Theory of Computing Systems (2022) 66:696–720
https://doi.org/10.1007/s00224-022-10071-2

Obvious Strategyproofness, Bounded Rationality and Approximation

Diodato Ferraioli[1] · **Carmine Ventre**[2]

Accepted: 10 January 2022

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 two paradigmatic problems: machine scheduling and facility location. We prove that, for both problems, "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

This article belongs to the Topical Collection: *Special Issue on Algorithmic Game Theory (SAGT 2019)*
Guest Editors: Dimitris Fotakis and Vangelis Markakis

Some of the results appeared in [1]. An extended abstract also appeared as [2].

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

Carmine Ventre is partially supported by EPSRC grant EP/V00784X/1.

✉ Diodato Ferraioli
dferraioli@unisa.it

Carmine Ventre
carmine.ventre@kcl.ac.uk

[1] Università degli Studi di Salerno, Fisciano, SA, Italy

[2] King's College London, London, UK

1 Introduction

A large body of work in computer science deals with the design of algorithms that return "good" (e.g., optimal) solutions. However, there are settings where the input to the algorithm is not readily available since it is owned by *selfish agents*. These agents might misguide the designer's algorithm if it is profitable for them to do so. For example, consider the facility location problem where the location of a facility needs to be chosen, with the goal to serve a group of users and the choice depending on these users' private locations. If a user could bring the facility closer to them by misreporting their location, then they may lie and obtain a better solution for themselves (possibly making the solution worse overall). Therefore, the algorithm needs to be designed to avoid providing such incentives, i.e., to be *incentive compatible*.

The tool typically used to deal with these issues is mechanism design — *strategyproof* (SP) (a.k.a., *truthful*) mechanisms guaranteeing that the algorithm and the agents' incentives are compatible.

Strategyproofness is based on the assumption of full rationality: agents are assumed to be able to consider all possible strategies and their combinations when they reason about their incentives. 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 [3]. One then needs a different notion to ensure that the algorithm is not misguided even 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 *performance* of the mechanisms?

The first question has been recently addressed by Li [4], who defines the concept of *obvious strategyproofness* (OSP-ness); this notion has attracted quite a lot of interest in the community [5–15]. According to this notion, the reasoning of each agent when choosing a strategy is simple:

the *worst* possible outcome that she can get when behaving well must be at least as good as the *best* outcome when misbehaving. Best/Worst are quantified over *all* the possible decisions taken by the agents after the decision of i. Li [4] proves that this solution concept identifies the algorithms that are incentive compatible for agents that have *no* contingent reasoning skills, a special model of bounded rationality: rather than requiring a careful case analysis, OSP mechanisms guarantee that honest behavior is the best strategy to follow no matter all the contingencies.

In this work we focus on the second question, using the OSP formalization of bounded rationality. 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 are considered in the agents' decision making process. On the other hand, OSP mechanisms appear to be quite limited, with respect to SP ones, in terms of their approximation guarantee [7, 12]. 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?*

1.1 Our Contribution

Lookahead can be defined as the ability to anticipate future events (e.g., the moves of a competitor). This concept is discussed in the literature in the context of (strategies to play) games, and agents with limited computational capabilities. De Groot [16] found that all chess players (of whatever standard) used essentially the same thought process – one based upon a lookahead heuristic. Shannon [17] 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 [18] 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 this 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. (A more technical discussion of our notion is deferred to Section 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 on two well-studied problems, machine scheduling and facility location. The former considers n agents who control related machines (i.e., machines have a private job-independent speed). The objective is to schedule a set of m (identical) jobs to the machines in order to minimize the *makespan* (i.e., the latest machine's completion time). Our technical contribution consists in proving a lower bound on the 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 [19]. We also show that this bound is tight up to a constant factor, at least for the case in which each machine can report a speed from a domain of three values. (Such a restriction is common to the state of the art of OSP mechanisms [12, 14].) 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., the protocol between the designer and the agents induced by the mechanism) 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 facility location problem considers n agents who are located on the real line, and the goal is to place a facility on the line so to minimize the total distance of the facility to agents. Here, agents have a private location in mind and can misreport it in the attempt to force the facility to be as close as possible to their own location. It is well known that a strategy-proof mechanism exists for this problem [20], even without money, whereas from [7], we know that no OSP mechanisms can perform better than a dictatorial mechanism. Here we show that the arguments therein can be generalized to prove a lower bound of $\frac{n-k-1}{k+1}$ to the approximation ratio of k-OSP mechanisms, whenever $k \leq \lceil \frac{n}{2} \rceil - 2$.

The main message of our work is that having more rational agents only slightly improves the approximation guarantee of incentive-compatible mechanisms, at least for these two optimization problems. In fact, a constant approximation of the optimum would require agents with very large lookahead. We can 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.

1.2 Related Work

Recent research in algorithmic mechanism design has suggested to focus on "simple" mechanisms to deal with bounded rationality [21–23]. OSP provides a formal definition for simple mechanisms, by focusing on imperfect rationality due to no contingent reasoning (see references above for the body of work on this notion).

