Obvious Strategyproofness, Bounded Rationality and Approximation
The Case of Machine Scheduling

Diodato Ferraioli[1(✉)] and Carmine Ventre[2]

[1] Università degli Studi di Salerno, Fisciano, SA, Italy
dferraioli@unisa.it
[2] King's College London, London, UK
carmine.ventre@kcl.ac.uk

Abstract. Obvious strategyproofness (OSP) has recently emerged as the solution concept of interest to study incentive compatibility in presence of agents with a specific form of bounded rationality, i.e., those who have *no* contingent reasoning skill whatsoever. We here want to study the relationship between the approximation guarantee of incentive-compatible mechanisms and the *degree* of rationality of the agents, intuitively measured in terms of the number of contingencies that they can handle in their reasoning. We weaken the definition of OSP to accommodate for cleverer agents and study the trade-off between approximation and agents' rationality for the paradigmatic machine scheduling problem. We prove that, at least for the classical machine scheduling problem, "good" approximations are possible if and only if the agents' rationality allows for a significant number of contingencies to be considered, thus showing that OSP is not too restrictive a notion of bounded rationality from the point of view of approximation.

Keywords: Mechanism design · Machine scheduling · Simple mechanisms · Bounded rationality · Lookahead

1 Introduction

Mechanism design is an established research field, by now rooted in a number of academic disciplines including theoretical computer science and AI. Its main objective is that of computing in presence of selfish agents who might misguide the designer's algorithm if it is profitable for them to do so. The concept of *strategyproofness* (SP-ness) (a.k.a., *truthfulness*) ensures that the algorithm and the agents' incentives are compatible and computation is indeed viable.

SP is based on the assumption of full rationality: agents are able to consider all possible strategies and their combinations to reason about their incentives.

An extended abstract of the paper appeared as [15].
D. Ferraioli—Partially supported by GNCS-INdAM and by the Italian MIUR PRIN 2017 Project ALGADIMAR "Algorithms, Games, and Digital Markets".

D. Fotakis and E. Markakis (Eds.): SAGT 2019, LNCS 11801, pp. 77–91, 2019.
https://doi.org/10.1007/978-3-030-30473-7_6

Nevertheless, this assumption is seldom true in reality and it is often the case that people strategize against mechanisms that are known to be truthful [4]. One then needs a different notion to compute in the presence of agents with bounded rationality. The problem here is twofold: how can we formalize strategyproofness for agents with (some kind of) bounded rationality? If so, can we *quantify* this bounded rationality and relate that to the *performances* of the mechanisms?

The first question has been recently addressed by Li [18], who defines the concept of *obvious strategyproofness* (OSP-ness); this notion has attracted quite a lot of interest in the community [3,6,12–14,17,19,21,24]. Here, the mechanism is seen as an extensive-form game; when a decision upon the strategy to play has to be made, it is assumed that the reasoning of each agent i is as simple as the following: the *worst* possible outcome that she can get when behaving well (this typically corresponds to playing the game according to the so-called agent's true *type*) must be at least as good as the *best* outcome when misbehaving (that is, following a different strategy). Best/Worst are quantified over *all* the possible strategies that the players playing in the game after i can adopt. Li [18] proves that this is the right solution concept for a model of bounded rationality wherein agents have *no* contingent reasoning skills; rather than thinking about the possible cases of if-then-else's, an agent is guaranteed that honesty is the best strategy to follow no matter all the contingencies.

Given the OSP formalization of bounded rationality, we focus, in this work, on the second question. On the one hand, OSP is too restrictive in that people might be able, within their computational limitations, to consider *some* contingent reasoning, that is, a few cases of if-then-else's. On the other hand, OSP mechanisms appear to be quite limited, with respect to SP ones, in terms of their approximation guarantee [12,13]. The question then becomes:

> *Can we quantify the trade-off between the "degree" of bounded rationality of the agents and the approximation guarantee of the mechanisms incentivizing them?*

Our Contribution. The concept of *lookahead* is discussed in the literature in the context of (strategies to play) games, and agents with limited computational capabilities. De Groot [9] found that all chess players (of whatever standard) used essentially the same thought process – one based upon a lookahead heuristic. Shannon [23] formally proposed the lookahead method and considered it a practical way for machines to tackle complex problems, whilst, in his classical book on heuristic search, Pearl [20] described lookahead as the technique being used by "almost all game-playing programs".

We propose to consider lookahead as a way *to quantify bounded rationality*, in relation to OSP. Whilst in OSP the players have no lookahead at all, we here consider the case in which the agents have lookahead k, k going from 0 (OSP) to $n-1$ (SP). Intuitively, k measures the number of players upon which each player reasons about in her decision making. We allow the set of k "lookahead" players to be player and time specific (that is, different players can reason about different competitors, and the set of players is not fixed but may change at different time steps of the mechanism). So when agent i has to decide upon the

strategy to play, she will consider all the possible cases (strategies) for these k agents at that time (à la SP) and a no-contingent reasoning (à la OSP) for the others. This definition, which is somewhat different from that of the next k moves in the game, is dictated by different subtleties of extensive-form mechanisms. In particular, these k agents can be chosen in different ways to cover diverse angles. (A more technical discussion is deferred to Sect. 2.) In absence of other formal definitions of incentive compatibility for different degrees of rationality, we regard our definition of *OSP with k-lookahead* (k-OSP, for short) as a major conceptual contribution of our work.

