New Constructions of Obviously Strategyproof Mechanisms*

Diodato Ferraioli† Adrian Meier‡ Paolo Penna† Carmine Ventre§

Abstract

Catering to the incentives of people with limited rationality is a challenging research direction that requires novel paradigms to design mechanisms. 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 and a *characterization* of set system instances for which OSP mechanisms that return optimal solutions exist. 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, when private types belong to a set with few values. In essence, we prove that OSP encompasses careful interleaving of ascending and descending auctions.

1 Introduction

Mechanism design has been a very active research area that aims to develop algorithms (a.k.a., social choice functions) 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 applicative 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 [28].

A vague explanation of this phenomenon is that, from the point of view of a bidder, the strategyproofness (a.k.a., truthfulness) of an ascending price auction is easier to grasp than the strategyproofness of the second-price sealed bid auction [6]. The key difference here is the way these two 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.

*Preliminary versions of the results herein have previously appeared in [19, 18].

†Università di Salerno, Email: dferraioli@unisa.it.

‡ETH Zurich, Email: meiera@student.ethz.ch, paolo.penna@inf.ethz.ch.

§King's College London, Email: carmine.ventre@kcl.ac.uk.

- In the ascending price auction (extensive-form implementation), each bidder is repeatedly offered some price that 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-case scenario* when truth-telling is at least as good as that of the *best-case scenario* when cheating. This intuition has been formalized by the recent definition of *obviously strategyproof (OSP)* mechanisms [33]: indeed, this definition allows to show that ascending auctions are OSP, while sealed-bid auctions are not. Interestingly, [33] proves that a mechanism is OSP if and only if truth-telling is dominant even for bidders who lack contingent reasoning skills, thus addressing a specific form of bounded rationality.

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 [36] are OSP for single-parameter agents[1] (as they essentially are ascending price auctions), but unfortunately their performance (measured in terms of their approximation guarantee to a given objective function) for several optimization problems is quite poor compared to what strategyproof mechanisms can do [14]. 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 quality of the solutions (e.g., approximation) and the time needed to compute a solution (e.g., complexity). The same type of questions seem much more challenging for OSP mechanisms, as such characterizations are not known. Recent work in the area [8, 39, 34] provided different versions of the revelation principle for OSP mechanisms. These versions allow to conveniently restrict how the mechanism should interact with the agents: e.g., it turns out that it is without loss of generality to consider mechanisms that interact with agents sequentially (rather than concurrently). However, just as the revelation principle is not sufficient to design truthful mechanisms, and further properties, such as monotonicity, have been introduced, we here look for similar properties that will help to design OSP mechanisms, in conjunction with the aforementioned notions of the OSP revelation principles.

1.1 An Example

To better exemplify this conundrum, let us consider path auctions introduced by [37]. In this problem, we have a network in which each edge is owned by a selfish agent. The agent owning link i has some private nonnegative cost t_i if her link is used, and zero cost otherwise. The utility of an agent is the payment received from the mechanism minus her cost. We remark that this cost is known only to agent i. We want to design a mechanism that is able to pick a path of minimal total cost (shortest path) between two given nodes s and t of the network. Since the standard VCG

[1] These are agents who report a single number to the mechanism – see below for a formal definition.

mechanism yields an (exact) strategyproof mechanism for this problem [37], one might consider the following natural question:

Can we compute the shortest path whilst guaranteeing OSP?

For a graph consisting of *parallel links*, we know from [33] that the answer is yes via a simple descending auction to select the cheaper edge. Already for *slightly* more general graphs, the answer is unclear. Consider, for example, the graph in Figure 1(a). To make things even simpler let us restrict to a two-value domain $\{L, H\}$, i.e., edges cost either L or $H > L$. In this setting, a simple OSP mechanism can be designed by querying the agents according to the *implementation tree* (i.e., extensive-form game used to implement the mechanism) in Figure 1(b). This algorithm is augmented with the following payments: H for edges in the selected path, 0 otherwise. It is not hard to see that, for every edge e, it is not possible that e is selected when she declares that her type is H and is not selected when she says L. In particular, edge (s, t) is always selected when she says L, while the remaining edges are never selected when they declare H. Then, if e declares her true type, she receives a utility of $H - L$ if the true type is L and e is selected, and 0 otherwise; by inspection, she would receive at most the same utility when cheating. It turns out that this line of reasoning is enough to prove that the mechanism is indeed OSP.

Does the same approach work, for example, on the slightly different graph in Figure 1(c)? Consider an edge e that is queried before the type of the remaining edges is known (that is, the first edge to be queried in the implementation tree of the mechanism). Suppose that the type of this edge is L. If she declares her type truthfully, then the worst that may occur is that the corresponding path is not selected (that occurs when this path costs $H + L$ and the alternative path costs $2L$), and thus e receives utility 0. If this edge, instead, cheats and declares H, then it is possible that the corresponding path is selected (if it costs $H + L$ and the alternative path costs $2H$) and e receives utility $H - L$. Thus, it is not obvious for an edge e lacking contingent reasoning skills, to understand that revealing the type truthfully is dominant. This raises the following questions: Is there a different payment rule that enables to design an optimal mechanism that is OSP in the latter case? If not, how how much longer than the shortest is the path that we can compute with OSP mechanisms?

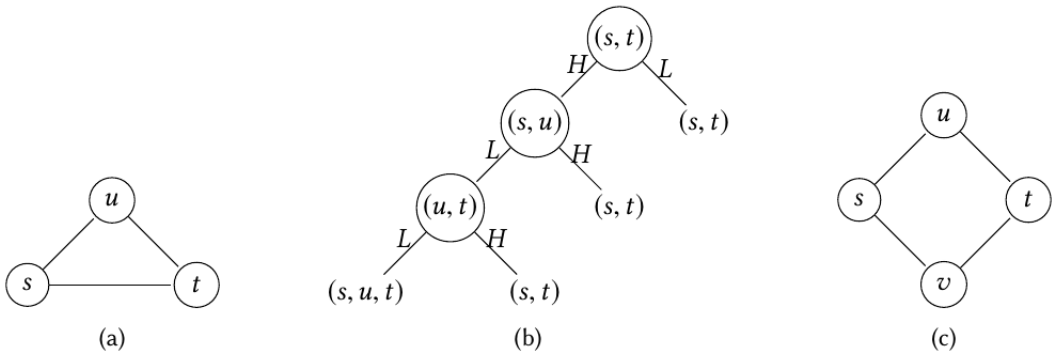

Figure 1: Two instances of path auctions are shown in (a) and (c), while (b) is an OSP mechanism for instance (a)

The goal of this work is to 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. Our goal is to understand the extent to which it is possible to design *OSP optimal mechanisms*, that is, OSP mechanisms that return an optimal solution for the optimization problem at hand. Whenever optimality is not compatible with OSP, we want to have bounds on the approximation guarantee of these mechanisms. Among others, our results show why deferred acceptance auctions [36] – 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.

1.2 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. In its more general form, the graph for truthfulness of agent i is complete and has a vertex for each possible outcome; edge weights account for the incentives of agent i in forcing a different outcome by lying.

We show that it is possible to carefully define graphs modeling the imperfectly rational agents' incentives so that non-negative weight cycles continue to characterize OSP. Our main conceptual contribution is here 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. Specifically, we define a graph for each agent i (just as in the case of truthfulness) with a node for each strategy profile (as opposed to each outcome) and we add an edge between two vertices only when there is an OSP constraint for i involving those profiles; the OSP constraints *depend on the particular extensive-form implementation* of the mechanism using the algorithm at hand. We prove that payments exist that can be paired to the algorithm in an OSP mechanism, implemented in the given extensive-form, if and only if this graph does not contain negative cycles.

Interestingly, our technique shows the interplay between algorithms (which outcome/solution to return) and how the mechanism is implemented as an extensive-form game (what we call the *implementation tree*). Roughly speaking, our characterization says which algorithms can be used given an 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 [36] always uses *the same* fixed extensive-form game 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 both characterize 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 for the problem of scheduling n related machines (for identical jobs) and a *characterization* of OSP optimal mechanisms for set system problems (which include path auctions as a special case). A caveat for our results is about the size of the agents' domains. While our lower bounds/necessary conditions hold regardless of the size of the domain, the mechanisms that we provide are shown to be OSP only for two- and three-value (agent-specific) domains, as we prove that these are the only cases in which non-negative two-cycles are necessary and sufficient. The treatment of longer cycles has been object of subsequent work, which shows how powerful, flexible and useful our OSP CMON technique is. In particular, an OSP mechanism for a variant of combinatorial auctions with single-minded bidders is given in [13]; its approximation ratio

matches the one of the best known SP mechanism. The authors of [20] provide a characterization of OSP mechanisms as a combination of greedy and deferred acceptance algorithms, for every set system. The OSP mechanisms we design herein are not restricted to binary allocation problems and show how it is possible to combine ascending and descending phases in the implementation trees to achieve OSP and obtain approximation guarantees better than what is possible with mechanisms that use only one phase, such as deferred acceptance.

Machine Scheduling. We here want to allocate a set of (identical) jobs to a set of n machines with job-independent and private speeds; the objective is to minimize the makespan (i.e., the latest completion time of a machine). We show that the optimum for machine scheduling can be implemented OSP-ly when the agents' domains have size two. We prove that given the optimum allocation that keeps the machine loads as balanced as possible, we can always find an implementation tree for which OSP is guaranteed. The mechanism directly asks the queried agents to reveal their type.

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 OSP 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.

> **Main Theorem about Machine Scheduling (informal).** *The tight approximation guarantee of OSP mechanisms that can be guaranteed over all three-value domains is $O(\sqrt{n})$. 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)).*

On a 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, the largest and the 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. 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. We in fact extend and adapt our analysis to prove that any ascending or descending (thus including DA) auction has an approximation of n (cf. Appendix B.4).

Set Systems. We first exemplify our approach on a type of set systems, namely path auctions on graphs comprised of two parallel paths, whose edges have two-value domains. We show how the topology (i.e., number of edges of either path) and values in the domains can change the OSP-implementability of optimal algorithms. Specifically, we show that our observation for shortest path

on the graph in Figure 1(a) is not an accident as for all the graphs where a path is a direct edge, we can design an optimum OSP mechanism no matter what the alternative path looks like. Similarly, we prove that there are no OSP optimal mechanisms for the graph in Figure 1(c) and all the graphs where the two paths are composed of the same number (larger than one) of edges. As for the graphs where neither path is direct and each has a different number of edges, the existence of an OSP mechanism returning the shortest path depends on the values in the domains. Given the simple setting considered for shortest paths, the mechanisms that we design are not very complex and it actually turns out that any implementation tree would lead to OSP optimal mechanisms, when these are feasible.

We then generalize the setting to any set system problem, wherein agents have three-value (heterogeneous) domains and fully characterize the properties needed to design OSP mechanisms.

> **Main Theorem about Set Systems (informal).** *There is an OSP optimal mechanism only if the set of feasible solutions are "aligned" with the agents' subdomains. If the agents have three-value domains, this is also sufficient. The OSP optimal mechanism, if any, combines ascending and descending auctions depending on the structure of the feasible solution set.*

The intuition behind the characterization is simple. From OSP CMON, we know that if an OSP mechanism selects an agent e when she has a "high" cost then it must select e when she has a "low" cost (akin to monotonicity for strategyproofness). Therefore, to design an OSP optimal mechanism we need to define an implementation tree which satisfies this property. At each node of the tree, the domain of the agents is restricted to a particular subdomain, depending on the particular history; in turn, the set of possible type profiles also shrinks. Hence, there may be solutions that are suboptimal for all type profiles in this set, and others that are still *alive* (i.e., optimal for at least one type profile in the set). When e is asked to separate a high cost from a low cost at node u of the tree, we need the alive solutions to be "aligned" for the subdomain at u, which roughly means that it should never be the case that there are two bid profiles in this subdomain for which e belongs to an optimal solution when she has a high cost and is not part of an optimal solution when she has a low cost.

The somehow surprising extra aspect is that even if the alive solutions were not aligned for one single subdomain then there would be no way to design an implementation tree to bypass this misalignment.

The technical definition of alignment has some nuisance to do with the particular ways in which the OSP monotonicity can be broken. On the positive side, however, alignment rather immediately suggests how to interleave ascending and descending phases to design an OSP optimal mechanism. Our characterization precisely shows a key difference between truthfulness and OSP. Whilst the former requires a monotonicity property between the solution computed on two instances — identical but for the lying agent cost — OSP mechanisms need a similar property to hold for a larger set of instances (encoded by agent subdomains at a particular history).

Moreover, this characterization also enables us to give a *testing algorithm*, running in time polynomial in the size of the set system instance, which decides whether an OSP optimal mechanism for the instance at hand is possible or not. This coupled with our mechanism gives a sort of automated mechanism design result in that the designer has a blackbox, comprised of a testing algorithm, and possibly our mechanism, to implement the optimal solution in an OSP way.

1.3 Related Works

The notion of OSP mechanism has been introduced recently by [33] and has received a lot of attention in the community. As mentioned above, the class of deferred-acceptance auctions [36] 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 [36]. [14] studies 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; [14, 29] provide a number of positive results where DA auctions are instead optimal. [24] 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 to apply it mainly to settings without money, namely matching and voting. In particular, revelation principles for OSP mechanisms are provided in [8, 39, 34]. In [8], following a similar characterization given in [5], it has been showed that every OSP mechanism can be implemented as a *gradual revelation mechanism*, in which at each time step an agent shrinks the set to which her type belongs, unless all possible outcomes are indifferent to her (when she fully reveals her type). In [39] OSP mechanisms are characterized as millipede games; in [34] they are showed equivalent to randomized round table mechanisms. The authors of [39, 45] define, among other results, stronger and weaker versions of OSP, respectively. Specific characterization of OSP mechanisms for special settings (without money) are instead provided in [43, 4, 3]. A couple of recent papers related to ours are [21], where among other settings the authors consider OSP mechanisms with money for machine scheduling, and [31], where this problem is studied in the setting without money. In particular, the lower bound for machine scheduling in [21] 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 (instead of their actual cost) whenever they overbid. Monitoring is also used in [31] to prove a tight bound for OSP mechanisms without money and a single task; the bound (asymptotically) matches the performances of strategyproof mechanisms. The use of verification [38] for OSP mechanisms is, instead, studied in [22]. The tradeoff between approximation guarantee (for machine scheduling) and relaxations of OSP is recently studied in [23].

Research in algorithmic mechanism design [42, 27, 11] 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 setting: from revenue maximization [7] to makespan minimization [17], for buyers with correlated valuations [1], or with complements [16], or for streams of buyers [12]. 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 – whilst being OSP – 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.

Another line of work [27, 15] focuses on extending to more complex settings the well-known result of [10]. This result states that the revenue-maximizing single item auction, i.e., the Myerson auction, performs worse than the "simpler" social welfare maximizing auction, i.e., the Vickrey auction, with one extra bidder. However, simplicity in this research is vaguely stated, since the focus is more on the complexity of designing a mechanism, than on the complexity of understanding

it.

Another paper relevant in the context of "simple" mechanisms is [25], which shares some of the motivation and philosophy behind OSP. Specifically, the authors notice that the implementation of a social choice function via a normal-form game is simpler/more obvious if there exists a "simple guide" that can always be used to discover the strategy combination which survives successive elimination of dominated strategies. They showed that backwards induction implementation via an extensive game is equivalent to such a "guided" implementation via a normal-form game.

CMON is a widely used technique in mechanism design that dates back to [40] – a general treatment is given in [35, 26]. This method has been used quite extensively to prove strategyproofness of mechanisms in the classical setting, cf., e.g., [9, 32] and when some form of verification can be adopted, see [44, 30]. Particularly relevant for our work is the research which shows that in order to establish strategyproofness it is sufficient to study cycles of length two as in [41].

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 applications, 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 $(f(\mathbf{b}), \mathbf{p}(\mathbf{b}))$, where $f(\mathbf{b})$ (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 $\mathbf{p}(\mathbf{b}) = (p_1(\mathbf{b}), \ldots, p_n(\mathbf{b})) \in \mathbb{R}^n$ maps the actions taken by the agents according to $\mathbf{b}$ to *payments* from the mechanism to the agents. We will sometimes refer only to f an $\mathbf{p}$ when the reference to $\mathbf{b}$ is clear from the context.

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 $X \in \mathcal{S}$, $u_i(t_i, X)$ 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., $u_i(t_i, \mathcal{M}(b_i, \mathbf{b}_{-i})) = p_i(b_i, \mathbf{b}_{-i}) - t_i(f(b_i, \mathbf{b}_{-i}))$.

For our applications, 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 $\mathbf{b}$.

2.1 Extensive-form Mechanisms and Obvious Strategyproofness

We now formally define the concept of obviously strategy-proof deterministic mechanisms. This concept has been introduced in [33]. However, our definition is built on the more accessible ones given by [5] and [21]. As shown in [8, 34], our definition is equivalent to Li's.[2]

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 $S(u) \in [n]$;
- 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

[2] More in detail, our definition of implementation tree is equivalent to the concept of round-table mechanism in [34]. 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.