Other 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 [24–28]. In these mechanisms one's own bid is immaterial for the price paid to get some goods of interest – this immediately suggests that trying to play the mechanism is worthless no matter the cognitive abilities of the agents.

A different approach is that of verifiably truthful mechanisms [29], wherein agents can run some algorithm to effectively check that the mechanism is incentive compatible. However, they seem to transfer the question from the mechanism itself to the verification algorithm.

The concept of "guided" implementation [30] shares some of the motivations and philosophy behind OSP. The authors note that the implementation (via normal-form games) of a social choice function is 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. In this context, backwards induction implementation via an extensive game is equivalent to such a guided implementation via a normal-form game.

2 OSP with Lookahead

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 (also called social choice function) 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 $S(u)$, and the outgoing edges from u are labelled with the subset of types in the domain of $S(u)$ that are compatible with the history leading to u; in particular, the labels of edges going out from u denote a partition of the domain of $S(u)$ compatible with the history. If at node u there are multiple outgoing edges, then $S(u)$ is called a *divergent agent* at u. 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_i has utility

$$u_i(b_i, \mathbf{b}_{-i}) = p_i(\mathbf{b}) - t(f(\mathbf{b})),$$

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 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

[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.

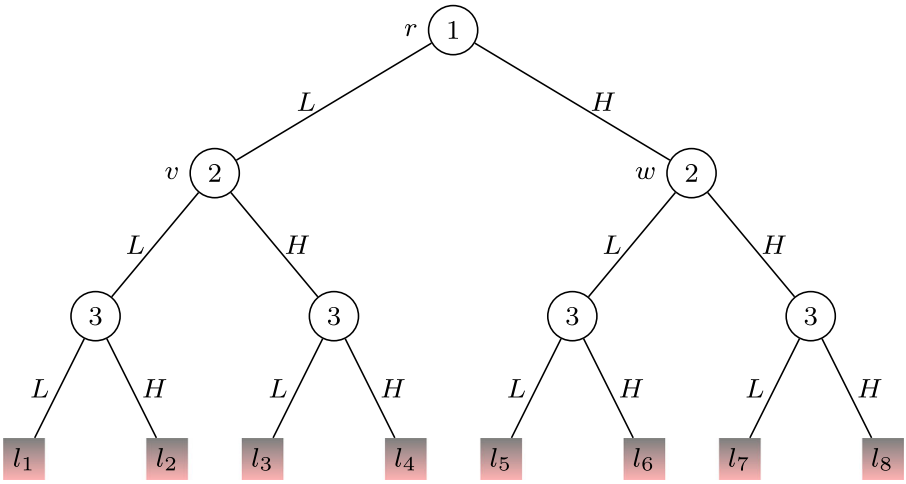

Fig. 1 An implementation tree with three players with two-value domains $\{L, H\}$; each player separates the domain types upon playing; at each leaf l_i the mechanism computes $f(\mathbf{b})$ and $p(\mathbf{b})$, $\mathbf{b}$ being the bid vector at l_i

respectively, then I prefer L, too; and so on". In OSP, the thinking process of the agents is not finely-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 for the agents to have some limited ability of performing 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). For example, (L, H, H) and (L, L, H) are compatible with node v on Figure 1 and diverge at that node, whilst (L, L, H) and (L, L, L) are compatible with v but do not diverge at v.

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 nodes 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 $Ł_k(u^i_{t,b})$, 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 $Ł(u)$ be the lookahead of agent $S(u)$ at u.)

Definition 1 (k-OSP) An extensive-form mechanism $\mathcal{M} = (f, \mathcal{T}, p)$ is OSP with k-lookahead (k-OSP, for short) given $Ł_k(u^i_{t,b})$, 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 $K = Ł_k(u^i_{t,b})$, $T = N \setminus (K \cup \{i\})$ and $D_A(u) = \times_{j \in A \subset N} D_j(u)$.

In words, a mechanism is OSP with lookahed 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 the 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 $Ł_k(u)$ in the definition above crucially captures our notion of lookahead. We highlight the following features of our definition.

– The size of set $Ł_k(u)$ 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 Figure 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 $Ł_1(r) = \{2\}$ 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 [16]. 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 $Ł_k(u)$ 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 $Ł_k(u) = Ł_k(v)$ 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 $Ł_k(u)$ 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 $Ł_k(u)$ also allows us to overcome a paradox that would arise if one defined the lookahead set 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 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, but i has no hint about the actions of j if she chooses $k \neq j$ as the next player). 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.

It must, however, 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 bounds continue 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 [9]. 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 bounds below (even if they were so rational to do that, the approximation guarantee would still 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.,

$$f^*(\mathbf{b}) \in \arg\min_{\mathbf{x}} \max_{i=1}^{n} b_i(\mathbf{x});$$

We say that 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 the function of n for which it holds that $n = \tau_k(n)(\tau_k(n)+k)$. Observe that $\tau_0(n) = \sqrt{n}$ and $\tau_{n-1}(n) = 1$. Henceforth, for sake of readability, let us denote $\tau := \tau_k(n)$. Then, we can prove the following theorem.

