essentially tight in the sense that OSP 2CMON does not imply OSP CMON (and thus OSP-ness) already in four-value domains.

▶ **Theorem 9.** *There exists a mechanism for which OSP 2CMON holds for every agent, but there is an agent i for which the mechanism does not satisfy OSP CMON, whenever $|D_i| \geq 4$. The claim holds even for a single-item auction setting and $D_j = D$ for every $j \neq i$.*

4 Scheduling Related Machines

In this section, we show how the domain structure impacts on the performance guarantee of OSP mechanisms, for the problem of scheduling related machines. Roughly speaking, the problem is easy for *two-value* domains, while it becomes difficult already for *three-value* domains and *two* jobs.

We can prove that an OSP optimal mechanism exists for the case in which each agent's domain has size two. Specifically, we have the following theorem.

▶ **Theorem 10.** *For the machine scheduling problem, there exists an optimal polynomial-time OSP mechanism for any number of agents with two-value domains $D_i = \{L_i, H_i\}$.*

Lower Bound for Three-value Domain

We now show how to strengthen Proposition 7 and prove a $\sqrt{n}$-inapproximability result for three-value domains.

▶ **Theorem 11.** *For the machine scheduling problem, no OSP mechanism can be better than $\sqrt{n}$-approximate. This also holds for three-value domains $D_i = \{L, M, H\}$.*

For the proof, we consider $m = n = c^2$, for some $c > 1$, and a three-value domain $D_i = \{L, M, H\}$ such that $M \geq m \cdot L$ and $H \geq m\sqrt{n} \cdot M$. Observe that, in such domains, every trivial mechanism must have an approximation ratio not lower than $\sqrt{n}$. Consider then a non-trivial mechanism $\mathcal{M}$ and let $\mathcal{T}$ be its implementation tree. Let us rename the agents as follows: Agent 1 is the 1st agent that diverges in $\mathcal{T}$; since the mechanism is not trivial agent 1 exists. We now call agent 2, the 2nd distinct agent that diverges in the subtree of $\mathcal{T}$ defined by agent 1 taking an action compatible with type H; if no agent diverges in this subtree of $\mathcal{T}$ we simply call 2 an arbitrary agent different from 1. More generally, agent i is the ith distinct agent that diverges, if any, in the subtree of $\mathcal{T}$ that corresponds to the case that the actions previously taken by agents are compatible with their type being H. As above, if no agent diverges in the subtree of interest, we just let i denote an arbitrary agent different from $1, 2, \ldots, i-1$. We denote with u_i the node in which i first diverges in the subtree in which all the other agents have taken actions compatible with H; if i does not diverge (i.e., got her id arbitrarily) we denote with u_i a dummy node in which we will say that i does not diverge and i takes an action compatible with every type in D_i. We then have the following lemma.

▶ **Lemma 12.** *Any OSP mechanism $\mathcal{M}$ which is k-approximate, with $k < \sqrt{n}$, must satisfy:*
1. *For every $i \leq n - \sqrt{n} + 1$, if agent i diverges at node u_i, it must diverge on M and H.*
2. *For every $i \leq n - \sqrt{n}$, if agent i diverges at node u_i and takes an action compatible with her type being H, then $\mathcal{M}$ does not assign any job to i, regardless of the actions taken by the other agents.*

Proof. Let us first prove part (1). Suppose that there is $i \leq n - \sqrt{n} + 1$ such that at node u_i i diverges on L and $\{M, H\}$. Consider the type profile $\mathbf{x}$ such that $x_i = M$, and $x_j = H$ for every $j \neq i$. Observe that $\mathbf{x}$ is compatible with node u_i. The optimal allocation for the type profile $\mathbf{x}$ assigns all jobs to machine i, with cost $OPT(\mathbf{x}) = mM$. Since $\mathcal{M}$ is k-approximate, then it also assigns all jobs to machine i. Indeed, if a job is assigned to a machine $j \neq i$, then the cost of the mechanism would be at least $H \geq \sqrt{n} \cdot mM > k \cdot OPT(\mathbf{x})$, that contradicts the approximation bound.