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 $\mathcal{M}(\mathbf{b}) = X(\ell(\mathbf{b}))$, i.e., the mechanism returns exactly the outcome that labels the unique leaf ℓ reached by the mechanism when the agents take actions compatible with $\mathbf{b}$. 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 above, 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 of agent i to be consistent with a fixed type. In particular, she could play according to b_i at some node of the tree and a different b'_i later on at a subsequent history. This is because, 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 big as the utility in every other leaf of $\mathcal{T}_u$.

Recall that a mechanism $\mathcal{M}$ is *strategyproof* 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. Hence, if a mechanism is obviously strategyproof, then it is also strategyproof.

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 $b \in D_i(u)$.

Let us state two further properties of obvious strategyproofness, that turn out to be very useful in the rest of the paper. First, it is not hard to see that if $\mathcal{M}$ is OSP when the type profile is

taken from $D = \times_i D_i$, then it continues to be OSP even if the types are only allowed to be selected from $D' = D'_1 \times \cdots \times D'_n$, where $D'_i \subseteq D_i$. Let us define $\mathcal{M}'$ obtained from $\mathcal{M}$ by *pruning* the paths involving actions corresponding to types in $D \setminus D'$. If $\mathcal{M}$ is OSP, then also $\mathcal{M}'$ enjoys this property [33].

2.2 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) — t_i is the type of agent i. The set of feasible solutions $\mathcal{S}$ is here an allocation (also called schedule) of the m jobs to the n machines. We let $\mathbf{b}$, with $b_i \in D_i$, denote a generic instance of the machine processing times. The social choice function f maps its input $\mathbf{b}$ to a possible schedule $f(\mathbf{b}) = (f_1(\mathbf{b}), \ldots, f_n(\mathbf{b}))$, where $f_i(\mathbf{b})$ denotes the job load assigned to machine i. The cost that agent i faces for the schedule $f(\mathbf{b})$ when her type is b_i is $b_i(f(\mathbf{b})) = b_i \cdot f_i(\mathbf{b})$. We focus on social choice functions f^* returning a schedule that minimizes the *makespan* for every instance $\mathbf{b}$, i.e.,

$$f^*(\mathbf{b}) \in \arg\min_{\mathbf{x} \in \mathcal{S}} MS(\mathbf{x}, \mathbf{b}),$$

where $MS(f, \mathbf{b}) = \max_{i=1}^{n} b_i(f_i(\mathbf{b}))$ denotes the makespan of solution $f(\mathbf{b})$ according to machine processing times $\mathbf{b}$. We say that f is ρ-approximate if it returns a solution whose cost is at most ρ times the optimum, that is, for all instances $\mathbf{b}$, we have $MS(f(\mathbf{b}), \mathbf{b}) \leq \rho \cdot MS(f^*(\mathbf{b}), \mathbf{b})$.

2.3 Set Systems

In a *set system* $(E, \mathcal{F})$ we are given a set E of elements and a family $\mathcal{F} \subseteq 2^E$ of feasible subsets of E. Each element $i \in E$ is controlled by a selfish agent, that is, the cost for using i is known only to agent i and is equal to some non-negative value t_i. For a generic instance of agents' costs $\mathbf{b}$, with $b_i \in D_i$, the social choice function f selects a feasible subset in $\mathcal{F}$ (in other words, $\mathcal{F} = \mathcal{S}$ for set systems). We can use the same notation above for single-parameter agents with the restriction that $f_i(\mathbf{b}) \in \{0, 1\}$ to mean that the element controlled by agent i is either chosen by f, with $f_i(\mathbf{b}) = 1$, or not, with $f_i(\mathbf{b}) = 0$. Here our objective is social cost minimization, that is,

$$f^*(\mathbf{b}) \in \arg\min_{\mathbf{x} \in \mathcal{F}} SC(\mathbf{x}, \mathbf{b}),$$

where $SC(\mathbf{x}, \mathbf{b}) = \sum_{i=1}^{n} b_i(\mathbf{x})$ denotes the social cost of solution $\mathbf{x}$ according to element costs $\mathbf{b}$. Several problems on graphs, such as the path auction discussed above, can be cast in this framework.

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 (a_i, b_i)-*separating *for agent i if the following holds: Node u is labelled with i, i.e., $i = S(u)$; there are two profiles $(a_i, \mathbf{a}_{-i})$ and $(b_i, \mathbf{b}_{-i})$ 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 (a_i, b_i)-separating vertices for agent i as the agent may be asked to separate a_i from b_i 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 $a_i, b_i \in D_i$, $a_i \neq b_i$, for all vertices u that are (a_i, b_i)-separating for i, the following holds:*

$$p_i(b_i, \mathbf{b}_{-i}) - p_i(a_i, \mathbf{a}_{-i}) \leq a_i(f(b_i, \mathbf{b}_{-i})) - a_i(f(a_i, \mathbf{a}_{-i})) \qquad \text{for all } \mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u)\ . \tag{1}$$

Proof. According to our definition, a mechanism $\mathcal{M}$ with implementation tree $\mathcal{T}$ is an OSP mechanism for the social choice function f if and only if there exists a payment function $\mathbf{p}$ such that $\mathcal{M} = (f, \mathbf{p})$ and, for every agent i with real type t_i, and for every vertex u such that $i = S(u)$, it holds that

$$p_i(t_i, \mathbf{b}_{-i}) - t_i(f(t_i, \mathbf{b}_{-i})) = u_i(t_i, \mathcal{M}(t_i, \mathbf{b}_{-i})) \geq u_i(t_i, \mathcal{M}(b'_i, \mathbf{b}'_{-i})) = p_i(b'_i, \mathbf{b}'_{-i}) - t_i(f(b'_i, \mathbf{b}'_{-i}))$$

for every $\mathbf{b}_{-i}, \mathbf{b}'_{-i}$ 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. Since the true type of i can be any value in D_i, then the mechanism is OSP if and only if this is true for any pair $a_i, b_i \in D_i$. □

Henceforth, we will refer to condition (1) as *OSP constraint*. We next restate the set of conditions coming from all the OSP constraints 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 $e = ((a_i, \mathbf{a}_{-i}), (b_i, \mathbf{b}_{-i}))$ for every $a_i, b_i \in D_i$, $a_i \neq b_i$, and $\mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u)$, where u is an (a_i, b_i)-separating vertex of $\mathcal{T}$. The weight of the edge is $w(e) = a_i(f(b_i, \mathbf{b}_{-i})) - a_i(f(a_i, \mathbf{a}_{-i}))$.*

Definition 4 (OSP CMON)**.** *We say that the OSP cycle monotonicity (OSP CMON) property holds for mechanism $\mathcal{M}$ implemented with tree $\mathcal{T}$ and using social choice function f if, for all i, the OSP-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 choice function f on finite domains if and only if OSP CMON holds.*

The proof of the theorem follows standard arguments used to prove cycle monotonicity for the classical definition of strategyproofness. For completeness, we give a proof in appendix.

For our applications, it is useful to recast as follows the OSP CMON and OSP 2CMON for the case of single-parameter agents (for which, as stated above, $t_i(f(\mathbf{b})) = t_i \cdot f_i(\mathbf{b})$).

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

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

Note that the above OSP 2CMON condition holds if the following stronger condition holds:

$$f_i(a_i, \mathbf{a}_{-i}) \geq f_i(b_i, \mathbf{b}_{-i}) \qquad \text{for all } \mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i} \ , \tag{3}$$

in which case OSP 2CMON holds for *any* implementation tree $\mathcal{T}$.

We next show some preliminary results about machine scheduling and set systems. These results give a taster of the power and flexibility of OSP CMON as a tool in this context.

3.1 Warm-up I: Bounding the Approximation Guarantee for Scheduling

We next give a simple lower bound for the machine scheduling problem.

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 $M \geq 2L$ and $H \geq 2M$.*

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 $M \geq 2L$ and $H \geq 2M$, 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). Moreover, let j denote the other agent. We show that this mechanism cannot satisfy OSP 2CMON, thus reaching a contradiction.

If i diverges at u on M and H, then consider $\mathbf{b} = (b_i, b_j) = (H, H)$ and $\mathbf{a} = (a_i, a_j) = (M, L)$. Since the mechanism is better than 2-approximate, it must satisfy $f_i(b_i, b_j) = 1$ and $f_i(a_i, a_j) = 0$. 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 agent j, and therefore $H, L \in D_j(u)$ as required to invoke (2) with our choice $b_j = H$ and $a_j = L$.

If i diverges at u on L and M, then consider $\mathbf{a} = (a_i, a_j) = (L, L)$ and $\mathbf{b} = (b_i, b_j) = (M, H)$. Since the mechanism is better than 2-approximate, it must satisfy $f_i(a_i, a_j) = 1$ and $f_i(b_i, b_j) = 2$. Similarly to the previous case, this violates the OSP 2CMON condition (2).

Finally, if i diverges at u between L and H, it must necessarily be the case i either diverges on M and H (if M is not separated from L at u) or on L and M (otherwise). □

In Appendix B we give a complete characterization of the approximability for two jobs and two agents for three-value domains, showing that in this specific case the above bound is tight. 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 OSP optimal 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 $\sqrt{n}$.

3.2 Warm-up II: Characterizing OSP Mechanisms for Parallel Paths

Next we provide a characterization of OSP optimal mechanisms for the path auction problem discussed in the introduction: this is a special case of a set system problem where the set of feasible solutions is the set of all the paths between the source node s and the destination node t in a given graph G. Moreover, we consider the case in which G has two parallel paths from the source to the destination; the first is comprised of a set T of t edges, that we will sometimes call top edges,

whilst the second is comprised of a set B of b edges, that we will call bottom edges. Without loss of generality, we assume that $t \geq b$.

This provides an illustration of the approach that we will adopt below to prove one of our main results, namely an analytical and algorithmic characterization of the set systems that admit an OSP optimal mechanism.

Proposition 8. *There is an OSP optimal mechanism for the shortest path problem on parallel paths and two-value domains $D = \{L, H\}^n$ if and only if either (1) $b = 1$ or (2) $t > b > 1$ and $\frac{H}{L} \leq \frac{t-1}{b-1}$.*

Proof. Let us start by proving the sufficient condition. Consider the optimal mechanism that returns the bottom path in case of ties. The sufficiency is even stronger in that we will prove it no matter the implementation tree, thus including direct-revelation mechanisms. We indeed show that the stronger version of OSP 2CMON in (3) holds. (Note that it will be enough to prove OSP 2-cycle monotonicity since, as proved below in Theorem 9, this is sufficient for two-value domains in this setting.)

Specifically, we will prove that the hypotheses of the theorem imply the following:

(i) For any top edge e, if the corresponding agent reports H, then the bottom path is chosen, that is, $f_e(H, \mathbf{b}_{-e}) = 0$ for all $\mathbf{b}_{-e}$;

(ii) For any bottom edge e, if the corresponding agent reports L, then the bottom path is chosen, that is, $f_e(L, \mathbf{a}_{-e}) = 1$ for all $\mathbf{a}_{-e}$.

Since $f_e(\cdot)$ is either 0 or 1 for this problem, the two items above imply that (3) holds for every agent e.

Consider first the case $t \geq b = 1$. This case is trivial, since the bottom path costs at most H (which proves (i)), and when the only edge in the bottom path reports L, the upper path cannot be better because its cost is at least $tL \geq L$ (which proves (ii)).

Let us now consider the case $t > b > 1$ and $(b-1)H \leq (t-1)L$. If e is a top edge, then whenever the corresponding agent reports H, the top path costs at least $H + (t-1)L$ while the bottom path costs at most $Hb = H + (b-1)H \leq H + (t-1)L$ (which implies (i)). If e is a bottom edge, then whenever the corresponding agent reports L, the bottom path costs at most $L + (b-1)H \leq L + (t-1)L = tL$, and the top path costs at least tL (which implies (ii)).

We next prove the necessity and show that when either (1) $t = b > 1$ or (2) $t > b > 1$ and $\frac{H}{L} > \frac{t-1}{b-1}$, no optimal mechanism $\mathcal{M}$ can be OSP. Since $\mathcal{M}$ is optimal, it is not trivial and at some point it must separate L from H for at least one agent. We consider the first divergent agent e, and show that OSP 2CMON is violated for this agent, thus implying that mechanism $\mathcal{M}$ is *not* OSP (Theorem 5). Note that, since e is the first divergent agent, in the implementation tree $\mathcal{T}$, there is a unique LH-separating node u for agent e; moreover, $\mathbf{a}_{-e}, \mathbf{b}_{-e} \in D_{-e}(u)$ for every $\mathbf{a}, \mathbf{b} \in \{L, H\}^n$. Assume that e is a top edge (the proof is exactly the same for the case in which e is a bottom edge). For the case in which $t = b > 1$, it is enough to consider $\mathbf{a}_{-e}$ comprised of at least one H for the top edges and all L for all the bottom edges, so that $f_e(L, \mathbf{a}_{-e}) = 0$; For $\mathbf{b}_{-e}$ consider the bid vector where all the top edges have value L and at least two of the bottom edges have bid H, we have $f_e(H, \mathbf{b}_{-e}) = 1$. Therefore, the OSP 2CMON condition (2) is violated. When $t > b > 1$ and $\frac{H}{L} > \frac{t-1}{b-1}$, consider instead the bid vector $\mathbf{a}_{-e}$ where all top edges have cost H and all bottom edges have cost L, and the bid vector $\mathbf{b}_{-e}$ where all top edges have cost L and all bottom edges have cost H. Since $L + (t-1)H > bL$ then $f_e(L, \mathbf{a}_{-e}) = 0$, while as $H + (t-1)L < bH$ then $f_e(H, \mathbf{b}_{-e}) = 1$. Therefore, the OSP 2CMON condition (2) is violated also in this case. □

3.3 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 show that, for single parameter settings, these properties turn 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 9. *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.*

Proof. One direction follows from Theorem 5. As for the other direction, we prove that OSP 2CMON implies OSP CMON.

Fix an agent i and an implementation tree $\mathcal{T}$, and consider a cycle $C = (\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^{k+1})$, with $\mathbf{a}^{k+1} = \mathbf{a}^1$, in the graph $OSP_i^{(f,\mathcal{T})}$. Since we are in a one-parameter setting, the weight of each edge $(\mathbf{a}^j, \mathbf{a}^{j+1})$ is

$$w(\mathbf{a}^j, \mathbf{a}^{j+1}) = a_i^j(f_i(\mathbf{a}^{j+1})) - a_i^j(f_i(\mathbf{a}^j)) = a_i^j \cdot \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right)$$

and therefore the weight of C is

$$w(C) = \sum_{j=1}^{k} a_i^j \cdot \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right).$$

We now partition the edges between those with positive length and those with negative length, and observe the following:

1. For $\left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right) > 0$, OSP 2CMON (2) implies $a_i^j > a_i^{j+1}$ (since $a_i^j \neq a_i^{j+1}$ according to the definition of OSP-graph – Definition 3), and thus $a_i^j \geq M_i$ (since we consider three-value domains). In this case, $a_i^j \cdot \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right) \geq M_i \cdot \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right)$.
2. For $\left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right) < 0$, OSP 2CMON (2) implies $a_i^j < a_i^{j+1}$ (since $a_i^j \neq a_i^{j+1}$ according to the definition of OSP-graph – Definition 3), and thus $a_i^j \leq M_i$ (since we consider three-value domains). In this case, $a_i^j \cdot \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right) \geq M_i \cdot \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right)$.

We have thus shown that $w(C) \geq M_i \cdot \sum_{j=1}^{k} \left(f_i(\mathbf{a}^{j+1}) - f_i(\mathbf{a}^j)\right) = 0$, and thus OSP CMON holds. □

We next show that the above 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 10. *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 $j \neq i$.*

Proof. Let $D_1 = \{H, M, B, L\}$ and $D_j = \{H, M, L\}$ for every $j \neq 1$, with $H > M > B > L$. Consider a social function f aiming to select the agent i with the smallest type. This can be seen as a procurement auction for a single item, or, alternatively, if types are negative, as a classical auction setting for a single item, in which we would like to select the agent with the maximum valuation for the item.

Consider the mechanism in which we first run a *descending auction* until we are left with only two candidates, and then we run an *ascending auction* just between these two candidates to select the winner. This can be done by the implementation tree $\mathcal{T}$ corresponding to the following algorithm:

- *Descending auction*: keep a set of candidates, initialized to the set of all agents; for every type $t \in D_1$ considered in *decreasing* order, first ask agent 1 if her type is t, and then, if the answer is *no* repeat the question to the remaining agents (in order of their index); if an agent answers *yes*, then remove this agent from the set of candidates; this process continues until only two candidates are left;

- *Ascending auction*: for every type $t \in D_i$ considered in *increasing* order ask the candidates left whether their type is t, by giving precedence to agent 1 if she is still a candidate; if an agent answers *yes*, then assign the item to this agent.

It is not hard to see that a mechanism that follows this implementation tree satisfies OSP 2CMON. Indeed, during the descending phase, if the agent gives a positive answer, then she is never assigned the item. Contrarily, during the ascending phase, if the agent gives a positive answer, then she is always assigned the item. (For a more formal proof of OSP 2CMON, we refer the reader to the proof of Theorem 20, where this property is proved for a generalization of the mechanism above.)