Theorem 1 *For the machine scheduling problem, no k-OSP mechanism can be better than τ-approximate, regardless of the value of the sets $Ł_k(\cdot)$. 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 $H \geq \tau \cdot mM$. Intuitively, these values tell us that no τ-approximate mechanism should put a job on a machine with type M if there are at least $\lfloor \tau \rfloor$ machines with type L. Similarly, no τ-approximate mechanism should put a job on a machine with type H if there is at least one machine with type M or smaller.

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 is the largest integer such that $k \leq \frac{n-c^2}{c}$.*

Proof We first show that $\tau \in [c, c+1)$. Observe that τ is decreasing in k, since $\frac{\partial \tau}{\partial k} = -\frac{1}{2}\left(1 - \frac{k}{\sqrt{k^2+4n}}\right)$ that is negative for every $k \geq 0$. Hence, from $k \leq \frac{n-c^2}{c}$, we get that

$$\tau \geq \frac{1}{2}\left(\sqrt{\frac{(n-c^2)^2}{c^2} + 4n} - \frac{n-c^2}{c}\right) = \frac{\sqrt{(n+c^2)^2} - n + c^2}{2c} = c.$$

In a similar way, using that $k > \frac{n-(c+1)^2}{c+1}$, we conclude that $\tau < c+1$.

Next we prove that $\delta = \tau - c \leq \frac{k}{\tau+k-1}$. That is, we need to prove that $\frac{\sqrt{k^2+4n}-k-2c}{2} \leq \frac{2k}{\sqrt{k^2+4n}+k-2}$. By simple algebraic manipulation, we obtain that this is equivalent to proving that $2(n-c) - (c+1)k \leq (c+1)\sqrt{k^2+4n}$. Since the r.h.s. is non-negative the claim trivially holds if the l.h.s. is non-positive. If instead the l.h.s. is positive, then we square both sides. By rearranging, our requirement

becomes

$$n^2 - ((c+1)(c+1+k) + 2c)\, n + c\,((c+1)k + c) \leq 0.$$

By solving this inequality, we obtain that

$$c - (c+1)\frac{\lambda}{2} \leq n \leq c + (c+1)\left(c + 1 + k + \frac{\lambda}{2}\right), \tag{1}$$

where $\lambda = \sqrt{(c+1+k)^2 + 4c} - (c+1+k) > 0$. The first inequality in (1) is always verified. Indeed, since $0 \leq k \leq \frac{n-c^2}{c}$, we have that $c \leq \sqrt{n}$, and thus $c - (c+1)\frac{\lambda}{2} < c \leq \sqrt{n} \leq n$. As for the second inequality, we note that it always hold since $k > \frac{n-(c+1)^2}{c+1}$, and thus

$$c + (c+1)\left(c + 1 + k + \frac{\lambda}{2}\right) > c + (c+1)\left(c + 1 + \frac{n-(c+1)^2}{c+1}\right) = c + n > n.$$

This concludes the proof. □

Suppose now that a mechanism $\mathcal{M}$ with approximation ratio $\rho < \tau$ exists for the setting described above, and let $\mathcal{T}$ be its implementation tree. Observe that in $\mathcal{T}$ there is at least one divergent agent, otherwise the mechanism has an approximation ratio larger than ρ: indeed, such a mechanism must return the same outcome for the instance where all machines have type H (for which assigning all jobs to the same machine will surely give an approximation ratio worse than ρ), and the instance whereone machine has type L and the remaining have type H (for which every allocation that does not assign all the jobs to the machine of type L has approximation ratio worse than ρ). Let us rename the agents as follows: Agent 1 is the first agent that diverges in $\mathcal{T}$. We now call agent 2, the first agent different from 1 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, we rename all agents as follows: agent i is the first agent different from $1, 2, \ldots, i-1$ 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 does not diverge (i.e., 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 $Ł(u_i)$ 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 $j < i$ or $j \in Ł(u_i)$, and $y_j = L$ for every $j > i$ such that $j \notin Ł(u_i)$. (We stress that our lower bound holds no matter the definition of the sets $Ł(u_i)$.) Note that there are $n - (i-1) - k$ machines j such that $y_j = L$. 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 $OPT(\mathbf{y}) = \left\lceil \frac{m}{n-i-k+1} \right\rceil L$: indeed, the optimal allocation cannot assign any job to machines j such that $y_j = H$ otherwise the allocation cost would be at least $H > \tau \cdot mL$; moreover, if m jobs must be assigned to at most $n - (i-1) - k$ machines, then there must exist at least one of these machines that receives at least $\left\lceil \frac{m}{n-i-k+1} \right\rceil$ jobs. 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 it would be $mL \geq \tau \left\lceil \frac{m}{n-i-k+1} \right\rceil L > \rho \cdot OPT(\mathbf{y})$, where we used that