Consider now the profile $\mathbf{y}$ such that $y_i = L$, $y_j = H$ for every $j < i$, and $y_j = L$ for every $j > i$. Observe that also $\mathbf{y}$ is compatible with node u_i. It is not hard to see that $OPT(\mathbf{y}) \leq \left\lceil \frac{m}{n-i+1} \right\rceil \cdot L$. Since $\mathcal{M}$ is k-approximate, then it cannot assign all jobs to machine i. Indeed, in this case the cost of the mechanism contradicts the approximation bound, since it would be $mL \geq \sqrt{n} \left\lceil \frac{m}{n-i+1} \right\rceil L > k \cdot OPT(\mathbf{y})$, where we used that $\sqrt{n} \left\lceil \frac{m}{n-i+1} \right\rceil \leq \sqrt{n} \left\lceil \frac{m}{\sqrt{n}} \right\rceil = \sqrt{n} \left\lceil \frac{n}{\sqrt{n}} \right\rceil = \sqrt{n} \cdot \sqrt{n} = n = m$.

Hence, we have that if i takes actions compatible with M, then there exists a type profile compatible with u_i such that i receives n jobs, whereas, if i takes a different action compatible with a lower type, then there exists a type profile compatible with u_i such that i receives less than n jobs. However, this contradicts the OSP CMON property.

Let us now prove part (2). Suppose that there is $i \leq n - \sqrt{n}$ and $\mathbf{x}_{-i}$ compatible with u_i such that if i takes an action compatible with type H, then $\mathcal{M}$ assigns at least a job to i. According to part (1), machine i diverges at node u_i on H and M.

Consider then the profile $\mathbf{y}$ such that $y_i = M$, $y_j = H$ for $j < i$, and $y_j = L$ for $j > i$. It is easy to see that the optimal allocation has cost $OPT(\mathbf{y}) = \left\lceil \frac{m}{n-i} \right\rceil \cdot L$. Since $\mathcal{M}$ is k-approximate, then it does not assign any job to machine i. Otherwise, the mechanism contradicts the approximation bound since his cost would be at least $M \geq mL \geq \sqrt{n} \left\lceil \frac{m}{n-i} \right\rceil L > k \cdot OPT(\mathbf{x})$, where we used that $\sqrt{n} \left\lceil \frac{m}{n-i} \right\rceil \leq \sqrt{n} \left\lceil \frac{m}{\sqrt{n}} \right\rceil = \sqrt{n} \left\lceil \frac{n}{\sqrt{n}} \right\rceil = \sqrt{n} \cdot \sqrt{n} = n = m$.

Hence, we have that if i takes actions compatible with H, then there exists a type profile compatible with u_i such that i receives one job, whereas, if i takes a different action compatible with a lower type, then there exists a type profile compatible with u_i such that i receives zero jobs. However, this contradicts the OSP CMON property. ◀

Proof of Theorem 11. Suppose that there is an OSP k-approximate mechanism $\mathcal{M}$ for some $k < \sqrt{n}$, thus implying that the mechanism is not trivial.

Assume first that for all $i \leq n - \sqrt{n}$ agent i diverges at u_i. Consider $\mathbf{x}$ such that $x_i = H$ for every i. Observe that $\mathbf{x}$ is compatible with u_i for every i. The optimal allocation consists in assigning a job to each machine, and has cost $OPT(\mathbf{x}) = H$. According to Part (2) of Lemma 12, if machines take actions compatible with $\mathbf{x}$, then the mechanism $\mathcal{M}$ does not assign any job to machine i, for every $i \leq n - \sqrt{n}$. Hence, the best outcome that $\mathcal{M}$ can return for $\mathbf{x}$ consists in assigning $\sqrt{n}$ jobs to each of the other $\sqrt{n}$ machines. Therefore, the cost of $\mathcal{M}$ is at least $\sqrt{n}H > kOPT(\mathbf{x})$, which contradicts the approximation ratio of $\mathcal{M}$.