We then look at the trade-off between the value of k and the approximation guarantee of k-OSP mechanisms. We focus of the well-studied problem of machine scheduling, where n agents control related machines and the objective is to schedule a set of m (identical) jobs to the machines so to minimize the *makespan* (i.e., the latest machine's completion time). In our main technical contribution, we prove a lower bound on approximation guarantee of $\tau_k(n) = \frac{\sqrt{k^2+4n}-k}{2}$, thus providing a smooth transition function between the known approximation factors of $\sqrt{n}$ for OSP mechanisms [12] and 1 for SP mechanisms [2]. We also show that this bound is tight, at least for three-values domains. (Such a restriction is common to the state of the art of OSP mechanisms [12].) Our lower and upper bounds significantly extend and generalize to k-OSP the analysis done in [12] for OSP mechanisms. Specifically, the lower bound needs to identify some basic properties of the function $\tau_k(n)$ and prove what features the *implementation tree* of a mechanism (i.e., extensive-form game induced by it) with good approximation guarantee must have. Our upper bound instead defines a mechanism (algorithm, implementation tree and payment function) which combines a descending auction phase, to identify a certain number of slowest machines, with an ascending auction to find out the $k+1$ fastest machines. The analysis of the approximation guarantee of our k-OSP mechanism is significantly more involved than the one used in [12] for $k = 0$.

The main message of our work is that having more rational agents only slightly improves the approximation guarantee of incentive-compatible mechanisms, at least in the case of machine scheduling. In fact, to have a constant approximation of the optimum makespan one would need agents with $\omega(1)$-lookahead. We can then conclude that, in the cases in which the agents are not that rational, OSP is not that restrictive a solution concept to study the approximation of mechanisms for agents with bounded rationality.

Related Work. Recent research in algorithmic mechanism design has suggested to focus on "simple" mechanisms to deal with bounded rationality [7,16,22]. OSP provides a formal definition for simple mechanisms, by focusing on a specific aspect of bounded rationality (see references above for the body of work on this concept). However, different concepts of simple mechanisms have been recently adopted in literature, most prominently posted-price mechanisms have received great attention and have been applied to many different settings [1,5,8,10,11].

2 The Definition

We have a set N of n agents; each agent i has a domain D_i of possible *types* – encoding some feature of theirs (e.g., their speed). The actual type of agent i is her private knowledge.

An extensive-form mechanism $\mathcal{M}$ is a triple $(f, p, \mathcal{T})$, where f is an algorithm that takes as input bid profiles and returns a feasible solution, $p = (p_1, \ldots, p_n)$ is the payment function, one for each agent, and $\mathcal{T}$ is an extensive-form game, that we call *implementation tree*[1]. Intuitively, $\mathcal{T}$ represents the steps that the mechanism will take to determine its outcome. More formally, each internal node u of $\mathcal{T}$ is labelled with a player $S(u)$, called the *divergent agent* at u, and the outgoing edges from u are labelled with types in the domain of $S(u)$ that are compatible with the history leading to u; the edge labels denote a partition of the compatible types. We denote by $D_i(u)$ the types in the domain of i that are compatible with the history leading to node $u \in \mathcal{T}$. The tree models how $\mathcal{M}$ interacts with the agents: at node u the agent $S(u)$ is queried and asked to choose an action, that corresponds to selecting one of u's outgoing edges. The chosen action *signals* that the type of $S(u)$ is in the set of types labeling the corresponding edge. The leaves of the tree will then be linked to (a set of) bid profiles; the mechanism will return (f, p) accordingly; in other words, each leaf corresponds to an outcome of the mechanism. (Observe that this means that the domain of f and p is effectively given by the leaves of $\mathcal{T}$.)

We use $\mathbf{b}$ to denote bid profiles, so that b_i stands for the type that i signalled to the mechanism. For simplicity, we use $f(\mathbf{b})$ and $p_1(\mathbf{b}), \ldots, p_n(\mathbf{b})$ to denote the outcome of (f, p) for the leaf of $\mathcal{T}$ to which $\mathbf{b}$ belongs. We assume that agents have quasi-linear utilities, that is, agent i of type t who signals (i.e., plays the game $\mathcal{T}$ according to) b has utility $u_i(b, \mathbf{b}_{-i}) = p_i(\mathbf{b}) - t(f(\mathbf{b}))$, where, with a slight abuse of notation, $t(f(\mathbf{b}))$ is the cost that player i pays to implement the outcome $f(\mathbf{b})$ when her type is t, and $\mathbf{b}_{-i}$ is the declaration vector of (i.e. types signalled by) all agents except i. (In general, we let $\mathbf{b}_A = (b_j)_{j \in A}$ for $A \subset N$.)

Figure 1 gives an example of an implementation tree where three players have a two-value domain $\{L, H\}$. The root partitions the domain of machine 1 into L and H. If we let v denote the left child of the root, then $D_1(v) = \{L\}$ as type H is no longer compatible with the history of v.

We now define *OSP with k-lookahead*. OSP informally implies that whenever an agent is asked to diverge, she is better off acting according to her true type in *any* possible future scenario: the worst possible outcome after selecting her true type is at least as good as the best possible outcome after misreporting her type, at that particular point in the implementation tree. This models agents with no contingent reasoning, i.e., those unable to think through hypothetical

[1] The literature on mechanism design usually omits $\mathcal{T}$ from the definition of mechanism, since it often focuses only on specific classes of mechanisms defined by a given implementation tree (e.g., direct revelation mechanisms, posted price mechanisms). However, it turns out that for OSP (and k-OSP) the design of the extensive-form implementation is essential to define the incentive constraints.

scenarios such as "if player 2 will play L and player 3 will play L, then I prefer L; if they will play L and H respectively, then I prefer L, too; ans so on". In OSP, agent thinking is gross-grained: "If I play L, then the outcome will correspond to leaves $l_1, \ldots, l_4$, otherwise it will correspond to leaves $l_5, \ldots, l_8$".