$$
\begin{aligned}
\tau \left\lceil \frac{m}{n-i-k+1} \right\rceil &\leq \tau \left\lceil \frac{n}{\lceil \tau \rceil} \right\rceil && \text{(since } i \leq n+1-\lceil \tau \rceil - k\text{)} \\
&= \tau \left\lceil \frac{\tau(\tau+k)}{\tau+1-\delta} \right\rceil && \text{(by def. of } \delta\text{)} \\
&\leq \tau \frac{\tau(\tau+k)+(\tau-\delta)}{\tau+1-\delta} \\
&= \tau \left((\tau+k) - \frac{k-\delta(\tau+k-1)}{\tau+1-\delta} \right) \\
&\leq \tau(\tau+k) = m && \text{(since, by Lemma 1, } \delta \leq \frac{k}{\tau+k-1}\text{)}.
\end{aligned}
$$

Hence, for the mechanism to be k-OSP we need that both the following conditions are satisfied:

$$
\begin{aligned}
&(i) \quad p_i(\mathbf{x}) - mM \geq p_i(\mathbf{y}) - \mu M, \text{ and} \\
&(ii) \quad p_i(\mathbf{y}) - \mu L \geq p_i(\mathbf{x}) - mL,
\end{aligned}
$$

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. Indeed, since both $\mathbf{x}$ and $\mathbf{y}$ are compatible with u then condition (i) ensures that when agent i has type M ($x_i = M$) then she has no incentive to deviate and signal type y_i. Similarly, condition (ii) assumes that the type of agent i is $y_i = L$ and thus we should have that i has no incentive to deviate and signal type x_i. However, conditions (i) and (ii) can be both satisfied only if $\mu(M - L) \geq m(M - L)$ that leads to the contradiction that $L \geq M$, since $\mu < m$.

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 $j \in Ł(u_i)$, such that if i takes an action signalling type H, then $\mathcal{M}$ assigns $\mu \geq 1$ jobs 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 $j < i$ or $j \in Ł(u_i)$, and $y_j = L$ for $j > i$ such that $j \notin Ł(u_i)$. 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

$$\begin{aligned} M &\geq \tau \left\lceil \frac{m}{\lfloor \tau \rfloor} \right\rceil L && \text{(by definition of M)} \\ &\geq \tau \left\lceil \frac{m}{n-i-k} \right\rceil L && \text{(since} i \leq n - \lceil \tau \rceil - k) \\ &> \rho \cdot OPT(\mathbf{y}). && \text{(since } \rho < \tau \text{ and by def. of } OPT(\mathbf{y})) \end{aligned}$$

Hence, for the mechanism to be k-OSP we need that both the following conditions are satisfied:

$$\begin{aligned} (i) \quad & p_i(\mathbf{x}) - \mu H \geq p_i(\mathbf{y}) - 0, \text{ and} \\ (ii) \quad & p_i(\mathbf{y}) - 0 \geq p_i(\mathbf{x}) - \mu M. \end{aligned}$$

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 $\mathcal{M}$ was k-OSP and ρ-approximate.

To this aim, assume first that each agent $i \leq n - \lfloor \tau \rfloor - k$ diverges at u_i, i.e., they do not get id arbitrarily. 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 (since the mechanism has finite approximation then $i > 1$). This means that all the machines $j \geq i$ will not diverge at u_i; let S denote this set of machines. Note that the $n - i + 1 \geq \lfloor\tau\rfloor + k + 1$ machines in S will have the same outcome no matter their types when the machines not in S have type H; in other words, any profile $\mathbf{x}$ where $x_j = H$ for $j \notin S$ is compatible with u_i. Consider $\mathbf{x}$ such that $x_j = H$ for $j \notin S$ and $x_j = L$ otherwise. Since $H \geq \tau n^2 L$, to guarantee approximation ρ, 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 $\left\lceil \frac{n}{|S|} \right\rceil$ jobs. Consider now $\mathbf{y}$ where $y_j = H$ for $j \notin S \cup \{j^*\}$ and $x_j = L$ otherwise. For this type profile, the mechanism must return the same allocation returned when the type profile is $\mathbf{x}$, since it cannot distinguish $\mathbf{x}$ from $\mathbf{y}$. Hence, the allocation returned by the mechanism on type profile $\mathbf{y}$ must have cost at least $\left\lceil \frac{n}{|S|} \right\rceil H \geq H$. However, since $|S| < n$, then there is at least one machine with type L, and thus the optimal allocation would have cost at most $mL < \frac{1}{\tau} H < \frac{1}{\rho} H$, thus contracting the approximation guarantee. □

3.2 Upper Bound

We known that the bound τ is tight for $k = n - 1$ [19] and for $k = 0$ [12]. We next show that for every remaining value of k and every possible choice of lookahead sets $\{Ł_k(u)\}_{u \in \mathcal{T}}$, the bound above is tight up to a constant factor 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 (Algorithm1) 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 ∈ D_i} for every i
2 while |A| > ⌈τ⌉ + k do
3     Set p = max_{a∈A}{t_a} and i = min{a ∈ A : 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 ∈ D_i : 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; the algorithm then proceeds by removing from A the slowest machines, until there are only $\lceil\tau\rceil + k$ left. Note that type domains are heterogeneous, and no assumption is made about the relation existing among types of different agents: it might for example be that $H_j < L_i$ for two given machines i and j. Hence, in order to correctly remove from A the slowest machines, we need to iterate over the set $\bigcup_i D_i$ of all possible type values, and to query machines with the selected value – we denote with p the variable implementing this iterator.