Consider now the case that there is $1 < i \leq n - \sqrt{n}$ that does not diverge at u_i (since the mechanism is not trivial $i > 1$). This means that all the machines $j \geq i$ will not diverge at u_i; let S denote this set of machines. Note that the $n - i + 1 \geq \sqrt{n} + 1$ machines in S will have

the same outcome no matter their types when the machines not in S have type H; in other words, any profile $\mathbf{x}$ where $x_j = H$ for $j \notin S$ is compatible with u_i. Consider $\mathbf{x}$ such that $x_j = H$ for $j \notin S$ and $x_j = L$ otherwise. Since $H \geq n^{5/2}L$, to guarantee approximation k, the mechanism must return a solution for $\mathbf{x}$ which keeps the machines not in S empty; then there is a $j^* \in S$ which is allocated at least $\left\lceil \frac{n}{\sqrt{n}+1} \right\rceil$ jobs. Consider now $\mathbf{y}$ where $y_j = H$ for $j \notin S \setminus \{j^*\}$ and $y_j = L$ otherwise. The mechanism must give in output the same allocation given in output for $\mathbf{x}$ since it cannot distinguish $\mathbf{x}$ from $\mathbf{y}$. However, giving that many jobs to machine j^* contradicts the approximation guarantee on $\mathbf{y}$. ◀

The arguments above can be used to prove that ascending and descending auctions do not help in this setting. Specifically, they cannot return an approximation better than n.

Upper bound for Three-value Domain

We describe our mechanisms for a generic domain, as this turns out to be useful in the analysis. In what follows, the usual bold notation $\mathbf{x}$ denotes vectors of n entries, while a "hat-bold" notation $\hat{\mathbf{x}}$ denotes vectors of $\lceil \sqrt{n} \rceil$ entries only.

A Mechanism for Many Jobs (Large m). We now introduce mechanism $\mathcal{M}_{many}$ whose approximation ratio approaches $\lceil \sqrt{n} \rceil$, whenever $m \gg \lceil \sqrt{n} \rceil$. The mechanism consists of a descending Phase (Algorithm 1) followed by an ascending Phase (Algorithm 2). The descending phase simply queries the agents to identify (and forget about) the $n - \sqrt{n}$ slowest machines; the ascending phase instead identifies the fastest machine and then computes the optimal solution to a vector where the types of the remaining $\sqrt{n} - 1$ machines is set to the best type of the slow machines found in the descending phase.

Algorithm 1 Descending Phase (for both mechanisms $\mathcal{M}_{many}$ and $\mathcal{M}_{few}$).

```
Set A = [n], and t_i = max{d ∈ D_i}
while |A| > ⌈√n⌉ do
    Set p = max_{a∈A}{t_a} and i = min{a ∈ A : t_a = p}
    Ask machine i if her type is equal to p
    if yes then remove i from A
    else set t_i = max{t ∈ D_i : t < p}
```

▶ **Proposition 13.** *Mechanism $\mathcal{M}_{many}$ is OSP for any three-value domain $D_i = \{L_i, M_i, H_i\}$.*

Proof. We prove that $\mathcal{M}_{many}$ satisfies OSP 2CMON. The claim then follows from Theorem 8. Specifically, for each machine i, for each node u in which the mechanism makes a query to i, for each pair of type profiles $\mathbf{x}, \mathbf{y}$ compatible with u such that i diverges at u between x_i and y_i, we need to prove that if $x_i > y_i$, then $f_i(\mathcal{M}_{many}(\mathbf{x})) \leq f_i(\mathcal{M}_{many}(\mathbf{y}))$.