We now consider a cycle of length four in the OSP-graph of agent 1 involving the profiles $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}$ defined as follows:

- in $\mathbf{x}$ we have $x_1 = H$ and $x_j < H$ for every $j \neq 1$;

- in $\mathbf{w}$ we have $w_1 = M$ and $w_j = H$ for every $j \neq 1$;

- in $\mathbf{z}$ we have $z_1 = L$, $z_2 = H$ and $z_j \geq M$ for every $j \neq 1, 2$;

- in $\mathbf{y}$ we have $y_1 = B$, $y_2 = H$ there is $k \neq 1, 2$ such that $y_k = L$, and $y_j \geq L$ for every $j \neq 1, 2, k$.

Observe that when agent 1 is queried about type H, then by answering *yes* profile $\mathbf{x}$ can occur, whereas $\mathbf{w}, \mathbf{z}, \mathbf{y}$ can all occur if agent 1 answers *no*. Hence, in the OSP-graph of agent 1, all edges $(\mathbf{x}, \mathbf{w})$, $(\mathbf{w}, \mathbf{x})$, $(\mathbf{x}, \mathbf{z})$, $(\mathbf{z}, \mathbf{x})$, $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{y}, \mathbf{x})$ exist. Moreover, if agent 1 is queried about type L, then by answering *yes* the outcome $\mathbf{z}$ can occur, whereas $\mathbf{w}$ and $\mathbf{y}$ can occur if agent 1 answers *no*. Hence, the OSP-graph of agent 1 also contains edges $(\mathbf{z}, \mathbf{w})$, $(\mathbf{w}, \mathbf{z})$, $(\mathbf{z}, \mathbf{y})$ and $(\mathbf{y}, \mathbf{z})$.

Observe that the algorithm assigns the item to agent 1 in $\mathbf{w}$ and $\mathbf{z}$, while agent 1 does not receive the item in $\mathbf{x}$ and $\mathbf{y}$. Hence, edges $(\mathbf{x}, \mathbf{y})$, $(\mathbf{y}, \mathbf{x})$, $(\mathbf{w}, \mathbf{z})$ and $(\mathbf{z}, \mathbf{w})$ have weight 0, and weights for other edges are as in Figure 2.

Finally note that the cycle $(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}, \mathbf{x})$ has weight $B - M < 0$, from which the claim follows. □

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. More specifically, we can compute the optimum for two-value domains, while approximation guarantee become "large" already for *three-value* domains.

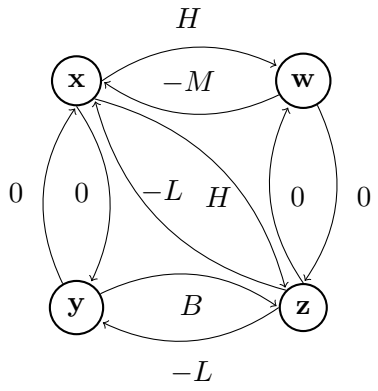

Figure 2: A negative weight cycle in the OSP-graph for agent 1 (proof of Theorem 10)

4.1 Two-value Domains

In this subsection, we show 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 11. *For the machine scheduling problem, there exists an OSP optimal mechanism that runs in polynomial-time for any number of agents with two-value domains* $D_i = \{L_i, H_i\}$.

The full proof of this theorem is given in Appendix B.2. Here we instead show a proof of the above theorem for the simpler setting in which domains are homogeneous, i.e., $D_i = \{H, L\}$ for every i. This case turns out to be interesting, since we can use an explicit, simple payment function P which rewards each agent an amount p, between L and H, for each unit of allocated work, i.e., $p_i(\mathbf{b}) = p \cdot f_i(\mathbf{b})$ for $L < p < H$. We call such a mechanism a mechanism with *proportional payments*. The main intuition behind these payments is that agents with low cost have positive utility, while those with high cost have a negative utility. In particular, agents with low cost aim to maximize their assigned work, and agents with high cost aim to minimize it. The following definition and theorem formalize this intuition.

Definition 12. *An algorithm* f *is* strongly monotone *for a two-value domain* $D_i = \{L, H\}$ *if* $f_i(L, \mathbf{b}_{-i}) \geq f_i(H, \mathbf{b}'_{-i})$ *for all* i *and for all* $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}$.

Theorem 13. *A direct-revelation mechanism* $\mathcal{M} = (f, \mathbf{p})$ *with the proportional payments is OSP for a two-value domain if and only if algorithm* f *is strongly monotone for the corresponding two-value domain.*

Proof. Since $p_i(\mathbf{b}) = pf_i(\mathbf{b})$, with $p > L$, and $f_i(L, \mathbf{b}_{-i}) \geq f_i(H, \mathbf{b}'_{-i})$ we have $u_i(L, \mathcal{M}(L, \mathbf{b}_{-i})) = (p - L) \cdot f_i(L, \mathbf{b}_{-i}) \geq (p - L) \cdot f_i(H, \mathbf{b}'_{-i})$. Moreover, since $p < H$, we have $u_i(H, \mathcal{M}(H, \mathbf{b}_{-i})) = (p - H) \cdot f_i(H, \mathbf{b}_{-i}) \geq (p - H) \cdot f_i(L, \mathbf{b}'_{-i})$. As these two conditions hold for all i and for all $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}$, we have shown that OSP holds. □

From this result we easily obtain an optimal OSP mechanisms for the scheduling problem.

Proposition 14. *For the machine scheduling problem, there exists an OSP optimal mechanism that runs in polynomial-time for any number of agents with two-value domains* $D_i = \{L, H\}$.

Proof. We show that there is an optimal allocation that is strongly monotone. Let us denote with $\mathbf{w}^{(\ell)} = (w_1^\ell, w_2^\ell, \ldots, w_n^\ell)$ some (suitably defined) optimal allocation for the case in which there are ℓ agents with type L and the remaining $n-\ell$ with type H, with these workloads in non-decreasing order (i.e., $w_1^{(\ell)} \geq w_2^{(\ell)} \geq \cdots \geq w_n^{(\ell)}$). Note that $\mathbf{w}^{(n)} = \mathbf{w}^{(0)}$ is the allocation when all types are the same (all L or all H). We also consider a fixed order π of the agents.

For a generic input $\mathbf{b}$ with ℓ types being L, we allocate jobs according to $\mathbf{w}^{(\ell)}$, following the given order π among the agents with the same type. That is, let π_L and π_H denote the restrictions of the order π to only agents that in $\mathbf{b}$ have type L and H, respectively. For example, if i is the machine with type L with smallest value of $\pi(i)$ we let $\pi_L(i) = 1$. Then, each agent i gets $w^{(\ell)}_{\pi_L(i)}$ if her type is L, and $w^{(\ell)}_{\ell+\pi_H(i)}$ otherwise. Note that, $\pi_L(i) \leq \pi(i) \leq \ell + \pi_H(i)$ for every ℓ and i.

Hence, for each i, each $\mathbf{b}_{-i}$, and each $\mathbf{b}'_{-i}$, we have that $f_i(L, \mathbf{b}_{-i}) = w_s^{(\ell)}$, where ℓ is the number of machines of type L in $(L, \mathbf{b}_{-i})$, and $s = \pi_L(i)$; similarly, $f_i(H, \mathbf{b}'_{-i}) = w_t^{(\ell')}$, where ℓ' is the number of machines of type L in $(H, \mathbf{b}'_{-i})$, and $t = \ell' + \pi_H(i) \geq s$. Then the allocation is strongly monotone if $w_s^{(\ell)} \geq w_t^{(\ell')}$ for every ℓ, ℓ', and every s and t as described above.

Suppose first that $\ell' \geq \ell$, and observe that in this case $\mathbf{w}^{(\ell)}$ can be achieved from $\mathbf{w}^{(\ell')}$ by moving jobs from machines of type H to machines of type L, and thus $w_s^{(\ell)} \geq w_s^{(\ell')} \geq w_t^{(\ell')}$, as desired. Similarly, if $\ell' < \ell$, $\mathbf{w}^{(\ell')}$ can be achieved from $\mathbf{w}^{(\ell)}$ by moving jobs from machines of type H to machines of type L, and thus $w_s^{(\ell)} \geq w_t^{(\ell)} \geq w_t^{(\ell')}$, as desired. □

4.2 Three-value Domains

In this section, we show that the problem becomes hard as soon as we have *three-value* domains. Our contributions are a *lower bound* of $\sqrt{n}$ on the approximation of any OSP mechanism, and a *matching upper bound*, that is, a pair of $O(\sqrt{n})$-approximate OSP mechanisms.

4.2.1 Lower Bound

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

Theorem 15. *For the machine scheduling problem, no OSP mechanism can be better than $\sqrt{n}$-approximate. This also holds for three-value domains $D_i = \{L, M, H\}$.*

In order to prove this lower bound, we consider $m = n = c^2$, for some $c > 1$, and a three-value domain $D_i = \{L, M, H\}$ such that

$$M \geq m \cdot L \qquad \text{and} \qquad H \geq m\sqrt{n} \cdot M. \tag{4}$$

First 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 16. *Any OSP mechanism $\mathcal{M}$ for machine scheduling which is k-approximate (with $k < \sqrt{n}$) on domains as in* (4) *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 player 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 were assigned to a machine $j \neq i$, then the cost of the mechanism would be at least $H \geq \sqrt{n} \cdot mM > k \cdot OPT(\mathbf{x})$ and the mechanism would not be k-approximate.

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

$$\sqrt{n} \left\lceil \frac{m}{n-i+1} \right\rceil \leq \sqrt{n} \left\lceil \frac{m}{\sqrt{n}} \right\rceil = \sqrt{n} \left\lceil \frac{n}{\sqrt{n}} \right\rceil = c \cdot c = n = m. \tag{5}$$

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

$$\sqrt{n} \left\lceil \frac{m}{n-i} \right\rceil \leq \sqrt{n} \left\lceil \frac{m}{\sqrt{n}} \right\rceil = \sqrt{n} \left\lceil \frac{n}{\sqrt{n}} \right\rceil = \sqrt{n} \cdot \sqrt{n} = n = m. \tag{6}$$

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. □

We are now ready to prove our lower bound.

Proof of Theorem 15. 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 16, 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 – observe that $n - 1 \geq |S| = n - i + 1 \geq \sqrt{n} + 1$. The 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 $\left\lceil \frac{n}{|S|} \right\rceil \geq 1$ 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 $\mathbf{y}$. Since $|S| \geq \sqrt{n} + 1 \geq 2$, there is at least one L in $\mathbf{y}$ and then giving even one job 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 (see Appendix B.4).

4.2.2 Upper Bound

We describe our mechanisms for a generic, not necessarily three-value, 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 $t^{out}_{\min}$, the best type of the slow machines found in the descending phase.

Theorem 17. *Mechanism $\mathcal{M}_{many}$ is OSP for any three-value domain $D_i = \{L_i, M_i, H_i\}$.*

Set $A = [n]$, and $t_i^{out} = \max\{d \in D_i\}$
while $|A| > \lceil \sqrt{n} \rceil$ **do**
    Set $q = \max_{a \in A}\{t_a^{out}\}$ and $i = \min\{a \in A \colon t_a^{out} = p\}$
    Ask machine i if her type is equal to q
    if *yes* **then** remove i from A
    else set $t_i^{out} = \max\{t \in D_i \colon t < q\}$
Set $t_{\min}^{out} = \min_{k \notin A} t_k^{out}$

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

Proof. We prove that $\mathcal{M}_{many}$ satisfies OSP 2CMON. The claim then follows from Theorem 9. 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 the value queried 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. Then, $y_i = p$, where p is the value queried 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 $(y_i, \hat{\mathbf{z}}_{-i})$, $\hat{\mathbf{z}}_{-i}$ being such that $\hat{z}_j = t_{\min}^{out}$ for every $j \in A \setminus \{i\}$. Observe that since u is a node of the ascending phase then for every $\mathbf{x}$ compatible with u, it must be the case that $x_j \geq y_i$ for every $j \in A$ (in other words, if there were a $j \in A$ with $x_j < y_i$ then $\mathbf{x}$ would not be compatible with u). 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 her speed decreases). □

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

Algorithm 2: Ascending Phase for $\mathcal{M}_{many}$

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

Algorithm 3: Ascending Phase for $\mathcal{M}_{few}$

The next theorem, whose proof can be found in appendix, bounds the approximation ratio of the mechanism.

Theorem 18. *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) computes the allocation in a different manner.

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 19. *If $m \leq \lceil \sqrt{n} \rceil^2$ then Algorithm 3 can be executed.*

Proof. We want to show that under the assumption on the number of jobs, we can execute Algorithm 3 and in particular we can always find a value for z in line 7. Let p denote a shorthand for $t^{out}_{\min}$. We next prove that it never occurs during the ascending phase that $\zeta \cdot q > \lceil \sqrt{n} \rceil \cdot p$. In particular, since $q \leq p$, this follows by proving that $\zeta \leq \lceil \sqrt{n} \rceil$. Indeed, for the first machine to reveal the type during the ascending phase, we have that $C = m \leq \lceil \sqrt{n} \rceil^2$, and, thus $\zeta \leq \lceil \sqrt{n} \rceil$, as desired.

Suppose instead that 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. We will show that these machines have cumulatively received at least a number of jobs $m' \geq |Q| \cdot \lceil \sqrt{n} \rceil$. Indeed, m' is minimized assigning ζ jobs to each machine in Q, that in turn corresponds to assigning $\lceil m/|A| \rceil$ jobs to the first $m \mod |A|$ machines in Q and $\lfloor m/|A| \rfloor$ to the remaining machines. Then $C = m - m' \leq (|A| - |Q|) \lceil \sqrt{n} \rceil$, and thus $\zeta \leq \lceil \sqrt{n} \rceil$, as desired. □

Theorem 20. *Mechanism $\mathcal{M}_{few}$ is OSP for three-value domains $D_i = \{L_i, M_i, H_i\}$.*

Proof. We prove that $\mathcal{M}_{few}$ satisfies the OSP 2CMON. The claim then follows from Theorem 9. 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 the value queried 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, $t^{out}_{\min}$ and q at that node. Observe that for every profile compatible with u, we have that 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, it must be the case that $y_i = q(u)$. 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 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. □

The proof of the approximation guarantee is deferred to the appendix.

Theorem 21. *Mechanism $\mathcal{M}_{few}$ is $\lceil\sqrt{n}\rceil$-approximate for $m \leq \lceil\sqrt{n}\rceil^2$.*

5 Set Systems

In this section we characterize when OSP optimal mechanisms exist for set systems. We will formally define the concept of alignment introduced above, with a different and more technical terminology. The main message is that the feasibility of OSP optimal mechanisms depends on structural properties of the feasible solutions *and* the values in the agents' domains.

5.1 Key Concepts

Let us first introduce the key concepts. Consider a set system problem $(E, \mathcal{F}, \mathbf{D})$. where $\mathbf{D} = (D_e)_{e\in E}$ denotes the domain. We assume that this triple is the input to our mechanism design problem; the implementation tree will query the agents in order to elicit the input instance $\mathbf{b}$ from $\mathbf{D}$ and return an optimal solution for $\mathbf{b}$ whilst guaranteeing OSP.

We next define some useful concepts and notation, to state our characterization and mechanism. Consider an arbitrary *subdomain* $\tilde{\mathbf{D}}$ of $\mathbf{D}$, that is, a type domain $\tilde{\mathbf{D}} = (\tilde{D}_e)_{e\in E}$ such that $\tilde{D}_e \subseteq D_e$ for all $e \in E$. We denote by $L(e, \tilde{\mathbf{D}}) = \min\{t \in \tilde{D}_e\}$ and $H(e, \tilde{\mathbf{D}}) = \max\{t \in \tilde{D}_e\}$ the lowest and the highest type for e according to the subdomain $\tilde{\mathbf{D}}$. Similarly, for any $P \subseteq E$, we let $L(P, \tilde{\mathbf{D}})$ and $H(P, \tilde{\mathbf{D}})$ be the lowest and the highest possible cost of P according to subdomain $\tilde{\mathbf{D}}$, i.e.,

$$L(P, \tilde{\mathbf{D}}) = \sum_{e\in P} L(e, \tilde{\mathbf{D}}) \qquad \text{and} \qquad H(P, \tilde{\mathbf{D}}) = \sum_{e\in P} H(e, \tilde{\mathbf{D}}).$$

(When clear from the context, we omit the reference to $\tilde{\mathbf{D}}$ in these notations.) Finally, we let $\prec$ denote a total order among the feasible solutions in $\mathcal{F}$; this order will be used to select the optimal solution to return in case of ties; in other words, our mechanism will use a fixed tie-breaking rule. We remark that we will build a suitable implementation tree based on $\prec$.

We now introduce some concepts, along with some properties of theirs, that relate implementation trees of an extensive-form mechanism with the optimality of the solutions.