However, it would be possible that agents have some limited ability of doing contingent reasoning: they can think through hypothetical scenarios corresponding to the action profiles of some players, but not all of them. Specifically, we would like to model a player able to reason as follows: "If player 2 will play L, I know that by choosing L I will finish either in l_1 or in l_2, otherwise I will finish in l_5 or l_6; if player 2 will play R, then my choice will be between the outcomes corresponding to l_3 and l_4 and the one corresponding to l_7 and l_8". That is, we here consider a more finely grained partition of the leaves of the tree, allowing for some steps of contingent reasoning by the divergent agent. Intuitively, our definition will allow the agent to reason about the moves of k agents; informally, OSP with k-lookahead then implies that whenever an agent is asked to diverge, she is better off acting according to her true type for *any fixed* choice of strategies of the k agents she reasons about (just like truthfulness) and *any* possible future scenario of the actions of the remaining $n - k - 1$ agents.

For the formal definition, we need to introduce some more notation. We call a bid profile $\mathbf{b}$ *compatible with* u if $\mathbf{b}$ is compatible with the history of u for all agents. We furthermore say that $(t, \mathbf{b}_{-i})$ and $(b, \mathbf{b}'_{-i})$ diverge at u if $i = S(u)$ and t and b are labels of different edges outgoing u (we sometimes will abuse notation and we also say that t and b diverge at u). E.g., (L, H, H) and (L, L, H) are compatible with node v on Fig. 1 and diverge at that node, whilst (L, L, H) and (L, L, L) are compatible with v but do not diverge at v.

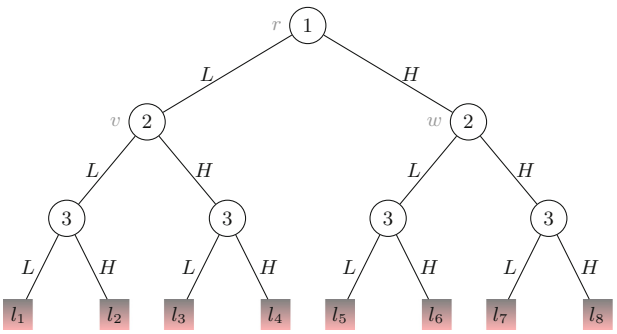

Fig. 1. An implementation tree with three players with two-value domains $\{L, H\}$; each player separates the domain types upon playing; at each leaf l_i the mechanism computes $f(\mathbf{b})$ and $p(\mathbf{b})$, $\mathbf{b}$ being the bid vector at l_i.

For every agent i and types $t, b \in D_i$, we let $u^i_{t,b}$ denote a vertex u in the implementation tree $\mathcal{T}$, such that $(t, \mathbf{b}_{-i})$ and $(b, \mathbf{b}'_{-i})$ are compatible with u, but diverge at u for some $\mathbf{b}_{-i}, \mathbf{b}'_{-i} \in D_{-i}(u) = \times_{j \neq i} D_j(u)$. Note that such a vertex might not be unique as agent i will be asked to separate t from b in different

paths from the root (but only once for every such path). We call these vertices of $\mathcal{T}$ *tb-separating* for agent i. For example, the node r in the tree in Fig. 1 is a LH-separating node for agent 1; while v and w are two LH-separating node for agent 2. These nodes are crucial, as at any point in which an agent distinguishes two different types we will need to add a (set of) constraints to account for her incentives. We finally denote i's lookahead at $u^i_{t,b}$ as $\mathcal{L}_k(u^i_{t,b})$, that is, a set of (at most) k agents that move in $\mathcal{T}$ after i. (When k is clear from the context, we simply let $\mathcal{L}(u)$ be the lookahead of agent $S(u)$ at u.)

Definition 1 (OSP with k-lookahead). *An extensive-form mechanism $\mathcal{M} = (f, \mathcal{T}, p)$ is OSP with k-lookahead (k-OSP, for short) given $\mathcal{L}_k(u^i_{t,b})$, if for all i, $t, b \in D_i$, t being i's true type, $u^i_{t,b} \in \mathcal{T}$, $\mathbf{b}_K \in D_K(u^i_{t,b})$ and $\mathbf{b}_T, \mathbf{b}'_T \in D_T(u^i_{t,b})$, it holds that*

$$u_i(t, \mathbf{b}_K, \mathbf{b}_T) \geq u_i(b, \mathbf{b}_K, \mathbf{b}'_T),$$

where $K = \mathcal{L}_k(u^i_{t,b})$, $T = N \setminus (K \cup \{i\})$ and $D_A(u) = \times_{j \in A \subset N} D_j(u)$.

In words, a mechanism is OSP with lookahead if each agent is willing to behave truthfully at each node of the tree in which she interacts with the mechanism, provided that she exactly knows the types of agents in K ($\mathbf{b}_K$ is the same either side of the inequality) but has no information about agents in T, except that their types are compatible with the history.

We remark that with $k = 0$ we get the definition of OSP – wherein K is empty – and with $k = n - 1$ we have truthfulness, T being empty.

Discussion. The set $\mathcal{L}_k(u)$ in the definition above crucially captures our notion of lookahead. We highlight the following features of our definition. The size of set $\mathcal{L}_k(u)$ tells us how many players, agent $S(u)$ can contingently reason about. This means that the boundaries of k indeed go from 0, which corresponds to OSP, to $n - 1$, which is equivalent to strategyproofness. In this sense, our definition represents a smooth transition between the two notions, measuring the degree of rationality of the players. For example, consider Fig. 1 and focus on player 1; when $k = 0$ then our notion is exactly OSP and the constraints require to compare the utility of 1 in the leaves $l_1, \ldots, l_4$ with her utility in $l_5, \ldots, l_8$; when, instead, $k = 1$ and $\mathcal{L}_1(r) = \{2\}$ then the constraints compare the utility of 1 in the leaves l_1, l_2 with that in l_5, l_6 (this corresponds to the case in which 2 plays L) *and* the utility of 1 in the leaves l_3, l_4 with that in l_7, l_8 (this corresponds to the case in which 2 plays H); finally, for $k = 2$ we get truthfulness as we need to compare the utility of 1 in l_j and l_{4+j} for $j = 1, \ldots, 4$. We note that intermediate values of k are consistent with the vast literature stating that human reasoning only has limited depth: for example, it is known that in chess most professional players are usually able to think ahead few steps only [9]. We remark that k-OSP differs from k-level reasoning: the latter considers a Nash equilibrium in which an agent plays a best response to what happens in the next k steps; the former considers a(n obviously) dominant strategy.