Let us first consider a node u corresponding to the descending phase of the mechanism. In this case, $x_i = p$, where p is as at node u. Moreover, in all profiles compatible with u there are at least $\lceil \sqrt{n} \rceil$ machines that either have a type lower than p, or they have type p but are queried after i. However, for every $\mathbf{x}_{-i}$ satisfying this property, we have that $f_i(\mathcal{M}_{many}(\mathbf{x})) = 0$, which implies that these two-cycles have non-negative weight.

Suppose now that node u corresponds to the ascending phase of the mechanism. In this case, $y_i = p$, where p is as at node u. Observe that for every $\mathbf{y}_{-i}$ compatible with node u, $f_i(\mathcal{M}_{many}(\mathbf{y})) = f_i^\star(y_i, \hat{\mathbf{z}}_{-i})$, where $f_i^\star(y_i, \hat{\mathbf{z}}_{-i})$ is the number of jobs assigned to machine i by the optimal outcome on input profile $\widehat{(y_i, \mathbf{z})_{-i}}$, $\hat{\mathbf{z}}_{-i}$ being such that $\hat{z}_j = \max_{k \in A} t_k$

for every $j \in A \setminus \{i\}$. Observe that for every $\mathbf{x}$ compatible with u, it must be the case that $x_j \geq y_i$ for every $j \in A$. Hence, we can distinguish two cases: if $\min_j x_j = x_i$, then $f_i(\mathcal{M}_{many}(\mathbf{x})) = f_i^\star(x_i, \hat{\mathbf{z}}_{-i}) \leq f_i^\star(y_i, \hat{\mathbf{z}}_{-i}) = f_i(\mathcal{M}_{many}(\mathbf{y}))$; if instead $\min_j x_j = x_k$, for some $k \neq i$, then $f_i(\mathcal{M}_{many}(\mathbf{x})) = f_i^\star(x_k, \hat{\mathbf{z}}_{-k}) \leq f_k^\star(x_k, \hat{\mathbf{z}}_{-k}) \leq f_i^\star(y_i, \hat{\mathbf{z}}_{-i}) = f_i(\mathcal{M}_{many}(\mathbf{y}))$, where we used that $\hat{\mathbf{z}}_{-k} = \hat{\mathbf{z}}_{-i}$ and the inequalities follow since: (i) in the optimal outcome the fastest machine must receive at least as many jobs as slower machines; (ii) the optimal outcome is monotone, (i.e., given the speeds of other machines, the number of jobs assigned to machine i decreases as its speeds decreases). ◀

Algorithm 2 Ascending Phase ($\mathcal{M}_{many}$).

1 Set $s_i = \min\{d \in D_i\}$
2 **while** $|A| > 0$ **do**
3 Set $p = \min_{a \in A}\{s_a\}$ and $i = \min\{a \in A \colon s_a = p\}$
4 Ask machine i if her type is p
5 **if** *yes* **then**
6 Let $\hat{\mathbf{z}}$ be s.t. $\hat{z}_i = p$ and $\hat{z}_j = \min_{k \notin A} t_k$ for $j \in A, j \neq i$
7 Let $f^\star(\hat{\mathbf{z}}) = (f_i^\star(\hat{\mathbf{z}}))_{i \in A}$ be the optimal assignment for profile $\hat{\mathbf{z}}$
8 Assign $f_j^\star(\hat{\mathbf{z}})$ jobs to each $j \in A$
9 Set $A = \emptyset$
10 **else** set $s_i = \min\{d \in D_i \colon d > p\}$

Algorithm 3 Ascending Phase ($\mathcal{M}_{few}$).