1 Set $s_i = \min\{d \in D_i\}$ for every i
2 Set $B = \emptyset$
3 **while** $|B| \leq k$ **do**
4 Set $p = \min_{a \in A \setminus B}\{s_a\}$ and $i = \min\{a \in A \setminus B : s_a = p\}$
5 Ask machine i if her type is equal to p
6 **if** *yes* **then** Set $s_i = p$ and insert i in B
7 **else** set $s_i = \min\{d \in D_i : d > p\}$
8 Consider a profile $\hat{\mathbf{z}}$ over machines in A only, such that $\hat{z}_i = s_i$ for $i \in B$ and $\hat{z}_j = \min_{w \notin A} t_w$ for $j \in A \setminus B$
9 Let $f^\star(\hat{\mathbf{z}}) = \left(f_i^\star(\hat{\mathbf{z}})\right)_{i \in A}$ be the assignment returned by the optimal greedy algorithm on input profile $\hat{\mathbf{z}}$
10 Assign $f_j^\star(\hat{\mathbf{z}})$ jobs to each machine $j \in A$ and a payment $M_j \cdot f_j^\star(\hat{\mathbf{z}})$

Algorithm 2 The ascending phase adds to B the k fastest machines and keeps in s_i the minimum non-discarded type for each agent; 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. As above, p denotes the iterator needed to sweep through the domains.

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 $n - \lceil \tau \rceil - k$ slowest machines and discarding them (i.e., no job will be assigned to these machines).

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)[2].

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 $\mathbf{z}_{\text{Ł}(u)}$ such that $z_j \in D_j(u)$ for $j \in \text{Ł}(u)$, 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 $j \in \text{Ł}(u)$.

[2]Note that profile $\hat{\mathbf{z}}$ is not necessarily a feasible profile since $\min_{w \notin A} t_w$ may not belong to D_j for some $j \in A \setminus B$.

This is obvious for $x_i = M_i$. Next we prove that $f_i(\mathcal{M}_k(\mathbf{x})) \leq f_i(\mathcal{M}_k(\mathbf{y}))$ if $x_i = H_i$, 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 $f_i(\mathcal{M}_k(\mathbf{y})) = f_i^\star(y_i, \mathbf{z}_{\text{Ł}(u)}, \hat{\mathbf{z}}_{-i,\text{Ł}(u)})$, where $f_i^\star(y_i, \mathbf{z}_{\text{Ł}(u)}, \hat{\mathbf{z}}_{-i,\text{Ł}(u)})$ is the number of jobs assigned to machine i by the optimal outcome when input consists of profile $(y_i, \mathbf{z}_{\text{Ł}(u)}, \hat{\mathbf{z}}_{-i,\text{Ł}(u)})$, $\hat{\mathbf{z}}_{-i,\text{Ł}(u)}$ being such that $\hat{z}_j = \max_{k \in A} t_k$ for every $j \in A \setminus (\{i\} \cup \text{Ł}(u))$.

Observe that for every $\mathbf{x}$ as described above, it must be the case that $x_j \geq y_i$ for every $j \in A \setminus \text{Ł}(u)$. Hence, we distinguish two cases: if $\min_{j \in A \setminus \text{Ł}(u)} x_j = x_i$, then

$$f_i(\mathcal{M}_k(\mathbf{x})) = f_i^\star(x_i, \mathbf{z}_{\text{Ł}(u)}, \hat{\mathbf{z}}_{-i,\text{Ł}(u)}) \leq f_i^\star(y_i, \mathbf{z}_{\text{Ł}(u)}, \hat{\mathbf{z}}_{-i,\text{Ł}(u)}) = f_i(\mathcal{M}_k(\mathbf{y}));$$

if instead $\min_{j \in A \setminus \text{Ł}(u)} x_j = x_k$, for some $k \neq i$, then

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

where we used that $\hat{\mathbf{z}}_{-k,\text{Ł}(u)} = \hat{\mathbf{z}}_{-i,\text{Ł}(u)}$ 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 $\mathcal{M}_k$, for $1 \leq k \leq n-2$, is $\left(2\frac{m+2k+\lceil\tau\rceil-1}{m}\lceil\tau\rceil\right)$-approximate.*

Proof Fix a type profile $\mathbf{x}$. We will denote with $OPT(\mathbf{x})$ the makespan of the assignment returned by the optimal greedy algorithm on input profile $\mathbf{x}$.[3] We also denote as $OPT_i(\mathbf{x})$ the number of jobs assigned to machine i in the optimal makespan, so that $OPT(\mathbf{x}) = \max_i x_i \cdot OPT_i(\mathbf{x})$.