The set $\mathcal{L}_k(u)$ depends on u; this means that the number and the identities of players on which $S(u)$ can reason about can (in principle) adaptively depend

on the actual position in the implementation tree. This in particular allows us to also capture extensive-form games where the choice of the players to query is adaptive and a definition of lookahead where the players on which $S(u)$ can reason about are (a subset of) those who move next: this is for example the case in many multi-player board games in which the player can take actions that change who is the next player to play, e.g., by blocking some opponents or reversing the order of play.

Note that whenever $\mathcal{L}_k(u) = \mathcal{L}_k(v)$ for $S(u) = S(v)$ then we model the case in which the lookahead is independent from the actual implementation tree and only depends on $S(u)$'s prior knowledge of the other agents.

Differently from the examples of chess and multi-player board games in which a player only looks ahead to opponents that play in the next rounds, our definition of $\mathcal{L}_k(u)$ allows this set to contain also players that will play far away in the future. This clearly makes our definition more general.

Moreover, we observe that this definition of $\mathcal{L}_k(u)$ also allows us to overcome a paradox that would arise if one defines the set of opponents that one looks ahead only with respect to the implementation tree. For the sake of argument, let us fix $k = 1$. Consider an adaptive implementation tree, where at node u different actions taken by agent $S(u)$ correspond to different players taking the next move. As a limit case, one can imagine that $S(u)$ has $n-1$ different available actions and each of them enables a different opponent to react (e.g., this is the case for those board games where each player can decide who plays next). Hence, assuming that $S(u)$ can look ahead to players moving in the next step means that $S(u)$ has the ability to look ahead to all of them. Hence, in this setting limited look-ahead is not limiting at all the ability of contingent reasoning of $S(u)$ (that is, in this setting every mechanism that is 1-OSP according to this tree-only definition of lookahead is actually SP).

This is not surprising, since in this setting we are giving each agent i the chance to "reason about" each opponent regardless of the action that i takes. A more realistic alternative would be to assume that the agent exactly knows the actions of an opponent j only when i takes an action that enables j to be the next player to play (e.g., in the board game example described above, the current player i is assumed to know which actions player j will take when i chooses j as the next player to play, but i has no hint about the actions of j if she chooses $k \neq j$ as the next player to play). However, in this case i would have to reason about all the possible action combinations of all the different players that move after her; this might not weaken OSP and indeed means that the agent is not more rational at all. In fact, a careful inspection shows that, in this case, 1-OSP according to this alternative definition of tree-only lookahead has the same constraints of OSP.

Anyway, it must be highlighted that in non-adaptive trees, i.e., trees where the identity of the next player to move after $S(u)$ is the same irrespectively of $S(u)$'s action, tree-only lookahead would indeed weaken OSP and effectively capture a more rational agent capable of one step of contingent reasoning. Since this is a special case of our notion, our lower bound continues to hold.

Our definition requires that an agent with k-lookahead is capable of exactly pinpointing the type of the agents in K. This is in fact the same assumption that is implicitly done in the classical definition of truthfulness. Moreover, this makes our definition of k-OSP mechanism a special case of mechanisms implementable with *partition dominant strategy* as defined in [24]. Consequently, our definition satisfies a natural generalization of the standard decision theory axioms of monotonicity, continuity and independence, necessary to model the reasoning of agents with a knowledge of the state of nature (e.g., the type profiles) limited only to partitions of the set of these states (e.g., the type profiles that are compatible with the history of the mechanism). We also observe that this requirement only reinforces our lower bound below (even if they were so rational to do that, still the approximation guarantee would be a constant only for non-constant values of k). However, we leave open the problem of understanding whether our upper bound is tight even for a weaker notion of rationality where the types of the agents in K are not fully known but only have further restricted domains (e.g., an agent with k-lookahead only knows the next ℓ actions, for some $\ell > 0$, that will be taken by the agents in K).

3 The Case of Machine Scheduling

We now study the relationship between lookahead and approximation for the well-studied problem of machine scheduling. Here, we are given a set of m identical jobs to execute and the n agents control related machines. Agent i's type is a job-independent processing time t_i per unit of job (equivalently, an execution speed $1/t_i$ that is independent from the actual jobs). The algorithm f must choose a possible schedule $f(\mathbf{b}) = (f_1(\mathbf{b}), \ldots, f_n(\mathbf{b}))$ of jobs to the machines, where $f_i(\mathbf{b})$ denotes the job load assigned to machine i when agents take actions signalling $\mathbf{b}$. The cost that agent i faces for the schedule $f(\mathbf{b})$ is $t_i(f(\mathbf{b})) = t_i \cdot f_i(\mathbf{b})$. We focus on algorithms f^* minimizing the *makespan*, i.e., $f^*(\mathbf{b}) \in \arg\min_{\mathbf{x}} \max_{i=1}^{n} b_i(\mathbf{x})$; f is α-approximate if it returns a solution with cost at most α times the optimum.

3.1 Lower Bound

Let $\tau_k(n) = \frac{\sqrt{k^2+4n}-k}{2}$. That is, τ_k is a function of n such that $n = \tau_k(n)(\tau_k(n)+k)$. Observe that $\tau_0(n) = \sqrt{n}$ and $\tau_{n-1}(n) = 1$. In this section, we prove the following theorem, that states the main result of our work. Henceforth, for sake of readability, let us denote $\tau := \tau_k(n)$.