Definition 22 (Selectable Solution)**.** *A feasible solution $P \in \mathcal{F}$ is said to be* selectable *for a subdomain $\tilde{\mathbf{D}}$ if for every other $P' \in \mathcal{F}$ it holds that*

$$L(P \setminus P', \tilde{\mathbf{D}}) < H(P' \setminus P, \tilde{\mathbf{D}})$$

or

$$L(P \setminus P', \tilde{\mathbf{D}}) = H(P' \setminus P, \tilde{\mathbf{D}}) \text{ and } P \prec P'.$$

Any implementation tree gradually shrinks $\mathbf{D}$ to subdomains $\tilde{\mathbf{D}}$ by querying the agents. If the implementation tree has already shrunk the domain to some $\tilde{\mathbf{D}}$, a selectable solution for $\tilde{\mathbf{D}}$ cannot be excluded a priori because, for some profile in $\tilde{\mathbf{D}}$, it is either the unique optimum or the optimum preferred according to the tie-breaking rule. Observe that there is at least one selectable solution for every subdomain $\tilde{\mathbf{D}}$. To see this consider the solution P which is optimum (according to $\prec$) for the instance in which every agent e has type $L(e, \tilde{\mathbf{D}})$. For every $P' \in \mathcal{F}$, we then have

$$\begin{aligned} L(P \setminus P', \tilde{\mathbf{D}}) + L(P \cap P', \tilde{\mathbf{D}}) &\leq L(P' \setminus P, \tilde{\mathbf{D}}) + L(P \cap P', \tilde{\mathbf{D}}) \\ &\leq H(P' \setminus P, \tilde{\mathbf{D}}) + L(P \cap P', \tilde{\mathbf{D}}). \end{aligned}$$

While the above concept refers only to implementation tree and optimality, the next concept will turn out to be useful to study when there is a way to shrink $\mathbf{D}$ (i.e., an implementation tree) that returns an optimal solution but also that is compatible with OSP.

Definition 23 (Strongly Selectable Solution)**.** *A selectable solution P is said to be* strongly selectable *for a subdomain $\tilde{\mathbf{D}}$ if, for all $e \in P$, it continues to be selectable even for the subdomain $(\tilde{\mathbf{D}}_{-e}, H(e, \tilde{\mathbf{D}}))$, where $\tilde{\mathbf{D}}_{-f} = (\tilde{D}_e)_{e \neq f}$ and, with a slight abuse of notation, $H(e, \tilde{\mathbf{D}})$ denotes $\{H(e, \tilde{\mathbf{D}})\}$.*

In words, this means that solution P is still potentially optimum also when any one of its elements has the largest possible cost $H(e, \tilde{\mathbf{D}})$ in the subdomain $\tilde{\mathbf{D}}$ under consideration.

Example 24 (Path auctions, selectable vs strongly selectable solutions)**.** *To understand the difference between selectable and strongly selectable solutions, consider path auction on the graph in Figure 1(a) for $\tilde{D}_e = \{L, H\}$, with $2L < H$, for all e. Both solutions are selectable for this subdomain $\tilde{\mathbf{D}}$: the bottom path because $L < 2H$ and the top path since $2L < H$. However, only the bottom path is strongly selectable since the top path cannot be the optimum as soon as one of its edges has cost H. On the contrary, for the graph in Figure 1(c), both paths are strongly selectable since, for either path, we can always set the cost of the alternative solution to $2H > H + L$.*

5.2 The Analytical Characterization: Necessary Conditions

Our next two lemmas identify *necessary* conditions for the implementation tree of an OSP optimal mechanism for set systems. To this aim, we define the *obstacle domain set* $\mathcal{X}$ to contain $\mathbf{D}$ and, for each f with $|D_f| > 2$, $\tilde{\mathbf{D}}_f^{\top} = (\mathbf{D}_{-f}, \tilde{D}_f^{\top})$, where $\tilde{D}_f^{\top} = D_f \setminus L(f, \mathbf{D})$, and $\tilde{\mathbf{D}}_f^{\perp} = (\mathbf{D}_{-f}, \tilde{D}_f^{\perp})$, where $\tilde{D}_f^{\perp} = D_f \setminus H(f, \mathbf{D})$.

The first necessary condition roughly says that if there is a domain in the obstacle domain set $\mathcal{X}$ where elements of strongly selectable solutions can be excluded when they reveal their type to be as low as possible, then there is no implementation tree which yields an OSP optimal mechanism.

Lemma 25. *There is no OSP optimal mechanism for a set system problem if there is a domain $\tilde{\boldsymbol{D}} \in \mathcal{X}$ such that the following properties are both satisfied:*

(i) the set $\mathcal{S}$ of strongly selectable solutions for $\tilde{\boldsymbol{D}}$ contains at least one P with $f \in P$ such that $|\tilde{D}_f| > 1$;

(ii) for every $P \in \mathcal{S}$ and every $f \in P$ such that $|\tilde{D}_f| > 1$, there is $\bar{P}_f \in \mathcal{S}$ with $f \notin \bar{P}_f$ such that $\bar{P}_f$ remains selectable even for $(\tilde{\boldsymbol{D}}_{-f}, L(f, \tilde{\boldsymbol{D}}))$.

Proof. Assume by contradiction that there is a domain $\tilde{\mathbf{D}} \in \mathcal{X}$ for which the conditions above are satisfied and yet there is an optimal OSP mechanism $\mathcal{M}$; let us denote with $\mathcal{T}$ its implementation tree.

Let $\mathcal{S}$ be the set of strongly selectable solutions defined in the statement, which is not empty by hypothesis. Consider the first node $u \in \mathcal{T}$ in which an agent $f \in \bigcup_{P \in \mathcal{S}} P$ diverges between $L(f, \tilde{\mathbf{D}})$ and $H(f, \tilde{\mathbf{D}})$ in the subtree compatible with the type of every agent $e \in \bigcup_{P \in \mathcal{S}} P$ being in $\tilde{D}_e$ and the type of every remaining agent e being $H(e, \tilde{\mathbf{D}})$. We first argue that, since the mechanism $\mathcal{M}$ is optimal, such a node u must exist.

Consider the following two bid profiles $\mathbf{b}$ and $\bar{\mathbf{b}}$ in $\tilde{\mathbf{D}}$ defined as follows:

$$b_e = \begin{cases} L(e, \tilde{\mathbf{D}}) & \text{if } e \in P \\ H(e, \tilde{\mathbf{D}}) & \text{if } e \notin P \end{cases} \qquad \text{and} \qquad \bar{b}_e = \begin{cases} L(e, \tilde{\mathbf{D}}) & \text{if } e \in \bar{P}_f \\ H(e, \tilde{\mathbf{D}}) & \text{if } e \notin \bar{P}_f \end{cases}$$

for some strongly selectable solution P. Since both P and $\bar{P}_f$ are (strongly) selectable for $\tilde{\mathbf{D}}$, Definition 22 says that the mechanism must return P on input $\mathbf{b}$ and $\bar{P}_f$ on input $\bar{\mathbf{b}}$. Therefore, $P = \mathcal{M}(\mathbf{b}) \neq \mathcal{M}(\bar{\mathbf{b}}) = \bar{P}_f$. Since $\mathbf{b}$ and $\bar{\mathbf{b}}$ differ only in the bids of agents e in $P \cup \bar{P}_f$, there must be some node $u \in \mathcal{T}$ in which some agent $f \in P \cup \bar{P}_f$ diverges between $L(f, \tilde{\mathbf{D}})$ and $H(f, \tilde{\mathbf{D}})$.

Given the existence of u and f as above, we now apply the hypothesis (ii) as follows to show a negative OSP 2-cycle. For the strongly selectable solution P^* containing f there is another solution Q^* which does not contain f, and Q^* is also selectable for $(\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$. We next define two profiles $\mathbf{b}^{(H)}$ and $\mathbf{b}^{(L)}$ where f has high and low cost, respectively, and it is selected only for the high cost:

$$b_e^{(H)} = \begin{cases} H(f, \tilde{\mathbf{D}}) & \text{for } e = f \\ L(e, \tilde{\mathbf{D}}) & \text{for } e \neq f \text{ and } e \in P^* \\ H(e, \tilde{\mathbf{D}}) & \text{otherwise} \end{cases} \qquad b_e^{(L)} = \begin{cases} L(f, \tilde{\mathbf{D}}) & \text{for } e = f \\ L(e, \tilde{\mathbf{D}}) & \text{for } e \in Q^* \text{ (NB } e \neq f) \\ H(e, \tilde{\mathbf{D}}) & \text{otherwise.} \end{cases} \tag{7}$$

It is not hard to see that $\mathbf{b}^{(H)}$ and $\mathbf{b}^{(L)}$ are compatible with node u, and thus the cycle between $\mathbf{b}^{(H)}$ and $\mathbf{b}^{(L)}$ exists. Since P^* is strongly selectable, and Q^* is selectable for $(\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$, from (7) we have $\mathcal{M}(\mathbf{b}^{(H)}) = P^*$ and $\mathcal{M}(\mathbf{b}^{(L)}) = Q^*$. Therefore, the cycle between $\mathbf{b}^{(H)}$ and $\mathbf{b}^{(L)}$ is negative. □

The second necessary property regards domains $\tilde{\mathbf{D}} \in \mathcal{X}$ for which there are solutions that are selectable but *not* strongly selectable. For each such solution P there is an agent w, that we will call the *witness* of P, such that P is no longer selectable for $\tilde{\mathbf{D}}_{hw} = (\tilde{\mathbf{D}}_{-w}, H(w, \tilde{\mathbf{D}}))$. A couple of easy properties of witnesses are summarized in the next observation.

Observation 26. *Let P be a solution that is selectable but not strongly selectable for a subdomain $\tilde{\boldsymbol{D}}$ and let w be a witness of P. Then, we have, (i) $|\tilde{D}_w| \geq 2$; (ii) there is $P' \in \mathcal{F}$ such that $w \in P \setminus P'$ and either $H(w) + L(P \setminus (P' \cup \{w\})) > H(P' \setminus P)$ or $H(w) + L(P \setminus (P' \cup \{w\})) = H(P' \setminus P)$ and $P' \prec P$.*

Proof. The first observation is easy as if $|\tilde{D}_w| = 1$ then P would be selectable also for $\tilde{\mathbf{D}}_{hw}$. Similarly, if $w \notin P \setminus P'$ for every $P' \in \mathcal{F}$, then the conditions of Definition 22 would be satisfied for $\tilde{\mathbf{D}}_{hw}$ since they are for $\tilde{\mathbf{D}}$. But then, since P is not selectable for $\tilde{\mathbf{D}}_{hw}$, there is a feasible P' such that $w \in P \setminus P'$ for which either

$$H(w) + L(P \setminus (P' \cup \{w\})) > H(P' \setminus P)$$

or

$$H(w) + L(P \setminus (P' \cup \{w\})) = H(P' \setminus P) \text{ and } P' \prec P. \qquad \square$$

The next lemma intuitively says that, if there exist domains where witnesses of solutions that are selectable but not strongly selectable can be excluded (included, respectively) when they reveal their type to be the lowest (highest, respectively) possible, then there is no implementation tree which yields an OSP optimal mechanism.

Lemma 27. *There is no OSP optimal mechanism for a set system problem if there is a domain $\tilde{\boldsymbol{D}} \in \mathcal{X}$ such that the following properties are both satisfied:*

(i) the set $\mathcal{S}$ of selectable solutions for $\tilde{\boldsymbol{D}}$ has size $|\mathcal{S}| \geq 2$, and there is at least one $P \in \mathcal{S}$ such that P is not strongly selectable;

(ii) for every f for which there is at least one selectable solution to which it belongs and at least one selectable to which it does not belong (i.e., $f \in \bigcup_{(P,P') \in \mathcal{S} \times \mathcal{S}} P \setminus P'$) both the following are true:

- *there is $\bar{P}_f \in \mathcal{S}$ s.t. $f \notin \bar{P}_f$ and $\bar{P}_f$ is selectable for $\bar{\boldsymbol{D}} = (\tilde{\boldsymbol{D}}_{-f}, L(f, \tilde{\boldsymbol{D}}))$;*
- *there is $\check{P}_f \in \mathcal{S}$ s.t. $f \in \check{P}_f$ and $\check{P}_f$ is selectable for $\check{\boldsymbol{D}} = (\tilde{\boldsymbol{D}}_{-f}, H(f, \tilde{\boldsymbol{D}}))$.*

Proof. Assume by contradiction that there is a domain $\tilde{\mathbf{D}} \in \mathcal{X}$ for which the conditions above are satisfied and yet there is an optimal OSP mechanism $\mathcal{M}$; let us denote with $\mathcal{T}$ its implementation tree.

Let $\bar{\mathcal{S}} \subseteq \mathcal{S}$ be the subset of solutions that are not strongly selectable, and let $W(\bar{\mathcal{S}})$ the set of agents f such that f is a witness for some $P \in \bar{\mathcal{S}}$. Consider the first node $u \in \mathcal{T}$ in which $f \in \bigcup_{(P,P') \in \mathcal{S} \times \mathcal{S}} P \setminus P'$ diverges between $L(f, \tilde{\mathbf{D}})$ and $H(f, \tilde{\mathbf{D}})$ in the subtree compatible with the type of every agent $e \in \bigcup_{(P,P') \in \mathcal{S} \times \mathcal{S}} P \setminus P'$ belonging to $\tilde{D}_e$ and the type of every remaining agent e being $H(e, \tilde{\mathbf{D}})$.

We next prove that since the mechanism $\mathcal{M}$ is optimal, then such a node must exist. Let us begin by observing that since there is at least one solution that is not strongly selectable, then there is at least one witness w who can diverge at u since from Observation 26, $w \in \bigcup_{(P,P') \in \mathcal{S} \times \mathcal{S}} P \setminus P'$ and $|\tilde{D}_w| \geq 2$. Let, now $P \in \mathcal{S}$ be a solution that is not strongly selectable, let w be one of its

witnesses, and let P' be the solution whose existence is guaranteed by Observation 26. Consider the following two bid profiles $\mathbf{b}$ and $\bar{\mathbf{b}}$ in $\tilde{\mathbf{D}}$ defined as

$$b_e = \begin{cases} L(e, \tilde{\mathbf{D}}) & \text{if } e \in P \setminus P' \\ H(e, \tilde{\mathbf{D}}) & \text{otherwise} \end{cases} \qquad \text{and} \qquad \bar{b}_e = \begin{cases} L(e, \tilde{\mathbf{D}}) & \text{if } e \in P \setminus P' \text{ and } e \neq w \\ H(e, \tilde{\mathbf{D}}) & \text{otherwise} \end{cases}$$

Since P is selectable then $\mathcal{M}$ returns P on input profile $\mathbf{b}$ and P' on input profile $\bar{\mathbf{b}}$, since P is not strongly selectable. Therefore $P = \mathcal{M}(\mathbf{b}) \neq \mathcal{M}(\bar{\mathbf{b}}) = P'$. Since $\mathbf{b}$ and $\bar{\mathbf{b}}$ differ only in the bids of agent $w \in W(\bar{\mathcal{S}}) \subseteq \bigcup_{(P,P') \in \mathcal{S} \times \mathcal{S}} P \setminus P'$, there must be some node $u \in \mathcal{T}$ in which w diverges between $L(w, \tilde{\mathbf{D}})$ and $H(w, \tilde{\mathbf{D}})$.

Now let f be the agent that diverge at u and let $\bar{P}_f$ and $\check{P}_f$ as in the claim. We are going to show a negative OSP 2-cycle for f. Consider profiles $\mathbf{b}^{(L)}$ and $\mathbf{b}^{(H)}$ as follows:

$$b_e^{(H)} = \begin{cases} H(f, \tilde{\mathbf{D}}) & \text{for } e = f \\ L(e, \tilde{\mathbf{D}}) & \text{for } e \in \check{P}_f \setminus \bar{P}_f \text{ and } e \neq f \\ H(e, \tilde{\mathbf{D}}) & \text{otherwise} \end{cases} \qquad b_e^{(L)} = \begin{cases} L(f, \tilde{\mathbf{D}}) & \text{for } e = f \\ L(e, \tilde{\mathbf{D}}) & \text{for } e \in \bar{P}_f \setminus \check{P}_f \\ H(e, \tilde{\mathbf{D}}) & \text{otherwise} \end{cases}$$

It is immediate to see that both $\mathbf{b}^{(L)}$ and $\mathbf{b}^{(H)}$ are compatible with node u. However, in $\mathbf{b}^{(L)}$, where agent f has type $L(f, \tilde{\mathbf{D}})$, an optimal mechanism must select $\bar{P}_f$ (and thus it does not select f), since it is selectable even if the type of f is $L(f, \tilde{\mathbf{D}})$. In $\mathbf{b}^{(H)}$, where agent f has type $H(f, \tilde{\mathbf{D}}) > L(f, \tilde{\mathbf{D}})$, instead mechanism $\mathcal{M}$ selects $\check{P}_f$ (and thus f), since it is selectable even if the type of f is $H(f, \tilde{\mathbf{D}})$. Therefore, the cycle between $\mathbf{b}^{(L)}$ and $\mathbf{b}^{(H)}$ is negative. □

5.3 The Analytical Characterization: the Mechanism

The two necessary conditions suggest that it is possible to design an OSP optimal mechanism when both the following properties are satisfied for *every* subdomain $\tilde{\mathbf{D}} \in \mathcal{X}$ containing more than one instance. When all selectable solutions are also strongly selectable then there is an f such that every P' with $f \notin P'$ ceases to be selectable if the type of f is $L(f, \tilde{\mathbf{D}})$ (this is the negation of Lemma 25). Moreover, if there is at least one selectable solution that is not strongly selectable for $\tilde{\mathbf{D}}$, then there is f such that either every P' with $f \notin P'$ ceases to be selectable if the type of f is $L(f, \tilde{\mathbf{D}})$, or every P' with $f \in P'$ ceases to be selectable if the type of f is $H(f, \tilde{\mathbf{D}})$ (this is the negation of Lemma 27). We prove that these properties are indeed sufficient by proving that Algorithm 4 can be augmented with payments to give rise to an OSP optimal mechanism, that we call $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$, for set systems with *three-value domains*, i.e., with $D_e \subseteq \{L_e, M_e, H_e\}$ with $L_e < M_e < H_e$ for every e.