The proof proceeds by comparing the makeskan $OPT(\mathbf{x})$ of the optimal allocation for profile $\mathbf{x}$ and the makespan of the allocation returned by the mechanism for $\mathbf{x}$. Rather than trying to directly characterize these assignments, our proof considers two simpler instances, each involving only two different types, and two special allocations for these instances that turn out to be related to $OPT(\mathbf{x})$ and the allocation returned by the mechanism. These proxy instances and solutions simplify our analysis.

[3]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.

Specifically, let A and B as at the end of the mechanism when agents behave according to $\mathbf{x}$. Let α be the smallest multiple of $|A|$ such that $\alpha \geq \sum_{i \in A} OPT_i(\mathbf{x})$. 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 $\frac{\alpha}{|A|} \cdot w = \max_{j \in A} \left(x_j \cdot OPT_j(\mathbf{x})\right)$.[4] Consider then the assignment $\mathbf{a}$ that assigns α jobs equally split among agents in A and $m - \alpha$ 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 $\mathbf{y}$: indeed, if $MS(\mathbf{a}, \mathbf{y}) = \frac{\alpha}{|A|} w$, then we have, by our choice of w, that $MS(\mathbf{a}, \mathbf{y}) = \max_{j \in A} \left(x_j \cdot OPT_j(\mathbf{x})\right) \leq OPT(\mathbf{x})$; if $MS(\mathbf{a}, \mathbf{y}) = \left\lceil \frac{m-\alpha}{n-|A|} \right\rceil t$, then, by our choice of α, $MS(\mathbf{a}, \mathbf{y}) \leq \left\lceil \frac{m-\sum_{i \in A} OPT_i(\mathbf{x})}{n-|A|} \right\rceil t \leq OPT(\mathbf{x})$, where the last inequality follows since in the optimal assignment there must be at least one machine not in A that receives at least $\left\lceil \frac{m-\sum_{i \in A} OPT_i(\mathbf{x})}{n-|A|} \right\rceil$ jobs, and its type is at least t.

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 $\overline{\beta}$ be the largest multiple of $|A \setminus B|$ such that $\overline{\beta} \leq \sum_{i \in B} OPT_i(\hat{\mathbf{z}})$. Moreover, let β be the smallest multiple of $|B|$ such that $\beta \geq \overline{\beta}$. Note that $\beta - \overline{\beta} \leq |B| - 1 = k$. We define the profile $\hat{\mathbf{y}}$ as follows: $\hat{y}_i = \hat{w}$ for every $i \in B$ and $\hat{y}_i = t$ otherwise, where $\hat{w}$ is chosen so that $\frac{\beta}{|B|} \cdot \hat{w} = \max_{j \in B} \left(x_j \cdot OPT_j(\hat{\mathbf{z}})\right)$. Consider then the assignment $\hat{\mathbf{a}}$ that assigns β jobs equally split among agents in B and $m - \overline{\beta}$ jobs equally split among agents in $A \setminus B$.[5] Observe that $\max_{j \in A \setminus B}(x_j \cdot OPT_j(\hat{\mathbf{z}})) = \left\lceil \frac{m-\sum_{i \in B} OPT_i(\mathbf{z})}{|A \setminus B|} \right\rceil \cdot t = \frac{m-\overline{\beta}}{|A \setminus B|} \cdot t = \max_{j \in A \setminus B}(\hat{y}_j \cdot \hat{a}_j)$. Moreover, $\max_{j \in B}(x_j \cdot OPT_j(\hat{\mathbf{z}})) = \frac{\beta}{|B|} \cdot \hat{w} = \max_{j \in B}(\hat{y}_j \cdot \hat{a}_j)$. Hence, $\mathcal{M}(\mathbf{x}) = OPT(\hat{\mathbf{z}}) = MS(\hat{\mathbf{a}}, \hat{\mathbf{y}})$.

The theorem then follows, by proving that $\frac{MS(\hat{\mathbf{a}}, \hat{\mathbf{y}})}{MS(\mathbf{a}, \mathbf{y})} \leq 2\frac{m+k+\lceil \tau \rceil - 1}{m} \lceil \tau \rceil$. To this aim, let us first to prove some useful lemmata.

Lemma 3 *$MS(\hat{\boldsymbol{a}}, \hat{\boldsymbol{y}}) = OPT(\hat{\boldsymbol{y}})$, i.e., $\hat{\boldsymbol{a}}$ is the optimal assignment for $\hat{\boldsymbol{y}}$ and maximizes the number of jobs assigned to machines in B.*

Proof Observe that $MS(\hat{\mathbf{a}}, \hat{\mathbf{y}}) = \max \left\{ \frac{\beta}{k+1} \cdot \hat{w}, \frac{m-\overline{\beta}}{\lceil \tau \rceil - 1} \cdot t \right\}$, and suppose it is not optimal for $\hat{\mathbf{y}}$. If $MS(\hat{\mathbf{a}}, \hat{\mathbf{y}}) = \frac{\beta}{k+1} \hat{w}$, then, if $\hat{\mathbf{a}}$ is not optimal for $\hat{\mathbf{y}}$, it must be that by moving one job from every machine in B to machines in $A \setminus B$ we have a smaller

[4] Note that we do not require that $\mathbf{y}$ is feasible: indeed, it is possible that neither w nor t belong to the domain of machines in A.