Theorem 1. *For the machine scheduling problem, no k-OSP mechanism can be better than τ-approximate, regardless of the value of the sets $\mathcal{L}_k(\cdot)$. This even holds for homogeneous three-value domains, i.e., $D_i = \{L, M, H\}$ for each i.*

Proof. Consider $m = n$. Moreover, consider a domain $D_i = \{L, M, H\}$ for every i, with $M \geq \tau \left\lceil \frac{m}{\lfloor \tau \rfloor} \right\rceil L$ and $H \geq \tau \cdot mM$.

The proof will work in three steps. First, we prove some algebraic property of τ (cf. Lemma 1). We then characterize implementation tree and algorithm of a k-OSP mechanism with approximation better than τ (cf. Lemma 2). Finally, we identify an instance for which any such mechanism cannot return an approximation better than τ – a contradiction.

Lemma 1. *$\tau = c + \delta$, with $\delta \in \left[0, \frac{k}{\tau+k-1}\right]$, where $c = \max\left\{\alpha \in \mathbb{N}\colon k \leq \frac{n-c^2}{c}\right\}$.*

Suppose now that a mechanism $\mathcal{M}$ with approximation ratio $\rho < \tau$ exists for the setting at the hand, and let $\mathcal{T}$ be its implementation tree. Let us rename the agents as follows: Agent 1 is the 1st distinct agent that diverges in $\mathcal{T}$; because of its approximation guarantee, the mechanism must have at least one divergent agent for our domain. We now call agent 2, the 2nd distinct agent that diverges in the subtree of $\mathcal{T}$ defined by agent 1 taking an action signalling 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 taken by agents that previously diverged are signalling 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 diverges in the subtree in which all the other agents have taken actions signalling H; if i got her id arbitrarily, then we denote with u_i a dummy node. We then have the following lemma.

Lemma 2. *Any k-OSP $\mathcal{M}$ which is ρ-approximate, with $\rho < \tau$, must satisfy the following conditions:*

1. *For every $i \leq n + 1 - \lceil\tau\rceil - k$, if agent i diverges at node u_i, it must diverge on M and H.*
2. *For every $i \leq n - \lfloor\tau\rfloor - k$, if agent i diverges at node u_i and takes an action signalling type H, then $\mathcal{M}$ does not assign any job to i whenever the action of agents in $\mathcal{L}(u_i)$ are all signalling H.*

Proof. Let us first prove part 1. Suppose that there is $i \leq n + 1 - \lceil\tau\rceil - k$ such that at node u_i i does not diverge on M and H (i.e., any action signalling M is signalling also H). Then it must diverge on L and M, since u_i must have at least two outgoing edges (since i is assumed to diverge at u_i), and the remaining edges can only be labeled with L. Consider the type profile $\mathbf{x}$ such that $x_i = M$, and $x_j = H$ for every $j \neq i$. Observe that, by definition of u_i, $x_j \in D_j(u_i)$ for every agent j. 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 ρ-approximate, then it also assigns all jobs to machine i. Indeed, if a job is assigned to a machine $j \neq i$, then the cost of the mechanism would be at least $H \geq \tau \cdot mM > \rho \cdot OPT(\mathbf{x})$, that contradicts the approximation bound.

Consider now the profile $\mathbf{y}$ such that $y_i = L$, $y_j = H$ for every $j < i$ and $j \in \mathcal{L}(u_i)$, and $y_j = L$ for every $j > i$ such that $j \notin \mathcal{L}(u_i)$. (We stress that our lower bound holds no matter the definition of the sets $\mathcal{L}(u_i)$.)

Observe that, as for $\mathbf{x}$, we have that $y_j \in D_j(u_i)$ for every agent j. It is not hard to see that $OPT(\mathbf{y}) \leq \left\lceil \frac{m}{n-i-k+1} \right\rceil L$. Let μ be the number of jobs that $\mathcal{M}$ assigns to machine i in this case. Since $\mathcal{M}$ is ρ-approximate, then $\mu < m$. Indeed, if $\mu = m$, then the cost of the mechanism contradicts the approximation bound, since $mL \geq \tau \left\lceil \frac{m}{n-i-k+1} \right\rceil L > \rho \cdot OPT(\mathbf{y})$, where we used that

$$\tau \left\lceil \frac{m}{n-i-k+1} \right\rceil \leq \tau \left\lceil \frac{n}{\lceil \tau \rceil} \right\rceil = \tau \left\lceil \frac{\tau(\tau+k)}{\tau+1-\delta} \right\rceil$$
$$\leq \tau \frac{\tau(\tau+k)+(\tau-\delta)}{\tau+1-\delta} \leq \tau(\tau+k) = m,$$

where the last inequality follows from $\delta \leq \frac{k}{\tau+k-1}$ by Lemma 1.

Hence, for the mechanism to be OSP with k-lookahead we need that both the following conditions are satisfied: (i) $p_i(\mathbf{x}) - mM \geq p_i(\mathbf{y}) - \mu M$, and (ii) $p_i(\mathbf{y}) - \mu L \geq p_i(\mathbf{x}) - mL$, where $p_i(\mathbf{x})$ and $p_i(\mathbf{y})$ denote the payment that i receives from the mechanism $\mathcal{M}$ when agents' actions are signalling $\mathbf{x}$ and $\mathbf{y}$, respectively. However, this leads to the contradiction that $L \geq M$.

Let us now prove part 2. Suppose that there is $i \leq n - \lfloor \tau \rfloor - k$ and $\mathbf{x}_{-i}$, with $x_j \in D_j(u_i)$ for every agent j and $x_j = H$ for every $j \in \mathcal{L}(u_i)$, such that if i takes an action signalling type H, then $\mathcal{M}$ assigns at least a job to i. According to part 2, 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 $j \leq i + k$ with $i \neq j$, and $y_j = L$ for $j > i + k$. Observe that $OPT(\mathbf{y}) = \left\lceil \frac{m}{n-i-k} \right\rceil \cdot L$. Since $\mathcal{M}$ is ρ-approximate, then it does not assign any job to machine i, otherwise its cost would be at least $M \geq \tau \left\lceil \frac{m}{\lfloor \tau \rfloor} \right\rceil L \geq \tau \left\lceil \frac{m}{n-i-k} \right\rceil L > \rho \cdot OPT(\mathbf{x})$.