1 Set $t_a = \min_i\{d \in D_i\}$ and $C = m$
2 **while** $|A| > 0$ **do**
3 Set $q = \min_{a \in A}\{t_a\}$ and $i = \min\{a \in A \colon t_a = q\}$
4 Ask machine i if her type is q
5 **if** *yes* **then**
6 Let $\zeta = \lceil C/|A| \rceil$
7 Let z be the largest integer in $[\zeta, C]$ such that $z \cdot q \leq \lceil \sqrt{n} \rceil \cdot p$
8 Assign z jobs to i
9 Set $C = C - z$
10 Remove i from A
11 **else** set $t_i = \min\{d \in D_i \colon d > p\}$

The next theorem bounds the approximation ratio of the mechanism

▶ **Theorem 14.** *Mechanism $\mathcal{M}_{many}$ is $(\lceil \sqrt{n} \rceil + 1)$-approximate for $m > \lceil \sqrt{n} \rceil^2$.*

A Mechanism for Few Jobs (Small m). We now introduce a mechanism $\mathcal{M}_{few}$ which is OSP and $\lceil \sqrt{n} \rceil$-approximate whenever $m \leq \lceil \sqrt{n} \rceil^2$. Like $\mathcal{M}_{many}$, $\mathcal{M}_{few}$ consists of a descending phase followed by an ascending phase. The descending phase is exactly the same (Algorithm 1) with the difference that the ascending phase (Algorithm 3) does not need the information on the type of the machines that are not in A at that point.

We show next that $\mathcal{M}_{few}$ is well defined under our assumption on m, is OSP and has approximation $\lceil \sqrt{n} \rceil$.

▶ **Lemma 15.** *If $m \leq \lceil \sqrt{n} \rceil^2$ then there exists a z in line 7 of Algorithm 3.*

Proof. We next show that it never occurs during the ascending phase that $\zeta \cdot q > \lceil \sqrt{n} \rceil \cdot p$. Indeed, for the first machine to reveal the type during the ascending phase, we have that $|P_0| = m \leq \lceil \sqrt{n} \rceil^2$, and, thus $\zeta \leq \lceil \sqrt{n} \rceil$. Hence, $\zeta \cdot q \leq \lceil \sqrt{n} \rceil \cdot p$ since $q \leq p$. If a set $Q \subset A$ of machines has previously revealed the type during the ascending phase, and the execution of this phase has not been stopped, then these machines received at least $m' = \lfloor |Q| m/|A| \rfloor + \min\{|Q|, m \mod |A|\}$ jobs. Then $|P_0| = m - m' \leq (|A| - |Q|) \lceil \sqrt{n} \rceil$, and thus $\zeta \leq \lceil \sqrt{n} \rceil$, and, since $q \leq p$, $\zeta \cdot q \leq \lceil \sqrt{n} \rceil \cdot p$. ◀

▶ **Proposition 16.** *Mechanism $\mathcal{M}_{few}$ is OSP for three-value domains $D_i = \{L_i, M_i, H_i\}$.*

Proof. We prove that $\mathcal{M}_{many}$ satisfies the OSP 2CMON. The claim then follows from Theorem 8. Specifically, for each machine i, for each node u in which the mechanism makes a query to i, for each pair of type profiles $\mathbf{x}, \mathbf{y}$ compatible with u such that x_i and y_i diverge at u, we need to prove that if $x_i > y_i$, then $f_i(\mathcal{M}_{few}(\mathbf{x})) \leq f_i(\mathcal{M}_{few}(\mathbf{y}))$.

Let us first consider a node u corresponding to the descending phase of the mechanism. In this case, $x_i = p$, where p is as at node u. Moreover, in all profiles compatible with u there are at least $\lceil \sqrt{n} \rceil$ machines that either have a type lower than p, or they have type p but they are queried after i. Hence, for every $\mathbf{x}_{-i}$ satisfying this property, we have that $f_i(\mathcal{M}_{few}(\mathbf{x})) = 0$, that implies the claim.