Theorem 28. *There is an OSP optimal mechanism for a set system problem with three-value domains if and only if there is no domain $\tilde{\mathbf{D}} \in \mathcal{X}$ for which the conditions of Lemma 25 or of Lemma 27 hold.*

The "only if" direction follows from Lemma 25 and Lemma 27.

For the "if" direction, we need to argue that if domains in $\mathcal{X}$ do not satisfy the conditions of Lemma 25 and Lemma 27 then no subdomain of $\mathbf{D}$ can satisfy these conditions.

Let us start by stating a key property of selectable solutions. Since the implementation tree may shrink the current subdomain $\tilde{\mathbf{D}}$ even further, it is useful to relate the optimality for the bigger domain with the optimality for smaller domains. The next lemma shows that an optimal mechanism can "forget" about solutions that are *not* selectable for the current subdomain $\tilde{\mathbf{D}}$.

Input: $E, \mathcal{F}, \mathbf{D}$
Output: A solution in $\mathcal{F}$ of minimum social cost

Initialize $R = \{P \in \mathcal{F}\colon P \text{ not selectable for } \mathbf{D}\}$, $\mathcal{P} = \mathcal{F} \setminus R$ and $\tilde{\mathbf{D}} = \mathbf{D}$
while $|R| < |\mathcal{F}| - 1$ **do**
  while *there is* $P \in \mathcal{P}$ *that is not strongly selectable for* $\tilde{\boldsymbol{D}}$ **do**
    if $\exists f \in \bigcup_{P\in\mathcal{P}} P$ *s.t. every* $P \in \mathcal{P}$*, with* $f \notin P$*, is not selectable for* $(\tilde{\boldsymbol{D}}_{-f}, L(f, \tilde{\boldsymbol{D}}))$ **then**
      Ask f if her type is $L(f, \tilde{\mathbf{D}})$
      if *yes* **then**
        $\tilde{\mathbf{D}} = (\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$
        Add to R and remove from $\mathcal{P}$ every P not selectable for $\tilde{\mathbf{D}}$
      else
        $\tilde{\mathbf{D}} = (\tilde{\mathbf{D}}_{-f}, D_f \setminus L(f, \tilde{\mathbf{D}}))$
        Add to R and remove from $\mathcal{P}$ every P not selectable for $\tilde{\mathbf{D}}$
    else
      Pick $f \in \bigcup_{P\in\mathcal{P}} P$ s.t. all $P \in \mathcal{P}$, with $f \in P$, are not selectable for $(\tilde{\mathbf{D}}_{-f}, H(f, \tilde{\mathbf{D}}))$
      Ask f if her type is $H(f, \tilde{\mathbf{D}})$
      if *yes* **then**
        $\tilde{\mathbf{D}} = (\tilde{\mathbf{D}}_{-f}, H(f, \tilde{\mathbf{D}}))$
        Add to R and remove from $\mathcal{P}$ every P not selectable for $\tilde{\mathbf{D}}$
      else
        $\tilde{\mathbf{D}} = (\tilde{\mathbf{D}}_{-f}, D_f \setminus H(f, \tilde{\mathbf{D}}))$
        Add to R and remove from $\mathcal{P}$ every P not selectable for $\tilde{\mathbf{D}}$
  if $|R| < |\mathcal{F}| - 1$ **then**
    Pick $f \in \bigcup_{P\in\mathcal{P}} P$ s.t. every $P \in \mathcal{P}$, with $f \notin P$, are not selectable for $(\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$
    Ask f if her type is $L(f, \tilde{\mathbf{D}})$
    if *yes* **then**
      $\tilde{\mathbf{D}} = (\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$
      Add to R and remove from $\mathcal{P}$ every P that is not selectable for $\tilde{\mathbf{D}}$
    else
      $\tilde{\mathbf{D}} = (\tilde{\mathbf{D}}_{-f}, D_f \setminus L(f, \tilde{\mathbf{D}}))$
      Add to R and remove from $\mathcal{P}$ every P that is not selectable for $\tilde{\mathbf{D}}$
Return the only solution in $\mathcal{P}$

Algorithm 4: The implementation tree of the optimal algorithm for mechanism $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$

Lemma 29. *If P is not selectable for $\tilde{\boldsymbol{D}}$, then it is not selectable for every subdomain $\tilde{\boldsymbol{D}}'$ of $\tilde{\boldsymbol{D}}$.*

Moreover, if $H(e, \tilde{\boldsymbol{D}}') = H(e, \tilde{\boldsymbol{D}})$ for all $e \in E$, then if P is not strongly selectable for $\tilde{\boldsymbol{D}}$, then it is not strongly selectable for $\tilde{\boldsymbol{D}}'$.

Proof. By contradiction, assume that P is selectable for $\tilde{\mathbf{D}}'$ and not selectable for $\tilde{\mathbf{D}}$. Since $\tilde{D}'_e \subseteq \tilde{D}_e$ for every $e \in E$, we have $L(e, \tilde{\mathbf{D}}') \geq L(e, \tilde{\mathbf{D}})$ and $H(e, \tilde{\mathbf{D}}') \leq H(e, \tilde{\mathbf{D}})$. Since P is selectable for $\tilde{\mathbf{D}}'$, we have that for every other $P' \in \mathcal{F}$

$$L(P \setminus P', \tilde{\mathbf{D}}) \leq L(P \setminus P', \tilde{\mathbf{D}}') \leq H(P' \setminus P, \tilde{\mathbf{D}}') \leq H(P' \setminus P, \tilde{\mathbf{D}})$$

where the second inequality is either strict, or it holds with equality together with $P \prec P'$ (cf. Definition 22). This means that P is selectable for $\tilde{\mathbf{D}}$, thus a contradiction.

Assume now that $H(e, \tilde{\mathbf{D}}') = H(e, \tilde{\mathbf{D}})$ for all $e \in E$ and assume by contradiction that P is strongly selectable for $\tilde{\mathbf{D}}'$ but not for $\tilde{\mathbf{D}}$. Similarly to the proof above, for $f \in P$ we then have

$$L(P \setminus (P' \cup \{f\}), \tilde{\mathbf{D}}) + H(f, \tilde{\mathbf{D}}) \leq L(P \setminus (P' \cup \{f\}), \tilde{\mathbf{D}}') + H(f, \tilde{\mathbf{D}}') \leq H(P' \setminus P, \tilde{\mathbf{D}}') = H(P' \setminus P, \tilde{\mathbf{D}}),$$

implying that P is strongly selectable also for $\tilde{\mathbf{D}}$, a contradiction. □

We continue be proving that for any subdomain where divergences are still possible, we can define an obstacle domain with convenient properties.

Lemma 30. *For each subdomain $\hat{\boldsymbol{D}}$ of $\boldsymbol{D}$ for which there is f such that $|\hat{D}_f| > 1$, there is an obstacle domain $\tilde{\boldsymbol{D}}$ such that for all $e \in E$, it holds*

$$L(e, \hat{\boldsymbol{D}}) = L(e, \tilde{\boldsymbol{D}}) \tag{8}$$

$$H(e, \hat{\boldsymbol{D}}) = H(e, \tilde{\boldsymbol{D}}) \tag{9}$$

$$\hat{D}_e \subseteq \tilde{D}_e \tag{10}$$

$$|\tilde{D}_f| > 1. \tag{11}$$

Proof. Let us define $\tilde{\mathbf{D}}$ as follows. We let $\tilde{\mathbf{D}} = (\mathbf{D}_{-f}, D_f \setminus \{L(f, \mathbf{D})\})$ if $L(f, \mathbf{D}) \notin \hat{D}_f$, $\tilde{\mathbf{D}} = (\mathbf{D}_{-f}, D_f \setminus \{H(f, \mathbf{D})\})$ if $H(f, \mathbf{D}) \notin \hat{D}_f$, and $\tilde{\mathbf{D}} = \mathbf{D}$ otherwise. We now argue that $\tilde{\mathbf{D}} \in \mathcal{X}$.

Recall that $|D_f| \geq |\hat{D}_f| > 1$; now, if $|D_f| = 2$ then $\hat{D}_f = \{L(f, \mathbf{D}), H(f, \mathbf{D})\}$ and then $\tilde{\mathbf{D}} = \mathbf{D} \in \mathcal{X}$. If, instead, $|D_f| = 3$ then either $\tilde{\mathbf{D}} = \mathbf{D}$ or $\tilde{\mathbf{D}} = \tilde{\mathbf{D}}_f^{\top}$ or $\tilde{\mathbf{D}} = \tilde{\mathbf{D}}_f^{\perp}$ depending on $\hat{D}_f$; in all cases, we can conclude that $\tilde{\mathbf{D}}$ is an obstacle domain. Finally, by inspection, we can observe that (8), (9), (10) and (11) are satisfied. □

The next two lemmas show that the restriction to obstacle domains in Lemma 25 and Lemma 27 is without loss of generality.

Lemma 31. *If the properties of Lemma 25 are not satisfied for every domain $\tilde{\boldsymbol{D}} \in \mathcal{X}$, then they are not satisfied for every subdomain $\hat{\boldsymbol{D}}$ of $\boldsymbol{D}$.*

Proof. The claim is obvious if in $\hat{\mathbf{D}}$ there is no strongly selectable solution P such that there is $f \in P$ with $|\hat{D}_f| > 1$ or for every pair of strongly selectable solutions P and $\bar{P}$ it holds that $|\hat{D}_f| = 1$ for every $f \notin P \cap \bar{P}$.

Consider then the case that there are two solutions P and $\bar{P}$ that are strongly selectable for $\hat{\mathbf{D}}$ such that there is $f \in P \setminus \bar{P}$ with $|\hat{D}_f| > 1$. We need then to prove that $\bar{P}$ is not selectable for domain $(\hat{\mathbf{D}}_{-f}, L(f, \hat{\mathbf{D}}))$.

Let us consider the obstacle domain $\tilde{\mathbf{D}}$, whose existence is in this case guaranteed by Lemma 30. Using (9) and (10) we can conclude, by Lemma 29, that every strongly selectable solution for $\hat{\mathbf{D}}$ must be strongly selectable also for $\tilde{\mathbf{D}}$. Hence, P and $\bar{P}$ are strongly selectable also for $\tilde{\mathbf{D}}$. But then, as $|\tilde{D}_f| > 1$ (cf. (11)) and since $\tilde{\mathbf{D}} \in \mathcal{X}$, by hypothesis, $\bar{P}$ is not selectable for $(\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$. By (8) we can invoke Lemma 29 again and obtain that $\bar{P}$ is not selectable for domain $(\hat{\mathbf{D}}_{-f}, L(f, \hat{\mathbf{D}}))$. □

Lemma 32. *If the properties of Lemma 27 are not satisfied for every obstacle domain $\tilde{\boldsymbol{D}} \in \mathcal{X}$, then they are not satisfied for every subdomain $\hat{\boldsymbol{D}}$ of $\boldsymbol{D}$.*

Proof. The claim is obvious if in $\hat{\mathbf{D}}$ there is only one selectable solution P, or all selectable solutions are also strongly selectable.

Consider then the following case. There are three solutions: P' is selectable but not strongly selectable for $\hat{\mathbf{D}}$, $\check{P}$ (not necessarily different from P') and $\bar{P}$ are selectable for $\hat{\mathbf{D}}$; there is $f \in (P' \cap \check{P}) \setminus \bar{P}$ that is a witness of P' not being strongly selectable. We need then to prove that either $\bar{P}$ is not selectable for domain $(\hat{\mathbf{D}}_{-f}, L(f, \hat{\mathbf{D}}))$ or $\check{P}$ is not selectable for domain $(\hat{\mathbf{D}}_{-f}, H(f, \hat{\mathbf{D}}))$. Recall that, by Observation 26, $|D_f| \geq |\hat{D}_f| > 1$.

As in the proof of Lemma 31, let us consider the obstacle domain $\tilde{\mathbf{D}}$ of Lemma 30. Using (10) we can conclude, by Lemma 29, that every selectable solution for $\hat{\mathbf{D}}$ must be selectable also for $\tilde{\mathbf{D}}$. Hence, in particular, P', $\bar{P}$ and $\check{P}$ are selectable also for $\tilde{\mathbf{D}}$. Furthermore, using (9) and invoking Lemma 29, we can conclude that since P' is not selectable for $(\hat{\mathbf{D}}_{-f}, H(f, \hat{\mathbf{D}}))$ then it is not selectable for $(\tilde{\mathbf{D}}_{-f}, H(f, \tilde{\mathbf{D}}))$. But then, as $|\tilde{D}_f| > 1$ (by (11)) and since $\tilde{\mathbf{D}} \in \mathcal{X}$, by hypothesis, either $\bar{P}$ is not selectable for $(\tilde{\mathbf{D}}_{-f}, L(f, \tilde{\mathbf{D}}))$ or $\check{P}$ is not selectable for $(\tilde{\mathbf{D}}_{-f}, H(f, \tilde{\mathbf{D}}))$. We can now argue as done above for P' and use either (8) or (9) in conjunction with Lemma 29 to conclude the proof. □

We are now ready to prove our main theorem on set systems.

Proof. As stated above, the "only if" direction follows from Lemma 25 and Lemma 27.

For the "if" direction, we can assume, according to Lemma 31 and Lemma 32, that for every subdomain of $\mathbf{D}$ the conditions of Lemmas 25 and 27 do not hold. This guides the definition of Algorithm 4. The algorithm incrementally removes from $\mathcal{F}$ the solutions that are found to be non-selectable for some subdomain (this is the set R in the algorithm). The construction of the subdomains closely exploits the properties coming from the unsatisfied lemmas. Specifically, the algorithm looks for an agent f we can safely ask for OSP-ness to diverge between their current $L(f)$ and $H(f)$; if f reveals type $L(f)$, then she will be securely selected, or if she reveals type $H(f)$, then she will be never selected. By hypothesis, such an agent always exists (either at Line 22 by the negation of Lemma 25 or at Lines 4–13 by the negation of Lemma 27). The algorithm makes the suitable queries. This query will update the current domain of agents, and thus the process can be repeated for the new domains. The algorithm continues until we are left with only one selectable solution, that will be returned.

Observe at each iteration either the set of $\mathcal{P} = \mathcal{F} \setminus R$ shrinks, or there is at least one agent whose type domain decreases in size. Hence, we eventually reach a point in which we either have one solution therein (and the algorithm stops) or the type of each agent has been revealed, i.e., $\tilde{\mathbf{D}}$ is such that $\tilde{D}_e = \{t_e\}$ for all $e \in E$. We now show that, also in this case, $|R| = |\mathcal{F}| - 1$. Suppose that this is not the case and that R contains two solutions P and P'. Let $H(e) = L(e) = t_e$ for each $e \in E$. Observe that $L(P \setminus P') = H(P \setminus P')$ and $L(P' \setminus P) = H(P' \setminus P)$. Since P and P' are selectable for $\tilde{\mathbf{D}}$, then

$$L(P' \setminus P) \leq H(P \setminus P') = L(P \setminus P') \leq H(P' \setminus P) = L(P' \setminus P)$$

and then the two solutions have the exact same cost. Thus, if $P \prec P'$, then only P is selectable, otherwise, only P' is selectable and one of them should have been in R.

Moreover, we will never remove all solutions from $\mathcal{F}$ since, as noted above, for any possible subdomain, there is always a selectable solution. We can then conclude that $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$ always returns a feasible solution. Moreover, it is immediate to check that the mechanism returns an optimal solution if for each e, the type t_e is compatible with the actions taken by agent e during the execution of

$\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$. Indeed, by Lemma 29 we know that the solutions removed for bigger subdomain, because they were not selectable, remain non-selectable for all the smaller domains.