Hence, for the mechanism to be OSP with k-lookahead we need that both the following conditions are satisfied: (i) $p_i(\mathbf{x}) - H \geq p_i(\mathbf{y}) - 0$, and (ii) $p_i(\mathbf{y}) - 0 \geq p_i(\mathbf{x}) - M$. However, this leads to the contradiction that $H \leq M$. □

Roughly speaking, Lemma 2 states that any k-OSP mechanism must have an implementation tree such that the first $n - \lfloor \tau \rfloor - k$ agents interacting with the mechanism, must be asked if their type is H, and, in the case of affirmative answer, they must not receive any job.

We next observe that such a mechanism cannot have approximation lower than τ, contradicting our hypothesis that M was k-OSP and ρ-approximate.

To this aim, assume first that for each agent $i \leq n - \lfloor \tau \rfloor - k$ diverges at u_i. We consider the profile $\mathbf{x}$ such that $x_i = H$ 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 2, since $\mathcal{M}$ is supposed to be k-OSP, if machines take actions that signal $\mathbf{x}$, then the mechanism $\mathcal{M}$ does not assign any job to machine i, for every $i \leq n - \lfloor \tau \rfloor - k$. Hence, the best outcome that $\mathcal{M}$ can return for $\mathbf{x}$ consists in fairly assigning the m jobs to the remaining $\lfloor \tau \rfloor + k$ machines. Observe that, if $\delta = 0$, i.e., τ is an integer, then each machine receives τ job, and thus the cost of $\mathcal{M}$ is at least $\tau H > \rho OPT(\mathbf{x})$, which contradicts the approximation

ratio of $\mathcal{M}$. Otherwise, there is at least one machine that receives at least $\lceil\tau\rceil$ jobs, since $\lfloor\tau\rfloor(\lfloor\tau\rfloor + k) < \tau(\tau + k) = m$. In this case, the cost of $\mathcal{M}$ is at least $\lceil\tau\rceil H > \tau H = \tau OPT(\mathbf{x})$, contradicting again the approximation ratio of $\mathcal{M}$.

Consider now the case that there is $1 < i \leq n - \lfloor\tau\rfloor - k$ that does not diverge at u_i. It is not hard to see that this would contradict the approximation of $\mathcal{M}$ given that it would be unaware of the type of too many machines. □

3.2 Upper Bound

We next show that for every k and every possible choice of lookahead sets $\{\mathcal{L}_k(u)\}_{u\in\mathcal{T}}$, the bound above is tight, for three-values domains, i.e., $D_i = \{L_i, M_i, H_i\}$ for every i. To this aim, consider the following mechanism $\mathcal{M}_k$, that consists of a Descending Phase (Algorithm 1) followed by an Ascending Phase (Algorithm 2). The algorithmic output is augmented with a payment, to agent i, of M_i for each unit of job load received.

1 Set $A = [n]$, and $t_i = \max\{d \in D_i\}$
2 **while** $|A| > \lceil\tau\rceil + k$ **do**
3 Set $p = \max_{a\in A}\{t_a\}$ and $i = \min\{a \in A\colon t_a = p\}$
4 Ask machine i if her type is equal to p
5 **if** *yes* **then** remove i from A, and set $t_i = p$
6 **else** set $t_i = \max\{t \in D_i\colon t < p\}$

Algorithm 1: The descending phase keeps in A the machines that are still *alive* and in t_i the maximum non-discarded type for each agent; then proceeds by removing from A the slowest machines, until there are only $\lceil\tau\rceil + k$ left.

1 Set $s_i = \min\{d \in D_i\}$
2 Set $B = \emptyset$
3 **while** $|B| \leq k$ **do**
4 Set $p = \min_{a\in A\setminus B}\{s_a\}$ and $i = \min\{a \in A \setminus B\colon s_a = p\}$
5 Ask machine i if her type is equal to p
6 **if** *yes* **then** Set $t_i = p$ and insert i in B
7 **else** set $s_i = \min\{d \in D_i\colon d > p\}$
8 Consider the profile $\hat{\mathbf{z}}$ with $\hat{z}_i = t_i$ for $i \in B$ and $\hat{z}_j = \min_{w\notin A} t_w$ for $j \in A \setminus B$
9 Let $f^\star(\hat{\mathbf{z}}) = (f_i^\star(\hat{\mathbf{z}}))_{i\in A}$ be the optimal assignment of jobs on input profile $\hat{\mathbf{z}}$
10 Assign $f_j^\star(\hat{\mathbf{z}})$ jobs to each machine $j \in A$

Algorithm 2: The ascending phase adds to B the k fastest machines; then it computes the optimal assignment by using the revealed type for machines in B and a suitably chosen placeholder type for the remaining machines.

In case of multiple optimal assignments in line 9 of Algorithm 2, we assume that the mechanism returns the one that maximizes the number of jobs assigned to machines in B. This is exactly the solution returned by the optimal greedy algorithm, and thus can be computed in polynomial time.

Roughly speaking, mechanism $\mathcal{M}_k$ works by discovering in the descending phase the $n - \lfloor \tau \rfloor - k$ slowest machines and discarding them (i.e., no job will be assigned to these machines). (Our mechanism satisfies the conditions of Lemma 2 thus showing that our analysis is tight for both approximation and design of the mechanism.) The ascending phase then serves to select a good assignment to the non-discarded machines. To this aim, the mechanism discovers in the ascending phase the $k+1$ fastest machines. The assignment that is returned is then the optimal assignment to the non-discarded machines in the case that the type of the $k+1$ fastest machines is as revealed, whereas the type of the remaining non-discarded machines is supposed to be as high as possible, namely equivalent to the type of the last discarded machine (i.e., the fastest among the slow machines).