[5] Note that assignment $\hat{\mathbf{a}}$ may assign more than m jobs.

makespan, i.e. $\left(\frac{m-\overline{\beta}}{\lceil\tau\rceil-1}+\left\lceil\frac{k+1}{|A\setminus B|}\right\rceil\right)\cdot t < \frac{\beta}{k+1}\cdot\hat{w} = \max_{j\in B}\left(x_j\cdot OPT_j(\hat{\mathbf{z}})\right) = \max_{j\in B}\left(\hat{z}_j\cdot OPT_j(\hat{\mathbf{z}})\right)$. However,

$$\left(\frac{m-\beta}{\lceil\tau\rceil-1}+\left\lceil\frac{k+1}{|A\setminus B|}\right\rceil\right)\cdot t = \left(\left\lceil\frac{m-\sum_{i\in B}OPT_i(\hat{\mathbf{z}})}{\lceil\tau\rceil-1}\right\rceil+\left\lceil\frac{k+1}{|A\setminus B|}\right\rceil\right)\cdot t$$
$$= \max_{j\in A\setminus B}\left(\hat{z}_j\cdot\left(OPT_j(\hat{\mathbf{z}})+\left\lceil\frac{k+1}{|A\setminus B|}\right\rceil\right)\right),$$

that implies $\max_{j\in A\setminus B}\left(\hat{z}_j\cdot\left(OPT_j(\hat{\mathbf{z}})+\left\lceil\frac{k+1}{|A\setminus B|}\right\rceil\right)\right) < \max_{j\in B}\left(\hat{z}_j\cdot OPT_j(\hat{\mathbf{z}})\right)$. In words, moving one job from any machine in B to machines in A/B will result in a makespan lower than $OPT(\hat{\mathbf{z}})$, that contradicts the optimality of $OPT(\hat{\mathbf{z}})$.

If instead, $MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) = \frac{m-\overline{\beta}}{\lceil\tau\rceil-1}\cdot t$, then, since $\hat{\mathbf{a}}$ is not optimal for $\hat{\mathbf{y}}$ we have:

$$\left(\frac{\beta}{k+1}+1\right)\hat{w} < \frac{m-\overline{\beta}}{\lceil\tau\rceil-1}\cdot t \quad = \left\lceil\frac{m-\sum_{i\in B}OPT_i(\hat{\mathbf{z}})}{\lceil\tau\rceil-1}\right\rceil\cdot t$$
$$= \max_{j\in A\setminus B}\left(\hat{z}_j\cdot OPT_j(\hat{\mathbf{z}})\right).$$

However, $\left(\frac{\beta}{k+1}+1\right)\hat{w} = \max_{j\in B}\left(x_j\cdot OPT_j(\hat{\mathbf{z}})\right)+\hat{w}$. If $\hat{w}\geq x_{j^*}$, where $j^\star$ is the agent in B that maximizes $x_j\cdot OPT_j(\hat{\mathbf{z}})$, then we conclude that

$$\max_{j\in B}\left(\hat{z}_j\cdot(OPT_j(\hat{\mathbf{z}})+1)\right) < \max_{j\in A\setminus B}\left(\hat{z}_j\cdot OPT_j(\hat{\mathbf{z}})\right),$$

that contradicts either the optimality of $OPT(\hat{\mathbf{z}})$ or the tie-breaking rule of the optimal algorithm.

Suppose instead that $\hat{w} < x_{j^*}$. Then there must exist $j'\in B$ with $x_{j'}\leq\hat{w}$ and $x_{j'}OPT_{j'}(\hat{\mathbf{z}}) < x_{j^*}OPT_{j^*}(\hat{\mathbf{z}})$.

Hence,

$$x_{j'}\left(OPT_{j'}(\hat{\mathbf{z}})+1\right) < \max_{j\in B}\left(x_j\cdot OPT_j(\hat{\mathbf{z}})\right)+\hat{w} < \max_{j\in A\setminus B}\left(\hat{z}_j\cdot OPT_j(\hat{\mathbf{z}})\right),$$

that again proves that moving a job from $A\setminus B$ to B will produce an assignment with at least the same makespan and one more job on the fastest machines, by contradicting either the optimality of $OPT(\hat{\mathbf{z}})$ or the tie-breaking rule of the optimal algorithm.