We finally prove that $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$ satisfies OSP 2CMON which, by Theorem 5 and 9, implies that $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$ is OSP. First observe that for every agent e and every node u at which agent e interacts with the mechanism, given that $D_e(u)$ is the domain of e compatible with the action previously taken by e, either e is asked to diverge among type $H(e, D_e(u))$ and $D_e(u) \setminus \{H(e, D_e(u))\}$ (at Line 14), or it is asked to diverge among type $L(e, D_e(u))$ and $D_e(u) \setminus \{L(e, D_e(u))\}$ (at Line 5 or at Line 23). In the first case, no negative 2-cycle is possible, since whenever e takes the action compatible with $H(e, D_e(u))$, then, by hypothesis (i.e., negation of Lemma 27), every solution to which e belongs ceases to be selectable, and thus this agent is never selected. In the second case, no negative 2-cycle is possible, since whenever e takes the action compatible with $L(e, D_e(u))$, then, by hypothesis (i.e., negation of both Lemma 25 and Lemma 27), every solution to which e does not belong ceases to be selectable, and thus this agent is always selected. □

Note that the mechanisms $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$ runs in polynomial time, since it makes at most 2 queries to each agent.

Interestingly, this mechanism is not a DA auction or a PCA mechanism [33]. Indeed, it may require that single agents are involved first in an ascending phase and then in a descending phase or vice versa.

5.4 The Algorithmic Characterization

Note that the obstacle domain set contains at most $2|E| + 1$ domains. So we can enumerate all elements in this set in time that is polynomial in the size of the set system instance. Observe also that it takes only polynomial time (in the number of feasible solutions) to verify whether a solution is (strongly) selectable or not[3]. Hence, the *testing algorithm*, that for every domain in the obstacle domain set checks for whether the conditions of Lemmas 25 and Lemma 27 are satisfied, is a polynomial-time algorithm.

In conclusion, we have that the apparently hard-to-grasp characterization of OSP optimal mechanism provided by Theorem 28, can be immediately transformed in an algorithmic characterization as follows.

Corollary 33. *There is a polynomial-time testing algorithm that, given a set system $(E, \mathcal{F}, \boldsymbol{D})$, $\boldsymbol{D}$ being a three-value domain, decides if an OSP optimal mechanism exists for $(E, \mathcal{F}, \boldsymbol{D})$. Moreover, if such a mechanism exists, then $\mathcal{M}^{\mathsf{opt}}_{\mathsf{set}}$ is an OSP optimal mechanism for $(E, \mathcal{F}, \boldsymbol{D})$.*

6 Conclusions

We have focused on OSP mechanisms, a compelling and needed notion of incentive compatibility for bounded rationality; [33] 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.

[3]In some set systems, the number of feasible solution can be exponentially large with respect to the number of agents, and the focus is usually on algorithms that are polynomial only in the number of agents. In these settings, the naive algorithm for verifying whether a solution is selectable is then undesirable. However, this does not exclude that better algorithms can be found by exploiting the structure of the problem at hand (as, e.g., for path auctions).

We have introduced a new technique to look at OSP mechanisms; we have given tight results on the approximation and a characterization of these mechanisms for two paradigmatic problems in the area, machine scheduling and set systems. 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 [21, 8]. 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 [21] and $1 + \epsilon$, for combinatorial auctions with additive bidders in [8]).

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? An algorithmic framework to answer this question has been provided in [20] for problems admitting only binary outcomes. A 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. Some success has been achieved for the design, but not the characterization, of OSP combinatorial auctions with multidimensional bidders [13].

Acknowledgements

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

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] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *FOCS 2001*, pages 482–491, 2001.

[3] R. P. Arribillaga, J. Massó, and A. Neme. All sequential allotment rules are obviously strategy-proof. 2019.

[4] R. P. Arribillaga, J. Massó, and A. Neme. On obvious strategy-proofness and single-peakedness. *Journal of Economic Theory*, 186:104992, 2020.

[5] I. Ashlagi and Y. A. Gonczarowski. Stable matching mechanisms are not obviously strategy-proof. *J. Economic Theory*, 177:405–425, 2018.

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

[7] 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.

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

[9] 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.

[10] J. Bulow and P. Klemperer. Auctions versus negotiations. *The American Economic Review*, 86(1):180–194, 1996.

[11] S. Chawla, J. Hartline, D. Malec, and B. Sivan. Multi-parameter mechanism design and sequential posted pricing. In *STOC 2010*, pages 311–320, 2010.

[12] 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.

[13] B. de Keijzer, M. Kyropoulou, and C. Ventre. Obviously strategyproof single-minded combinatorial auctions. In *ICALP 2020*, pages 71:1–71:17, 2020.

[14] P. Dütting, V. Gkatzelis, and T. Roughgarden. The performance of deferred-acceptance auctions. *Math. Oper. Res.*, 42(4), 2017.

[15] A. Eden, M. Feldman, O. Friedler, I. Talgam-Cohen, and S. M. Weinberg. The competition complexity of auctions: A Bulow-Klemperer result for multi-dimensional bidders. In *EC 2017*, pages 343–343, 2017.

[16] 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.

[17] M. Feldman, A. Fiat, and A. Roytman. Makespan minimization via posted prices. In *EC 2017*, pages 405–422, 2017.

[18] D. Ferraioli, A. Meier, P. Penna, and C. Ventre. Automated optimal osp mechanisms for set systems: The case of small domains. In *WINE 2019*, 2019.

[19] D. Ferraioli, A. Meier, P. Penna, and C. Ventre. Obviously strategyproof mechanisms for machine scheduling. In *ESA 2019*, 2019.

[20] D. Ferraioli, P. Penna, and C. Ventre. Two-way greedy: Algorithms for imperfect rationality. *CoRR*, abs/2007.11868, 2020.

[21] D. Ferraioli and C. Ventre. Obvious strategyproofness needs monitoring for good approximations. In *AAAI 2017*, pages 516–522, 2017.

[22] D. Ferraioli and C. Ventre. Probabilistic verification for obviously strategyproof mechanisms. In *IJCAI 2018*, 2018.

[23] D. Ferraioli and C. Ventre. Obvious strategyproofness, bounded rationality and approximation: The case of machine scheduling. In *SAGT 2019*, 2019.

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

[25] J. Glazer and A. Rubinstein. An extensive game as a guide for solving a normal game. *Journal of Economic Theory*, 70:32–42, 1996.

[26] H. Gui, R. Müller, and R. V. Vohra. Dominant strategy mechanisms with multidimensional types. Discussion paper 1392, Northwestern Univ., 2004.

[27] J. Hartline and T. Roughgarden. Simple versus optimal mechanisms. In *EC 2009*, pages 225–234, 2009.

[28] 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.

[29] A. Kim. Welfare maximization with deferred acceptance auctions in reallocation problems. In *ESA 2015*, pages 804–815, 2015.

[30] P. Krysta and C. Ventre. Combinatorial auctions with verification are tractable. *Theor. Comput. Sci.*, 571:21–35, 2015.

[31] M. Kyropoulou and C. Ventre. Obviously strategyproof mechanisms without money for scheduling. In *AAMAS 2019*, 2019.

[32] R. Lavi and C. Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. *Games and Economic Behavior*, 67(1):99–124, 2009.

[33] S. Li. Obviously strategy-proof mechanisms. *American Economic Review*, 107(11):3257–87, 2017.

[34] A. Mackenzie. A revelation principle for obviously strategy-proof implementation. Research Memorandum 014, (GSBE), 2017.

[35] A. Malakhov and R. V. Vohra. Single and multi-dimensional optimal auctions - a network approach. Discussion paper 1397, Northwestern Univ., 2004.

[36] P. Milgrom and I. Segal. Deferred-acceptance auctions and radio spectrum reallocation. In *EC 2014*, 2014.

[37] N. Nisan and A. Ronen. Algorithmic Mechanism Design. *Games and Economic Behavior*, 35:166–196, 2001.

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

[39] M. Pycia and P. Troyan. Obvious dominance and random priority. In *EC 2019*, 2019.

[40] J.-C. Rochet. The taxation principle and multitime Hamilton-Jacobi equations. *Journal of Mathematical Economics*, 14(2):113–128, 1985.

[41] M. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *EC 2005*, pages 286–293, 2005.

[42] T. Sandholm and A. Gilpin. Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. In *AMEC 2003*, pages 73–91, 2003.

[43] P. Troyan. Obviously strategy-proof implementation of top trading cycles. *International Economic Review*, 60(3):1249–1261, 2019.

[44] C. Ventre. Truthful optimization using mechanisms with verification. *Theor. Comput. Sci.*, 518:64–79, 2014.

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

Online Appendix

A Extending CMON to OSP Mechanisms: the Proof

Proof of Theorem 5. We initially show that if $OSP_i^{(f,\mathcal{T})}$ has no negative-weight cycles then there exist payments $\mathbf{p}$ such that $\mathcal{M} = (f, \mathbf{p})$ is OSP with implementation tree $\mathcal{T}$. Fix i and $\mathcal{T}$ and consider the graph $OSP_i^{(f,\mathcal{T})}$. The idea is that an edge $((a, \mathbf{a}_{-i}), (b, \mathbf{b}_{-i}))$, for $a, b \in D_i$, $a \neq b$, and $\mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u)$, u being an ab-separating vertex for player i, in $OSP_i^{(f,\mathcal{T})}$ encodes the OSP constraint $P_i(\mathbf{b}) - P_i(\mathbf{a}) \leq a(f(b, \mathbf{b}_{-i})) - a(f(a, \mathbf{a}_{-i}))$, where $\mathbf{a} = (a, \mathbf{a}_{-i})$ and $\mathbf{b} = (b, \mathbf{b}_{-i})$.

Augment $OSP_i^{(f,\mathcal{T})}$ with a node ω and edges $(\omega, \mathbf{b})$ for any $\mathbf{b} \in D$ each of weight 0. Observe that ω does not belong to any cycle of the augmented $OSP_i^{(f,\mathcal{T})}$ since it has outgoing edges only. Therefore we can focus our attention on cycles of $OSP_i^{(f,\mathcal{T})}$. For any $\mathbf{b}$ in D, we let $\mathrm{SP}(\omega, \mathbf{b})$ be the length of the shortest path from ω to $\mathbf{b}$ in the augmented OSP-graph. Since $OSP_i^{(f,\mathcal{T})}$ is finite and does not have negative-weight cycles then $\mathrm{SP}(\omega, \mathbf{b})$ is well defined. It suffices, for all $\mathbf{b} \in D$, to set $P_i(\mathbf{b}) = \mathrm{SP}(\omega, \mathbf{b})$. Indeed, consider any edge $(\mathbf{a}, \mathbf{b})$ in $OSP_i^{(f,\mathcal{T})}$. The fact that the shortest path from ω to $\mathbf{a}$ followed by the edge $(\mathbf{a}, \mathbf{b})$ is a path from ω to $\mathbf{b}$ implies, by shortest path definition, that $\mathrm{SP}(\omega, \mathbf{b}) \leq \mathrm{SP}(\omega, \mathbf{a}) + a(f(b, \mathbf{b}_{-i})) - a(f(a, \mathbf{a}_{-i}))$. We can the conclude that for all $a, b \in D_i$, $a \neq b$, and $\mathbf{a}_{-i}, \mathbf{b}_{-i} \in D_{-i}(u)$, $P_i(\mathbf{b}) - P_i(\mathbf{a}) = \mathrm{SP}(\omega, \mathbf{b}) - \mathrm{SP}(\omega, \mathbf{a}) \leq a(f(b, \mathbf{b}_{-i})) - a(f(a, \mathbf{a}_{-i}))$.

Now we assume that the OSP-graph $OSP_i^{(f,\mathcal{T})}$ contains a negative-weight cycle. Let the negative-weight cycle be $C = (\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^{k+1})$, with $\mathbf{a}^{k+1} = \mathbf{a}^1$. Cycle C corresponds to the following OSP constraints:[4]

$$\begin{aligned}
P_i(\mathbf{a}^2) - P_i(\mathbf{a}^1) &\leq a_i^1(f(\mathbf{a}^2)) - a_i^1(f(\mathbf{a}^1)),\\
P_i(\mathbf{a}^3) - P_i(\mathbf{a}^2) &\leq a_i^2(f(\mathbf{a}^3)) - a_i^2(f(\mathbf{a}^2)),\\
&\ldots\\
P_i(\mathbf{a}^k) - P_i(\mathbf{a}^{k-1}) &\leq a_i^{k-1}(f(\mathbf{a}^k)) - a_i^{k-1}(f(\mathbf{a}^{k-1})),\\
P_i(\mathbf{a}^1) - P_i(\mathbf{a}^k) &\leq a_i^k(f(\mathbf{a}^1)) - a_i^k(f(\mathbf{a}^k)).
\end{aligned}$$

Suppose that there is a solution for the P's satisfying each of these k inequalities. This solution must also satisfy the inequality that results when we sum the k inequalities together. If we sum the left-hand sides, each unknown P^j is added in once and subtracted out once, so that the sum of the left-hand side is 0. The right-hand side sums to the cycle weight $w(C)$, and thus we obtain $0 \leq w(C)$. But since C is a negative-weight cycle, $w(C) < 0$, and we obtain the contradiction that $0 \leq w(C) < 0$. □

B Postponed Material about Machine Scheduling

B.1 Characterizing the Case of Two Jobs and Two Machines

The next result then gives a complete characterization of the approximability for two jobs and two agents in the three-value domains.

[4]Note that by the existence of $(\mathbf{a}^{j-1}, \mathbf{a}^j)$ and $(\mathbf{a}^j, \mathbf{a}^{j+1})$, , $1 < j \leq k$, we know that $\mathbf{a}_{-i}^j \in D_{-i}(u^i_{a_i^{j-1}, a_i^j}) \cap D_{-i}(u^i_{a_i^j, a_i^{j+1}})$ for opportune separating vertices $u^i_{a_i^{j-1}, a_i^j}$ and $u^i_{a_i^j, a_i^{j+1}}$.

Theorem 34. *For the machine scheduling problem, with two jobs and two agents with three values domains $D_i = \{L, M, H\}$, there is an OSP optimal mechanism if and only if $M \leq 2L$ or $H \leq 2M$.*

Proof. The "only if" direction follows from Proposition 7. Next we prove the "if" direction, by presenting OSP mechanisms whenever $M \leq 2L$ or $H \leq 2M$.

In the sequel, we write $x \approx y$ if $x < y \leq 2x$, and $x \prec y$ if $x < 2x < y$. With this notation, we require that $L \approx M$ or $M \approx H$. The optimal schedule is as follows, depending on each possible case:

$L \approx M \prec H$: If one agent has high cost and the other does not, schedule both jobs on the agent with lower cost. The OSP mechanism is depicted in Figure 3, where leaf nodes show the allocation and the payment of agent 1: The constants p_- and p_+ satisfy $L < p_- < M < p_+ < H$ and $2p_- \geq L + p_+$. As for agent 2, we pay p_+ for each assigned job.

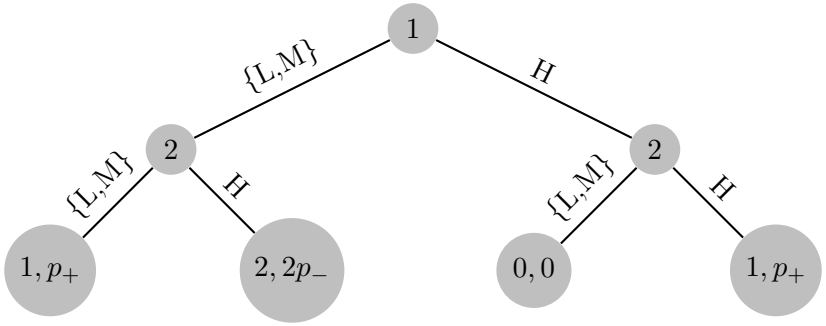

Figure 3: An exact OSP mechanism for scheduling two identical jobs when $M \leq 2L$ and $H > 2M$.

$L \prec M \approx H$: If one agent has low cost and the other does not, schedule both jobs on the agent with low cost. The OSP mechanism is shown in Figure 4, with payments and allocations being described as in the previous case, except that for agent 2 we pay p_- for each assigned job.

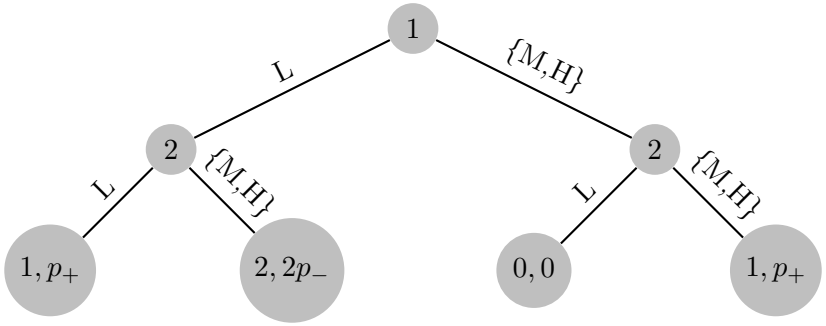

Figure 4: An exact OSP mechanism for scheduling two identical jobs when $M > 2L$ and $H \leq 2M$.