Proposition 1. *Mechanism $\mathcal{M}_k$ is k-OSP if $D_i = \{L_i, M_i, H_i\}$ for each i.*

Proof. We prove that $M_i \cdot f_i(\mathcal{M}_k(\mathbf{x})) - x_i \cdot f_i(\mathcal{M}_k(\mathbf{x})) \geq M_i \cdot f_i(\mathcal{M}_k(\mathbf{y})) - x_i \cdot f_i(\mathcal{M}_k(\mathbf{y}))$ for each machine i, for each node u in which the mechanism makes a query to i, for every $\mathbf{z}_{\mathcal{L}(u)}$ such that $z_j \in D_j(u)$ for $j \in \mathcal{L}(u)$, for every x_i and y_i that diverge at u, for each pair of type profiles $\mathbf{x}, \mathbf{y}$ such that $x_j \in D_j(u)$, $y_j \in D_j(u)$ for every agent j and $x_j = y_j = z_j$ for every $j \in \mathcal{L}(u)$.

This is obvious for $x_i = M_i$. We next prove that $x_i = H_i$ implies $f_i(\mathcal{M}_k(\mathbf{x})) \leq f_i(\mathcal{M}_k(\mathbf{y}))$, that immediately implies the desired claim. Let us first consider a node u corresponding to the descending phase of the mechanism. In this case, $x_i = p$, where p is as at node u. Moreover, in all profiles as described above there are at least $\lceil \tau \rceil + k$ 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 $f_i(\mathcal{M}_k(\mathbf{x})) = 0 \leq f_i(\mathcal{M}_k(\mathbf{y}))$ for every alternative profile $\mathbf{y}$.

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

Observe that for every $\mathbf{x}$ as described above, it must be the case that $x_j \geq y_i$ for every $j \in A \setminus \mathcal{L}(u)$. Hence, we distinguish two cases: if $\min_{j \in A \setminus \mathcal{L}(u)} x_j = x_i$, then $f_i(\mathcal{M}_k(\mathbf{x})) = f_i^\star(x_i, \mathbf{z}_{\mathcal{L}(u)}, \hat{\mathbf{z}}_{-i,\mathcal{L}(u)}) \leq f_i^\star(y_i, \mathbf{z}_{\mathcal{L}(u)}, \hat{\mathbf{z}}_{-i,\mathcal{L}(u)}) = f_i(\mathcal{M}_k(\mathbf{y}))$; if instead $\min_{j \in A \setminus \mathcal{L}(u)} x_j = x_k$, for some $k \neq i$, then

$$\begin{aligned} f_i(\mathcal{M}_k(\mathbf{x})) = f_i^\star(x_k, \mathbf{z}_{\mathcal{L}(u)}, \hat{\mathbf{z}}_{-k,\mathcal{L}(u)}) &\leq f_k^\star(x_k, \mathbf{z}_{\mathcal{L}(u)}, \hat{\mathbf{z}}_{-k,\mathcal{L}(u)}) \\ &\leq f_i^\star(y_i, \mathbf{z}_{\mathcal{L}(u)}, \hat{\mathbf{z}}_{-i,\mathcal{L}(u)}) = f_i(\mathcal{M}_k(\mathbf{y})), \end{aligned}$$

where we used that $\hat{\mathbf{z}}_{-k,\mathcal{L}(u)} = \hat{\mathbf{z}}_{-i,\mathcal{L}(u)}$ and the inequalities follow since: (i) in the optimal outcome the fastest machine must receive at least as many jobs as slower machines; (ii) in the optimal outcome, given the speeds of other machines, the number of jobs assigned to machine i decreases as its speeds decreases. □

Proposition 2. *Mechanism $\mathcal{M}_k$ is $\left(\frac{m+k+\lceil \tau \rceil - 1}{m} \lceil \tau \rceil\right)$-approximate.*

Proof (Sketch). 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 n machines and if it is computed and on $\lceil\tau\rceil + k$ machines, since these cases are distinguished through the input profile.

Fix a type profile $\mathbf{x}$. Let A and B as at the end of the mechanism when agents behave according to $\mathbf{x}$. Let β be the smallest multiple of $|A|$ such that $\beta \geq \sum_{i\in A} OPT_i(\mathbf{x})$. Moreover, let $t = \min_{j\notin A} t_j$. We define the profile $\mathbf{y}$ as follows: $y_i = w$ for every $i \in A$ and $y_i = t$ otherwise, where w is chosen so that $\frac{\beta}{|A|} \cdot w = \max_{j\in A} (x_j \cdot OPT_j(\mathbf{x}))$. Consider then the assignment $\mathbf{a}$ that assigns β jobs equally split among agents in A and $m - \beta$ jobs equally split among agents not in A. It is immediate to see that $OPT(\mathbf{x}) \geq MS(\mathbf{a}, \mathbf{y})$, where $MS(\mathbf{a}, \mathbf{y})$ is the makespan of the assignment $\mathbf{a}$ with respect to the type profile $\mathbf{y}$.

Let $\mathcal{M}(\mathbf{x})$ be the makespan of the assignment returned by our mechanism if agents behave according to $\mathbf{x}$. Then, $\mathcal{M}(\mathbf{x})$ is equivalent to $OPT(\hat{\mathbf{z}})$, where $\hat{\mathbf{z}}$ is such that $\hat{z}_j = x_j$ for $j \in B$ and $\hat{z}_j = t$ for $j \in A \setminus B$. Let α be the smallest multiple of $|B|$ such that $\alpha \geq \sum_{i\in B} OPT_i(\hat{\mathbf{z}})$. We define the profile $\hat{\mathbf{y}}$ as follows: $\hat{y}_i = \hat{w}$ for every $i \in B$ and $y_i = t$ otherwise, where $\hat{w}$ is chosen so that $\frac{\alpha}{|B|} \cdot \hat{w} = \max_{j\in B} (x_j \cdot OPT_j(\hat{\mathbf{z}}))$. Consider then the assignment $\hat{\mathbf{a}}$ that assigns α jobs equally split among agents in B and $m - \alpha$ jobs equally split among agents in $A \setminus B$. It is immediate to see then $\mathcal{M}(\mathbf{x}) = OPT(\hat{\mathbf{z}}) = MS(\hat{\mathbf{a}}, \hat{\mathbf{y}})$. The theorem then follows, since it occurs that $\frac{OPT(\hat{\mathbf{y}})}{MS(\mathbf{a},\mathbf{y})} \leq \frac{m+k+\lceil\tau\rceil-1}{m}\lceil\tau\rceil$. □

The next corollary follows by simple algebraic manipulations.