Suppose now that node u corresponds to the ascending phase of the mechanism. Let $C(u)$, $A(u)$, $p(u)$ and $q(u)$ be the value of C, A, p and q at that node. Observe that for every profile compatible with u, the type of machines not in $A(u)$ is fixed, whereas for every machine in $A(u)$, the type is at least $q(u)$. Moreover, $y_i = q(u)$. Hence, $f_i(\mathcal{M}_{few}(\mathbf{y}))$ is the largest integer $z \leq C(u)$ such that $z \cdot y_i \leq \lceil \sqrt{n} \rceil \cdot p(u)$. On the other side, for every $x_i > y_i$, $f_i(\mathcal{M}_{few}(\mathbf{x}))$ is at most the largest integer $z' \leq C(u)$ such that $z' \cdot x_i \leq \lceil \sqrt{n} \rceil \cdot p(u)$. Since $x_i > y_i$, then $z' \leq z$, and the lemma follows. ◀

▶ **Proposition 17.** *Mechanism $\mathcal{M}_{few}$ is $\lceil \sqrt{n} \rceil$-approximate.*

5 Conclusions

We have focused on OSP mechanisms, a compelling and needed notion of incentive compatibility for bounded rationality; [23] proves that OSP is the notion that captures strategyproofness for agents who lack contingent reasoning skills. It is thus paramount to understand the limitations and the power of these mechanisms.

We have introduced a new technique to look at OSP mechanisms, and shown its power by giving tight results on the approximation for a paradigmatic problem in the area. Our contribution highlights how there are two dimensions, algorithms and their implementation, to the design of these mechanisms. The interplay between these dimensions is encapsulated by OSP CMON and plays a central role, as shown by the limitations of fixing the implementation beforehand (as in DA auctions or direct revelation mechanisms).

Furthermore, the significance of the technique can be seen by comparing the previously known lower bounds on the approximation guarantee of OSP mechanisms given in [12, 5]. These results focus on the first divergent agent only and bound the *strategyproof* payments for the identified instances in order to understand and limit the behavior of the algorithm. As a result, their bounds are small constants (2 for machine scheduling in [12] and $1 + \epsilon$, for combinatorial auctions with additive bidders in [5]).

We leave a number of open problems. A technical one is about the domain size and the difference between 2-cycles and longer ones; to what extent adding an extra type in the domain can deteriorate the approximation ratio of OSP mechanisms? A second, more conceptual question, is about dealing with multi-parameter agents. Even with the machinery of OSP CMON, it does not seem immediate to characterize the implementation trees for this kind of agents as there is not a concept of relative ordering of types. Hence, the common pattern of OSP mechanisms, where at each node of the implementation tree an extreme of the current domain is separated from the rest, cannot be adopted.

References

1 M. Adamczyk, A. Borodin, D. Ferraioli, B. de Keijzer, and S. Leonardi. Sequential Posted Price Mechanisms with Correlated Valuations. In *WINE 2015*, pages 1–15, 2015.

2 I. Ashlagi and Y. A. Gonczarowski. Stable Matching Mechanisms are not Obviously Strategy-proof. *J. Economic Theory*, 177:405–425, 2018.

3 L. M Ausubel. An Efficient Ascending-bid Auction for Multiple Objects. *American Economic Review*, 94(5):1452–1475, 2004.

4 M. Babaioff, N. Immorlica, B. Lucier, and S M. Weinberg. A Simple and Approximately Optimal Mechanism for an Additive Buyer. In *FOCS 2014*, pages 21–30, 2014.

5 S. Bade and Y. A. Gonczarowski. Gibbard-Satterthwaite success stories and obvious strategyproofness. In *EC 2017*, page 565, 2017.

6 S. Bikhchandani, S. Chatterji, R. Lavi, A. Muálem, N. Nisan, and A. Sen. Weak Monotonicity Characterizes Deterministic Dominant-Strategy Implementation. *Econometrica*, 74(4):1109–1132, 2006.

7 S. Chawla, J. Hartline, D. Malec, and B. Sivan. Multi-parameter Mechanism Design and Sequential Posted Pricing. In *STOC 2010*, pages 311–320, 2010.