$L \approx M \approx H$: Here we distinguish two subcases. If $2L \geq H$, then we always schedule one job per machine. Otherwise, for $2L < H$, we only need to distinguish if one agent has cost L and the other has cost H (in which case both jobs go to the low cost agent). The mechanism is shown in Figure 5, with payments and allocations for agent 1 being described as in the previous cases. As for agent 2, her payment per unit of work is p_+ when separating $\{L, M\}$ from H (left subtree), and p_- when separating L from $\{M, H\}$ (right subtree). □

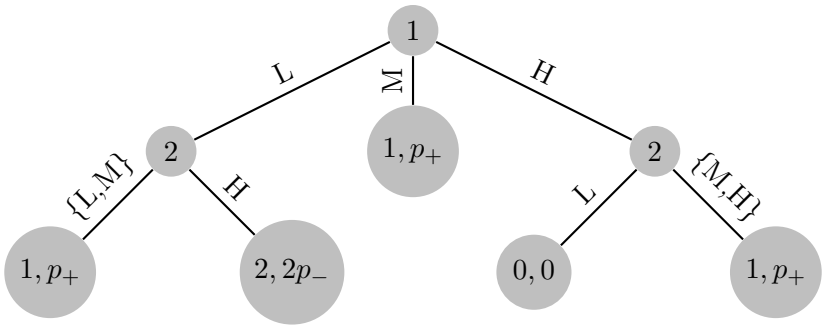

Figure 5: An exact OSP mechanism for scheduling two identical jobs when $M \leq 2L$, $H \leq 2M$, and $H > 2L$.

B.2 The Case of Heterogeneous Two-value Domains (Proof of Theorem 11)

In this section, we consider machine scheduling on *heterogeneous* two-value domains $D_i = \{L_i, H_i\}$, where $L_i \leq H_i$ for each agent i. In particular, we give an OSP optimal mechanism that runs in polynomial-time for this case (thus proving Theorem 11).

Notation. We denote the *makespan* of a solution (scheduling) $\mathbf{x}$ with respect to a profile (machines reported costs) $\mathbf{b}$ by $MS(\mathbf{x}, \mathbf{b}) := \max_i b_i \cdot x_i$, that is, the maximum agent cost (machine completion time). For any subset S of agents, and any profile $\mathbf{b}$, we denote by $\mathbf{b}_S$ the profile restricted to the agents in S only, and by $\mathbf{b}_{-S}$ the profile restricted to the agents not in S. Moreover, $\mathbf{L}_S$ ($\mathbf{H}_S$, respectively) denotes the profile in which all agents in S bid L_i (H_i, respectively). Analogously, $\mathbf{L}_{-S}$ and $\mathbf{H}_{-S}$ are the profiles in which all agents *not* in S bid L_i and H_i, respectively.

An optimal (greedy) algorithm. We next describe a simple optimal greedy algorithm GR which will lead to an optimal OSP mechanism.

Algorithm GR: Allocate jobs one by one greedily, breaking *ties* in favor of faster machines or, in case of same speed, in favor of lower index machines.

It is well known that the greedy algorithm returns the optimal scheduling for identical jobs. In addition, this particular tie-breaking rule will ensure that the algorithm returns the "most balanced" allocation. As stated in Fact 35 below, if we move a job from a machine to another one, the makespan relative to these two machines only (the maximum cost between them) will increase.

Fact 35. *Algorithm GR satisfies the following condition. For any two machines i and j, such that $b_i < b_j$, or $b_i = b_j$ and $i < j$, it holds that*

$$b_i \cdot (GR_i(\mathbf{b}) + 1) > b_j \cdot GR_j(\mathbf{b}) \qquad \textit{and} \qquad b_j \cdot (GR_j(\mathbf{b}) + 1) \geq b_i \cdot GR_i(\mathbf{b}) \ . \tag{12}$$

Proof. For every pair of machines i and j as above, $b_j \cdot (GR_j(\mathbf{b}) + 1) \geq b_j \cdot (GR_j^t(\mathbf{b}) + 1) \geq b_i \cdot (GR_i^t(\mathbf{b}) + 1) = b_i \cdot GR_i(\mathbf{b})$, where $GR_j^t(\mathbf{b})$ ($GR_i^t(\mathbf{b})$, respectively) denotes the number of jobs assigned to machine j (i, respectively) before step t in which the greedy algorithm assigns the last job to i, and $b_i \cdot (GR_i(\mathbf{b}) + 1) \geq b_i \cdot (GR_i^\tau(\mathbf{b}) + 1) > b_j \cdot (GR_j^\tau(\mathbf{b}) + 1) = b_j \cdot GR_j(\mathbf{b})$, where τ is the time in which the greedy algorithm assigns the last job to j. □

Note also that it is impossible that both conditions in (12) are unsatisfied for arbitrary b_i and b_j, otherwise we would have that $b_i \cdot (GR_i(\mathbf{b}) + 1) \leq b_j \cdot GR_j(\mathbf{b}) < b_j \cdot (GR_j(\mathbf{b}) + 1) < b_i \cdot GR_i(\mathbf{b})$ — a contradiction (recall that $b_i > 0$).

Moreover, if $b_j \cdot (GR_j(\mathbf{b}) + 1) \geq b_i \cdot GR_i(\mathbf{b})$, then it is impossible to satisfy both conditions even for the assignment $\mathbf{x}$ achieved from $GR(\mathbf{b})$ by moving a job from i to j: indeed, we have that $b_i \cdot (x_i + 1) = b_i \cdot GR_i(\mathbf{b}) \leq b_j \cdot (GR_j(\mathbf{b}) + 1) = b_j \cdot x_j$, so that the first condition is not satisfied.

Similarly, if $b_i \cdot (GR_i(\mathbf{b}) + 1) > b_j \cdot GR_j(\mathbf{b})$, then it is impossible to satisfy both the conditions even for the assignment $\mathbf{y}$ achieved from $GR(\mathbf{b})$ by moving a job from j to i: indeed, we have that $b_j \cdot (y_j + 1) = b_j \cdot GR_j(\mathbf{b}) < b_i \cdot (GR_i(\mathbf{b}) + 1) = b_i \cdot y_i$, so that the second condition is not satisfied.

The classical monotonicity condition says that, whenever an agent increases her own cost, her load will either decrease or stay the same. The next lemma says that, in our algorithm GR, the load of *every* other agent will either increase or stay the same.

Lemma 36. *For any profile* $\mathbf{b}$*, and for any two distinct agents (machines)* i *and* j*, it holds that*

$$GR_i(\mathbf{b}_{-j}, L_j) \leq GR_i(\mathbf{b}_{-j}, H_j).$$

Proof. Let $\mathbf{b}^L = (\mathbf{b}_{-j}, L_j)$ and $\mathbf{b}^H = (\mathbf{b}_{-j}, H_j)$, and $\mathbf{y}^L = GR(\mathbf{b}_{-j}, L_j)$ and $\mathbf{y}^H = GR(\mathbf{b}_{-j}, H_j)$. Using standard argument (see e.g. [2]), we first prove that the algorithm is monotone, that is, $y_j^L \geq y_j^H$. Suppose by contradiction that $y_j^L < y_j^H$. Observe that

$$MS(\mathbf{y}^L, \mathbf{b}^L) \leq MS(\mathbf{y}^H, \mathbf{b}^L) \leq MS(\mathbf{y}^H, \mathbf{b}^H) \leq MS(\mathbf{y}^L, \mathbf{b}^H)\ , \tag{13}$$

where the first and the last inequalities are due to the optimality of the algorithm, and the second inequality follows from the fact that $\mathbf{b}^H$ is obtained from $\mathbf{b}^L$ by increasing one machine cost. We next distinguish two cases according to which machine determines the makespan $MS(\mathbf{y}^L, \mathbf{b}^H)$. If this is machine j, meaning that $MS(\mathbf{y}^L, \mathbf{b}^H) = y_j^L H_j$, then (13) and the hypothesis $y_j^L < y_j^H$ implies

$$MS(\mathbf{y}^H, \mathbf{b}^H) \leq MS(\mathbf{y}^L, \mathbf{b}^H) = y_j^L H_j < y_j^H H_j \leq MS(\mathbf{y}^H, \mathbf{b}^H)$$

where the last inequality follows by definition of makespan. Thus we have a contradiction. Otherwise, $MS(\mathbf{y}^L, \mathbf{b}^H) = y_i^L b_i^H$ for some machine $i \neq j$, which means that $b_i^H = b_i^L$ as $\mathbf{b}^L$ and $\mathbf{b}^H$ differ only in machine j. Therefore, $MS(\mathbf{y}^L, \mathbf{b}^H) = y_i^L b_i^H = y_i^L b_i^L \leq MS(\mathbf{y}^L, \mathbf{b}^L)$. This inequality, combined with (13), implies that all inequalities in (13) hold with '='. We argue that this contradicts the fact that the algorithm breaks ties in a fixed manner, since

$$MS(\mathbf{y}^L, \mathbf{b}^H) = MS(\mathbf{y}^H, \mathbf{b}^H) \qquad\qquad MS(\mathbf{y}^H, \mathbf{b}^L) = MS(\mathbf{y}^L, \mathbf{b}^L) \tag{14}$$

and thus $\mathbf{y}^L$ would also be optimal for $\mathbf{b}^H$ and $\mathbf{y}^H$ would also be optimal for $\mathbf{b}^L$.

Since we assumed $y_j^H > y_j^L$, there must be another machine k such that $y_k^H < y_k^L$. On input $\mathbf{b}^H$, the definition of greedy algorithm implies

$$b_k \cdot (y_k^H + 1) \geq H_j \cdot y_j^H \tag{15}$$

because otherwise the algorithm would have assigned the last job of machine j to machine k instead.

Similarly, on input $\mathbf{b}^L$, the greedy algorithm must satisfy $L_j \cdot (y_j^L + 1) \geq b_k \cdot y_k^L$. Since $H_j \cdot y_j^H > L_j(y_j^L + 1)$ and $b_k \cdot y_k^L \geq b_k \cdot (y_k^H + 1)$, we have that $b_k \cdot (y_k^H + 1) \leq b_k y_k^L \leq L_j(y_j^L + 1) < H_j \cdot y_j^H$, which contradicts (15).

The lemma then follows from our choice for the tie-breaking rule. Suppose indeed that there is a machine $i \neq j$ that receives less jobs in $\mathbf{y}^H$ than in $\mathbf{y}^L$. Then it must exist also another machine $k \neq j$ that receives more jobs in $\mathbf{y}^H$ than in $\mathbf{y}^L$. Observe that it must be the case that, starting from $\mathbf{y}^H$, moving a job from k to i does not increase the makespan, i.e., $b_i(y_i^H + 1) \leq MS(\mathbf{y}^H, \mathbf{b}^H)$, otherwise $MS(\mathbf{y}^L, \mathbf{b}^L) \geq b_i y_i^L \geq b_i(y_i^H + 1) > MS(\mathbf{y}^H, \mathbf{b}^H) \geq MS(\mathbf{y}^H, \mathbf{b}^L)$, which contradicts the optimality of $\mathbf{y}^L$ with respect to $\mathbf{b}^L$. Hence, both $\mathbf{y}^H$ and the assignment $\bar{\mathbf{y}}^H$ achieved from $\mathbf{y}^H$ by moving a job from machine k to machine i are optimal for $\mathbf{b}^H$. Since $\mathbf{y}^H$ has been returned, it must be because it satisfies the tie-breaking condition for each pair of machines. In particular it must hold that

$$b_i \cdot (y_i^H + 1) \geq b_k \cdot y_k^H \tag{16}$$

with strict inequality if $b_k > b_i$, or $b_k = b_i$ and $k > i$.

Moreover, as showed above, it must be the case that $b_i \cdot (\bar{y}_i^H + 1) \leq b_k \cdot \bar{y}_k^H$ (with strict inequality if $b_k < b_i$ or $b_k = b_i$ and $k < i$). Since only one of the two tie-breaking conditions may fail, we have that $b_k \cdot y_k^H = b_k \cdot (\bar{y}_k^H + 1) \geq b_i \cdot \bar{y}_i^H = b_i \cdot (y_i^H + 1)$ (with strict inequality if $b_k < b_i$ or $b_k = b_i$ and $k < i$), that contradicts (16). □

Two next corollaries follow from a repeated application of Lemma 36.

Corollary 37. *For any subset S of agents (machines), for any $\mathbf{b}_S$, and for any $i \in S$, it holds that $GR_i(\mathbf{b}_S, \mathbf{L}_{-S}) \leq GR_i(\mathbf{b}_S, \mathbf{H}_{-S})$.*

Corollary 38. *For any partition of the agents (machines) into three subsets $(S, T, \{k\})$, and for every $\mathbf{b}_S$, $\mathbf{b}_T$, and b_k, it holds that*

$$GR_k(\mathbf{b}_S, b_k, \mathbf{L}_T) \leq GR_k(\mathbf{b}_S, b_k, \mathbf{b}_T) \leq GR_k(\mathbf{b}_S, b_k, \mathbf{H}_T).$$

Lemma 39. *For any subset $S \subset [n]$ of agents (machines), and for every $\mathbf{b}_S$, there is an agent $i_S \notin S$ such that*

$$GR_{i_S}(\mathbf{b}_S, \mathbf{L}_{-S}) \geq GR_{i_S}(\mathbf{b}_S, \mathbf{H}_{-S}). \tag{17}$$

Proof. We use Corollary 37 to compare the total number of jobs allocated to S in the two inputs, namely, we have that $\sum_{i \in S} GR_i(\mathbf{b}_S, \mathbf{L}_{-S}) \leq \sum_{i \in S} GR_i(\mathbf{b}_S, \mathbf{H}_{-S})$. As all jobs must be allocated to some machine, there must be a machine $i_S \notin S$ such that the reverse inequality holds for this machine, i.e., $GR_{i_S}(\mathbf{b}_S, \mathbf{L}_{-S}) \geq GR_{i_S}(\mathbf{b}_S, \mathbf{H}_{-S})$. □

Consider now the following mechanism:

Mechanism $\mathcal{M}_{GR}$:

1. Initially set $S = \emptyset$ and $\mathbf{b}_S = \emptyset$;
2. While $S \subset [n]$ do the following:
   - Pick an arbitrary agent $i_S \notin S$ satisfying (17);
   - Ask to i_S her bid b_{i_S};
   - Add i_S to S and update $\mathbf{b}_S$ accordingly;
3. Return $GR(\mathbf{b})$.

Theorem 40. *For any two-value domain $D_i = \{L_i, H_i\}$, the above mechanism $\mathcal{M}_{GR}$ is OSP.*

Proof. We show that GR satisfies OSP 2-CMON. By definition of $\mathcal{M}_{GR}$, each node where some agent diverges is a node u_{i_S}, and the agent i_S that diverges satisfies (17). We next show that i_S will receive at least $GR_{i_S}(\mathbf{b}_S, \mathbf{L}_{-S})$ jobs when her bid is $b_{i_S} = L_{i_S}$, and at most $GR_{i_S}(\mathbf{b}_S, \mathbf{H}_{-S})$ jobs when her bid is $b_{i_S} = H_{i_S}$:

$$GR_{i_S}(\mathbf{b}_S, L_{i_S}, \mathbf{b}_T) \geq GR_{i_S}(\mathbf{b}_S, L_{i_S}, \mathbf{L}_T) = GR_{i_S}(\mathbf{b}_S, \mathbf{L}_{-S}) \tag{18}$$

$$GR_{i_S}(\mathbf{b}_S, H_{i_S}, \mathbf{b}'_T) \leq GR_{i_S}(\mathbf{b}_S, H_{i_S}, \mathbf{H}_T) = GR_{i_S}(\mathbf{b}_S, \mathbf{H}_{-S}) \tag{19}$$

where inequalities are due to Corollary 38. Since i_S satisfies (17), i.e.,

$$GR_{i_S}(\mathbf{b}_S, \mathbf{L}_{-S}) \geq GR_{i_S}(\mathbf{b}_S, \mathbf{H}_{-S}),$$

the above inequalities imply $GR_{i_S}(\mathbf{b}_S, L_{i_S}, \mathbf{b}_T) \geq GR_{i_S}(\mathbf{b}_S, H_{i_S}, \mathbf{b}'_T)$ for all $\mathbf{b}_S$, $\mathbf{b}_T$ and $\mathbf{b}'_T$. □

B.3 Proving the approximation guarantee of $\mathcal{M}_{many}$ and $\mathcal{M}_{few}$

Proof of Theorem 18. We show that the mechanism $\mathcal{M}_{many}$ returns $\left(\frac{m+\lceil\sqrt{n}\rceil-1}{m} \cdot \lceil\sqrt{n}\rceil\right)$-approximate allocation. The claim then follows by simple algebraic manipulations. We denote with $OPT(\mathbf{x})$ the makespan of the optimal assignment when machines have type profile $\mathbf{x}$. We will use the same notation both if the optimal assignment is computed on a set of n machines and if it is computed and on a set of $\lceil\sqrt{n}\rceil$ machines, since these two cases can be distinguished through the input profile.