Corollary 1. *Mechanism $\mathcal{M}_k$ is $(\lceil\tau\rceil + 1)$-approximate for $m > \lceil\tau\rceil (k + \lceil\tau\rceil)$ and the approximation tends to $\lceil\tau\rceil$ as m increases.*

4 Conclusions

We have studied the relationship between the bounded rationality of the agents and the approximation guarantee of mechanisms incentivizing these agents. We have relaxed the popular notion of OSP [18] to allow for more fine grained notions of rationality. For machine scheduling, we proved that more rational agents do not help in getting close to the optimum, unless the level of rationality is significant to a point where the meaning of bounded becomes questionable. On one hand, our findings motivate the focus on OSP for future work on the approximation guarantee of mechanisms for agents with bounded rationality. On the other hand, one might wonder whether similar results hold also for different optimization problems. To this aim, we observe that the techniques that we use in our proof have a resemblance with the ones used in [13] for proving the inapproximability of OSP mechanisms for the facility location problem (with money). Hence, we believe that results similar to the ones we give for machine scheduling may be proved for facility location. As for other problems, we highlight that no approximation result is known even for OSP mechanisms. In particular, for binary allocation problems (that have been considered already in [18]), only a characterization of optimal OSP mechanism is known.

References

1. Adamczyk, M., Borodin, A., Ferraioli, D., de Keijzer, B., Leonardi, S.: Sequential posted price mechanisms with correlated valuations. In: Markakis, E., Schäfer, G. (eds.) WINE 2015. LNCS, vol. 9470, pp. 1–15. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48995-6_1
2. Archer, A., Tardos, É.: Truthful mechanisms for one-parameter agents. In: FOCS 2001, pp. 482–491 (2001)
3. Ashlagi, I., Gonczarowski, Y.A.: Stable matching mechanisms are not obviously strategy-proof. J. Econ. Theor. **177**, 405–425 (2018)
4. Ausubel, L.M.: An efficient ascending-bid auction for multiple objects. Am. Econ. Rev. **94**(5), 1452–1475 (2004)
5. Babaioff, M., Immorlica, N., Lucier, B., Weinberg, S.M.: A simple and approximately optimal mechanism for an additive buyer. In: FOCS 2014, pp. 21–30 (2014)
6. Bade, S., Gonczarowski, Y.A.: Gibbard-Satterthwaite success stories and obvious strategyproofness. In: EC 2017, p. 565 (2017)
7. Chawla, S., Hartline, J., Malec, D., Sivan, B.: Multi-parameter mechanism design and sequential posted pricing. In: STOC 2010, pp. 311–320 (2010)
8. Correa, J., Foncea, P., Hoeksma, R., Oosterwijk, T., Vredeveld, T.: Posted price mechanisms for a random stream of customers. In: EC 2017, pp. 169–186 (2017)
9. De Groot, A.: Thought and Choice in Chess. Mouton, Oxford (1978)
10. Eden, A., Feldman, M., Friedler, O., Talgam-Cohen, I., Weinberg, S.M.: A simple and approximately optimal mechanism for a buyer with complements. In: EC 2017, pp. 323–323 (2017)
11. Feldman, M., Fiat, A., Roytman, A.: Makespan minimization via posted prices. In: EC 2017, pp. 405–422 (2017)
12. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: Obviously strategyproof mechanisms for machine scheduling. In: ESA 2019 (2019)
13. Ferraioli, D., Ventre, C.: Obvious strategyproofness needs monitoring for good approximations. In: AAAI 2017, pp. 516–522 (2017)
14. Ferraioli, D., Ventre, C.: Probabilistic verification for obviously strategyproof mechanisms. In: IJCAI 2018 (2018)
15. Ferraioli, D., Ventre, C.: Obvious strategyproofness, bounded rationality and approximation. In: AAMAS 2019 (2019)
16. Hartline, J., Roughgarden, T.: Simple versus optimal mechanisms. In: EC 2009, pp. 225–234 (2009)
17. Kyropoulou, M., Ventre, C.: Obviously strategyproof mechanisms without money for scheduling. In: AAMAS 2019 (2019)
18. Li, S.: Obviously strategy-proof mechanisms. Am. Econ. Rev. **107**(11), 3257–87 (2017)
19. Mackenzie, A.: A revelation principle for obviously strategy-proof implementation. Research Memorandum 014, (GSBE) (2017)
20. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading (1984)

21. Pycia, M., Troyan, P.: Obvious dominance and random priority. In: EC 2019 (2019)
22. Sandholm, T., Gilpin, A.: Sequences of take-it-or-leave-it offers: near-optimal auctions without full valuation revelation. In: Faratin, P., Parkes, D.C., Rodríguez-Aguilar, J.A., Walsh, W.E. (eds.) AMEC 2003. LNCS (LNAI), vol. 3048, pp. 73–91. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25947-3_5
23. Shannon, C.: Programming a computer for playing chess. Philos. Mag. **41**(314), 256–275 (1950)
24. Zhang, L., Levin, D.: Bounded rationality and robust mechanism design: an axiomatic approach. Am. Econ. Rev. **107**(5), 235–39 (2017)