8 J. Correa, P. Foncea, R. Hoeksma, T. Oosterwijk, and T. Vredeveld. Posted Price Mechanisms for a Random Stream of Customers. In *EC 2017*, pages 169–186, 2017.

9 P. Dütting, V. Gkatzelis, and T. Roughgarden. The Performance of Deferred-Acceptance Auctions. *Math. Oper. Res.*, 42(4), 2017.

10 A. Eden, M. Feldman, O. Friedler, I. Talgam-Cohen, and S. M. Weinberg. A Simple and Approximately Optimal Mechanism for a Buyer with Complements. In *EC 2017*, pages 323–323, 2017.

11 M. Feldman, A. Fiat, and A. Roytman. Makespan Minimization via Posted Prices. In *EC 2017*, pages 405–422, 2017.

12 D. Ferraioli and C. Ventre. Obvious Strategyproofness Needs Monitoring for Good Approximations. In *AAAI 2017*, pages 516–522, 2017.

13 D. Ferraioli and C. Ventre. Probabilistic Verification for Obviously Strategyproof Mechanisms. In *IJCAI 2018*, 2018.

14 D. Ferraioli and C. Ventre. Obvious Strategyproofness, Bounded Rationality and Approximation: The Case of Machine Scheduling. In *SAGT 2019*, 2019.

15 V. Gkatzelis, E. Markakis, and T. Roughgarden. Deferred-acceptance auctions for multiple levels of service. In *EC 2017*, pages 21–38, 2017.

16 H. Gui, R. Müller, and R. V. Vohra. Dominant Strategy Mechanisms with Multidimensional Types. Discussion paper 1392, Northwestern Univ., 2004.

17 J. Hartline and T. Roughgarden. Simple versus Optimal Mechanisms. In *EC 2009*, pages 225–234, 2009.

18 J. Kagel, R. Harstad, and D. Levin. Information Impact and Allocation Rules in Auctions with Affiliated Private Values: A Laboratory Study. *Econometrica*, pages 1275–1304, 1987.

19 A. Kim. Welfare Maximization with Deferred Acceptance Auctions in Reallocation Problems. In *ESA 2015*, pages 804–815, 2015.

20 P. Krysta and C. Ventre. Combinatorial Auctions with Verification are Tractable. *Theor. Comput. Sci.*, 571:21–35, 2015.

21 M. Kyropoulou and C. Ventre. Obviously Strategyproof Mechanisms without Money for Scheduling. In *AAMAS 2019*, 2019.

22 R. Lavi and C. Swamy. Truthful Mechanism Design for Multi-dimensional Scheduling via Cycle Monotonicity. *Games and Economic Behavior*, 67(1):99–124, 2009.

23 S. Li. Obviously Strategy-proof Mechanisms. *American Economic Review*, 107(11):3257–87, 2017.

24 A. Mackenzie. A Revelation Principle for Obviously Strategy-proof Implementation. Research Memorandum 014, (GSBE), 2017.

25 A. Malakhov and R. V. Vohra. Single and Multi-Dimensional Optimal Auctions - A Network Approach. Discussion paper 1397, Northwestern Univ., 2004.

26 P. Milgrom and I. Segal. Deferred-acceptance Auctions and Radio Spectrum Reallocation. In *EC 2014*, 2014.

27 M. Pycia and P. Troyan. Obvious Dominance and Random Priority. In *EC 2019*, 2019.

28 J.-C. Rochet. The Taxation Principle and Multitime Hamilton-Jacobi Equations. *Journal of Mathematical Economics*, 14(2):113–128, 1985.

29 M. Saks and L. Yu. Weak Monotonicity Suffices for Truthfulness on Convex Domains. In *EC 2005*, pages 286–293, 2005.

30 C. Ventre. Truthful Optimization Using Mechanisms with Verification. *Theor. Comput. Sci.*, 518:64–79, 2014.

31 L. Zhang and D. Levin. Bounded rationality and robust mechanism design: An axiomatic approach. *American Economic Review*, 107(5):235–39, 2017.