Fix a type profile $\mathbf{x}$. Let A be as at the beginning of the ascending phase. Let w be the last query done in this phase and let i be the last queried agent, i.e., the one that answered yes. Moreover, let $t = \min_{j \notin A} t_j^{out}$. It is immediate to see that $OPT(\mathbf{x}) \geq OPT(\mathbf{y})$ where $\mathbf{y}$ is such that $y_j = w$ for every $j \in A$, and $y_j = t$, otherwise. Moreover, let $\mathcal{M}(\mathbf{x})$ be the makespan of the assignment returned by our mechanism on the same input. Then, $\mathcal{M}(\mathbf{x})$ is equivalent to $OPT(\hat{\mathbf{z}})$, where $\hat{\mathbf{z}}$ is such that $\hat{z}_i = w$ and $\hat{z}_j = t$ for each remaining machine j. Hence, the theorem follows by proving that

$$\frac{OPT(\hat{\mathbf{z}})}{OPT(\mathbf{y})} \leq \frac{m + \lceil\sqrt{n}\rceil - 1}{m} \cdot \lceil\sqrt{n}\rceil .$$

Observe that the optimal assignment on input $\hat{\mathbf{z}}$ assigns a_i jobs to machine i and the remaining $m - a_i$ jobs to the remaining $\lceil\sqrt{n}\rceil - 1$ machines, for some $a_i \in \{1, \ldots, m\}$. In case of multiple assignment achieving the optimal makespan, we pick the one that maximizes a_i. Hence,

$$OPT(\hat{\mathbf{z}}) = \max\left\{a_i \cdot w, \left\lceil \frac{m - a_i}{\lceil\sqrt{n}\rceil - 1} \right\rceil \cdot t\right\}.$$

We next show that

$$OPT(\hat{\mathbf{z}}) \leq \frac{(m + \lceil\sqrt{n}\rceil - 1)tw}{t + (\lceil\sqrt{n}\rceil - 1)w}.$$

Suppose first that $OPT(\hat{\mathbf{z}}) = a_i \cdot w$. Since this is the optimal assignment, then it must be the case that

$$a_i \cdot w \leq \max\left\{(a_i - 1) \cdot w, \left\lceil \frac{m - a_i + 1}{\lceil\sqrt{n}\rceil - 1} \right\rceil \cdot t\right\} = \left\lceil \frac{m - a_i + 1}{\lceil\sqrt{n}\rceil - 1} \right\rceil \cdot t = \frac{m - a_i + 1 + \delta}{\lceil\sqrt{n}\rceil - 1} \cdot t,$$

where $\delta = 0$ if $r = (m - a_i + 1) \mod (\lceil\sqrt{n}\rceil - 1) = 0$, and $\delta = \lceil\sqrt{n}\rceil - 1 - r$, otherwise. Hence, we have that $a_i \leq \frac{(m+1+\delta)t}{t+(\lceil\sqrt{n}\rceil-1)w}$, and, consequently, $OPT(\hat{\mathbf{z}}) \leq \frac{(m+1+\delta)tw}{t+(\lceil\sqrt{n}\rceil-1)w} \leq \frac{(m+\lceil\sqrt{n}\rceil-1)tw}{t+(\lceil\sqrt{n}\rceil-1)w}$.

If instead $OPT(\hat{\mathbf{z}}) = \left\lceil \frac{m-a_i}{\lceil\sqrt{n}\rceil-1} \right\rceil \cdot t = \frac{m-a_i+\delta'}{\lceil\sqrt{n}\rceil-1}$, where $\delta' = \lceil\sqrt{n}\rceil - 2$ if $\delta = 0$ and $\delta' = \delta - 1$, otherwise, then it must be the case that, by our tie-breaking rule among optimal assignments, $\frac{m-a_i+\delta'}{\lceil\sqrt{n}\rceil-1} \cdot t < (a_i + 1)w$, from which we have $a_i > \frac{(m+\delta')t-(\lceil\sqrt{n}\rceil-1)w}{t+(\lceil\sqrt{n}\rceil-1)w}$, and, consequently,

$$OPT(\hat{\mathbf{z}}) < \frac{m + \delta' - \frac{(m+\delta')t-(\lceil\sqrt{n}\rceil-1)w}{t+(\lceil\sqrt{n}\rceil-1)w}}{\lceil\sqrt{n}\rceil - 1} \cdot t = \frac{(m + 1 + \delta')tw}{t + (\lceil\sqrt{n}\rceil - 1)w} \leq \frac{(m + \lceil\sqrt{n}\rceil - 1)tw}{t + (\lceil\sqrt{n}\rceil - 1)w}.$$

The optimal assignment on input $\mathbf{y}$ assigns b_i jobs to the $\lceil\sqrt{n}\rceil$ machines in A, for some $b_i \in \{1, \ldots, m\}$, and $m - b_i$ jobs to the remaining $n - \lceil\sqrt{n}\rceil$ machines. Hence,

$$OPT(\mathbf{y}) = \max\left\{ \left\lceil \frac{b_i}{\lceil\sqrt{n}\rceil} \right\rceil \cdot w, \left\lceil \frac{m - b_i}{n - \lceil\sqrt{n}\rceil} \right\rceil \cdot t \right\}.$$

We next show that

$$OPT(\mathbf{y}) \geq \frac{mtw}{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w}.$$

If $OPT(\mathbf{y}) = \left\lceil \frac{b_i}{\lceil\sqrt{n}\rceil} \right\rceil \cdot w = \frac{b_i+\gamma}{\lceil\sqrt{n}\rceil} \cdot w$, where $\gamma = 0$ if $r = b_i \mod \lceil\sqrt{n}\rceil = 0$, and $\gamma = \lceil\sqrt{n}\rceil - r$, otherwise, then $\frac{b_i+\gamma}{\lceil\sqrt{n}\rceil} \cdot w \geq \left\lceil \frac{m-b_i}{\lceil\sqrt{n}\rceil-1} \right\rceil \cdot t = \frac{m-b_i+\lambda}{n-\lceil\sqrt{n}\rceil} \cdot t$, where $\lambda = 0$ if $\rho = (m-b_i) \mod (n-\lceil\sqrt{n}\rceil) = 0$, and $\lambda = n - \lceil\sqrt{n}\rceil - \rho$, otherwise. Hence, we have $b_i \geq \frac{(m+\lambda)\lceil\sqrt{n}\rceil t-(n-\lceil\sqrt{n}\rceil)\gamma w}{\lceil\sqrt{n}\rceil t+(n-\lceil\sqrt{n}\rceil)w}$, and, consequently,

$$OPT(\mathbf{y}) \geq \frac{(m + \gamma + \lambda)tw}{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w} \geq \frac{mtw}{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w}.$$

If $OPT(\mathbf{y}) = \left\lceil \frac{m-b_i}{n-\lceil\sqrt{n}\rceil} \right\rceil \cdot t = \frac{m-b_i+\lambda}{n-\lceil\sqrt{n}\rceil} \cdot t$, then, $\frac{b_i+\gamma}{\lceil\sqrt{n}\rceil} \cdot w \leq \frac{m-b_i+\lambda}{n-\lceil\sqrt{n}\rceil} \cdot t$. Hence,

$$b_i \leq \frac{(m + \lambda)\lceil\sqrt{n}\rceil\, t - (n - \lceil\sqrt{n}\rceil)\gamma w}{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w},$$

and, consequently,

$$OPT(\mathbf{y}) \geq \frac{(m + \gamma + \lambda)tw}{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w} \geq \frac{mtw}{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w}.$$

Thus, putting things together, we get

$$\frac{OPT(\hat{\mathbf{z}})}{OPT(\mathbf{y})} \leq \frac{m + \lceil\sqrt{n}\rceil - 1}{m} \cdot \frac{\lceil\sqrt{n}\rceil\, t + (n - \lceil\sqrt{n}\rceil)w}{t + (\lceil\sqrt{n}\rceil - 1)w} \leq \frac{m + \lceil\sqrt{n}\rceil - 1}{m} \cdot \lceil\sqrt{n}\rceil,$$

where we used that $n - \lceil\sqrt{n}\rceil \leq \lceil\sqrt{n}\rceil^2 - \lceil\sqrt{n}\rceil = \lceil\sqrt{n}\rceil\,(\lceil\sqrt{n}\rceil - 1)$. □

Proof of Theorem 21. Let $\mathcal{OPT}(\mathbf{x})$ be the optimal outcome on a type profile $\mathbf{x}$. Let $A(\mathcal{M}_{few})$ be the set of alive machines at the end of the descending phase of mechanism $\mathcal{M}_{few}$.

We first consider the case that the optimal mechanism assigns jobs to only the $\lceil\sqrt{n}\rceil$ machines with lower type. In particular, let B be the set of machines that received a job in the optimal allocation, i.e., $B = \{i\colon f_i(\mathcal{OPT}(\mathbf{x})) > 0\}$. Note that we can assume that the machines that receive a job are in $A(\mathcal{M}_{few})$, and they are exactly the first $|B|$ machines to be removed from this set during the descending phase of our mechanism. We can also assume that assignment to these $|B|$ machines is monotone non-increasing, that is $f_i(\mathcal{OPT}(\mathbf{x})) \leq f_j(\mathcal{OPT}(\mathbf{x}))$ if j reveals her type before i. Indeed, these properties are satisfied by opportunely breaking ties among optimal outcomes.

Observe that $f_i(\mathcal{OPT}(\mathbf{x})) \cdot x_i \leq |B| \cdot t^{out}_{\min}$, otherwise a lower makespan would be achieved by moving a job from every machine in B to the one with type $t^{out}_{\min}$. However, this implies that the mechanism $\mathcal{M}_{few}$ also assigns at least $\min\{f_i(\mathcal{OPT}(\mathbf{x})), |P_0|_i\}$ jobs to machine $i \in B$, where $|P_0|_i$ is the number of jobs that mechanism $\mathcal{M}_{few}$ has not assigned yet when the type of i is revealed.

Now, for every $i \in B$, let B_i be the set of machines that revealed her type before i. Observe that, since the optimal outcome is monotone with respect to the order of revelation, then it must assign to i at least $\frac{m-\sum_{j\in B_i} f_j(\mathcal{OPT}(\mathbf{x}))}{|B|-|B_i|}$ jobs. On the other hand, our mechanism assigns to i at most

$$m - \sum_{j\in B_i} f_j(\mathcal{M}_{few}(\mathbf{x})) \leq m - \sum_{j\in B_i} f_j(\mathcal{OPT}(\mathbf{x})),$$

where the inequality follows from the observation above. Hence, the approximation ratio is at most $|B| - |B_i| \leq |B| \leq \lceil\sqrt{n}\rceil$.

Suppose now that optimal mechanism assigns jobs also to a machine that $\mathcal{M}_{few}$ remove during the descending phase. Observe that, since by hypothesis, $\mathcal{M}_{few}$ returns an outcome, then during the execution of the mechanism it always hold that $\zeta \cdot q \leq \lceil\sqrt{n}\rceil \cdot t^{out}_{\min}$. Since $\mathcal{OPT}(\mathbf{x})$ assigns at least one job to machines not in $A(\mathbf{x})$, we have in this case that $\max_i f_i(\mathcal{OPT}(\mathbf{x})) \geq t^{out}_{\min}$. By hypothesis, instead, we have that $f_i(\mathcal{M}_{few}(\mathbf{x})) \leq \lceil\sqrt{n}\rceil \cdot t^{out}_{\min}$ for every machine $i \in A(\mathbf{x})$ that receives at least one job. The bound then immediately follows. □

B.4 Ascending and Descending Auctions have Linear Approximation Ratio

Consider $m = n$ and, for every i, $L_i = L$, $M_i = M \geq n^2 \cdot L$, and $H_i = H \geq n^2 \cdot M$.

Consider a mechanism $\mathcal{M}$ and let $\mathcal{T}$ be its implementation tree. We restrict our attention to *full* mechanisms $\mathcal{M}$, i.e., mechanisms where all agents diverge on the path from the root of $\mathcal{T}$ where players play according to H.

Lemma 41. *Every non-full mechanism for the machine scheduling problem has approximation ratio at least n.*

Proof. Suppose $\mathcal{M}$ is a non-full mechanism for machine scheduling. This means that there exists at least one agent, say i, who will not diverge between H and L when all other agents have played according to H. But then the mechanism must return the same outcome on both the profiles $\mathbf{x} = (L, \mathbf{b}_{-i})$ and $\mathbf{y} = (H, \mathbf{b}_{-i})$ where $b_j = H$, for all $j \neq i$. Now if the outcome does not assign all jobs to i then the approximation of $\mathcal{M}$ on $\mathbf{x}$ is n; on the contrary, if $\mathcal{M}$ assigns all jobs to i then the approximation of $\mathcal{M}$ on $\mathbf{y}$ is n. □

Let us now rename the agents as follows: Agent 1 is the 1st agent that diverges in $\mathcal{T}$; note that agent 1 is well defined for all non-trivial mechanisms. Agent 2 is the 2nd agent that diverges,

different from agent 1, in the subtree of $\mathcal{T}$ defined by agent 1 taking an action compatible with type H; more generally, agent i is the ith agent that diverges, different from the agents $1, 2, \ldots, i-1$, in the subtree of $\mathcal{T}$ in which the actions taken by agents that previously diverged are compatible with their type being H. We let u_i denote the node of $\mathcal{T}$ in which i diverges as from above. Moreover, we let $c_i = M$ if, at node u_i, i diverges between L and M, and $c_i = H$ otherwise.

We have the following results, whose proofs follow the same ideas of the one given in Section 4.2.1.

Theorem 42. *Every non-trivial OSP mechanism for the machine scheduling problem such that $c_1 = M$ has approximation ratio at least n.*

Proof. Suppose that there is a k-approximate OSP mechanism $\mathcal{M}$ with $k < n$ — $\mathcal{M}$ is clearly non-trivial. Consider the type profile $\mathbf{x}$ such that $x_1 = M$, and $x_j = H$ for every $j \neq 1$. Observe that $\mathbf{x}$ is compatible with node u_1. The optimal allocation for the type profile $\mathbf{x}$ assigns all jobs to machine 1, with cost $OPT(\mathbf{x}) = n \cdot M$. Since $\mathcal{M}$ is k-approximate, then it also assigns all jobs to machine 1. Indeed, if a job is assigned to a machine $j \neq 1$, then the cost of the mechanism would be at least $H \geq n^2 \cdot M > k \cdot OPT(\mathbf{x})$, that contradicts the approximation bound.

Consider now the profile $\mathbf{y}$ such that $y_j = L$ for every j. Observe that also $\mathbf{y}$ is compatible with node u_1. Clearly, $OPT(\mathbf{y}) = L$. Since $\mathcal{M}$ is k-approximate, then it cannot assign all jobs to machine 1. Indeed, in this case the cost of the mechanism would be $nL > k \cdot OPT(\mathbf{y})$, that contradicts the approximation bound.

Hence, we have that if agent 1 takes actions compatible with M, then there exists a type profile compatible with u_1 such that 1 receives n jobs, whereas, if 1 takes a different action compatible with a lower type, then there exists a type profile compatible with u_1 such that 1 receives less than n jobs. However, this contradicts that the mechanism is OSP. □

Lemma 43. *For every $i < n$, and every full OSP k-approximate mechanism $\mathcal{M}$, with $k < n$, if $c_i = H$ and i at node u_i 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 machines.*

Proof. Suppose that there is $i < n$ and $\mathbf{x}_{-i}$ compatible with u_i such that if i takes an action compatible with type H, then $\mathcal{M}$ assigns a job to i. According to the definition of c_i, 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 $OPT(\mathbf{y}) = \left\lceil \frac{n}{n-i} \right\rceil \cdot L$. Since $\mathcal{M}$ is k-approximate, then it does not assign any job to machine i. Indeed, in this case the cost of the mechanism would be at least $M \geq n^2 L \geq n \cdot \left\lceil \frac{n}{n-i} \right\rceil L > k \cdot OPT(\mathbf{x})$, that contradicts the approximation bound.

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 that the mechanism is OSP. □

Theorem 44. *Every full OSP mechanism for the machine scheduling problem such that $c_i = H$ for every $i < n$ has approximation ratio at least n.*

Proof. Suppose now that there is an OSP k-approximate mechanism $\mathcal{M}$, for some $k < n$, such that $c_i = H$ for every $i < n$. 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 each job to a different machine, and has cost $OPT(\mathbf{x}) = H$.

According to Lemma 43, if machines take actions compatible with $\mathbf{x}$, then the mechanism $\mathcal{M}$ does not assign any job to machine i for every $i < n$. Hence, the outcome that $\mathcal{M}$ returns for type profile $\mathbf{x}$ consists in assigning all jobs to the remaining machine. Therefore, the cost of M is $n \cdot H > kOPT(\mathbf{x})$, that contradicts the approximation bound. □

This result implies a lower bound of n for the approximation of ascending and descending auctions.

Corollary 45. *Every ascending or descending auction has approximation ratio at least n for the machine scheduling problem.*

Proof. An ascending mechanism has $c_1 = M$ and, therefore, the lower bound follows from Theorem 42.

A full descending auction, instead, has $c_i = H$ for all i and then in particular for all $i < n$. The lower bound then follows from Lemma 41 and Theorem 44. □