Finally, by our assumption on the tie-breaking rule used by the mechanism for choosing the optimal outcome, and the definition of β, it follows that $\hat{\mathbf{a}}$ is the optimal assignment for $\hat{\mathbf{y}}$ that maximizes the number of jobs that are assigned to machines in B. □

Lemma 4 *$MS(\hat{\boldsymbol{a}},\hat{\mathbf{y}})\leq\frac{(m+2k+\lceil\tau\rceil-1)\hat{\nu}\hat{\theta}}{\hat{\nu}+\hat{\theta}}$, where $\hat{\nu}=\frac{\hat{w}}{k+1}$ and $\hat{\theta}=\frac{t}{\lceil\tau\rceil-1}$.*

Proof Recall that $MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) = \min\left\{\frac{\beta}{k+1}\cdot\hat{w},\frac{m-\overline{\beta}}{\lceil\tau\rceil-1}\cdot t\right\}$. Suppose first that $MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) = \frac{\beta}{k+1}\cdot\hat{w} = \beta\hat{\nu}$. Since, by Lemma 3, $\hat{\mathbf{a}}$ is the optimal assignment for $\hat{\mathbf{y}}$,

then it must be the case that by moving a job from every machine in B to machines in $A \setminus B$, the makespan does not decrease. Specifically,

$$\beta\hat{\nu} \le \left\lceil \frac{m-\overline{\beta}+k+1}{\lceil\tau\rceil-1} \right\rceil \cdot t = (m-\overline{\beta}+k+1+\delta)\hat{\theta},$$

where δ is the smallest integer such that $m-\beta+k+1+\delta$ is a multiple of $\lceil\tau\rceil-1$. If $\beta=\overline{\beta}$, we have that $\beta \le \frac{(m+k+1+\delta)\hat{\theta}}{\hat{\nu}+\hat{\theta}}$, and, consequently,

$$\begin{aligned} MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) &\le \frac{(m+k+1+\delta)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}} \\ &\le \frac{(m+k+\lceil\tau\rceil-1)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}} \le \frac{(m+2k+\lceil\tau\rceil-1)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}}, \end{aligned}$$

where the second inequality uses that $\delta \le \lceil\tau\rceil-2$. If $\beta > \overline{\beta}$, we have that $\beta \le \frac{(m+(\beta-\overline{\beta})+k+1+\delta)\hat{\theta}}{\hat{\nu}+\hat{\theta}} \le \frac{(m+2k+\delta+1)\hat{\theta}}{\hat{\nu}+\hat{\theta}}$, where we used that $\beta-\overline{\beta} \le k$. Consequently,

$$MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) \le \frac{(m+2k+\delta+1)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}} \le \frac{(m+2k+\lceil\tau\rceil-1)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}},$$

where the second inequality follows from $\delta \le \lceil\tau\rceil-2$.

Suppose instead $MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) = \frac{m-\beta}{\lceil\tau\rceil-1}\cdot t = (m-\beta)\hat{\theta}$. Since, by Lemma 3, $\hat{\mathbf{a}}$ is the optimal assignment for $\hat{\mathbf{y}}$ that maximizes the number of jobs in $|B|$, then moving one job from $A \setminus B$ to a machine in B results in an higher makespan. Specifically,

$$\begin{aligned} (m-\overline{\beta})\hat{\theta} &< \left(\frac{\beta}{k+1}+1\right)\cdot\hat{w} = (\beta+k+1)\hat{\nu} \\ &= (\overline{\beta}+(\beta-\overline{\beta})+k+1)\hat{\nu} \le (\overline{\beta}+2k+1)\hat{\nu}. \end{aligned}$$

Hence, we have $\overline{\beta} > \frac{m\hat{\theta}-(2k+1)\hat{\nu}}{\hat{\nu}+\hat{\theta}}$, and, consequently,

$$\begin{aligned} MS(\hat{\mathbf{a}},\hat{\mathbf{y}}) &< \left(m-\frac{m\hat{\theta}-(2k+1)\hat{\nu}}{\hat{\nu}+\hat{\theta}}\right)\cdot\hat{\theta} = \frac{(m+2k+1)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}} \\ &\le \frac{(m+2k+\lceil\tau\rceil-1)\hat{\theta}\hat{\nu}}{\hat{\nu}+\hat{\theta}}, \end{aligned}$$

where the second inequality uses that $\lceil\tau\rceil-1 \ge 1$ since $k < n-1$. □

Lemma 5 *$MS(\boldsymbol{a},\boldsymbol{y}) \ge \frac{m\nu\theta}{\nu+\theta}$, where $\nu=\frac{w}{\lceil\tau\rceil+k}$ and $\theta=\frac{t}{n-\lceil\tau\rceil-k}$.*

Proof Recall that $MS(\mathbf{a},\mathbf{y}) = \max\left\{\frac{\alpha}{\lceil\tau\rceil+k}\cdot w, \left\lceil\frac{m-\alpha}{n-\lceil\tau\rceil-k}\right\rceil\cdot t\right\}$.

If $MS(\mathbf{a},\mathbf{y}) = \frac{\alpha}{\lceil\tau\rceil+k}\cdot w = \alpha\nu$, then $\alpha\nu \ge \left\lceil\frac{m-\alpha}{n-\lceil\tau\rceil-k}\right\rceil\cdot t = \frac{m-\alpha+\lambda}{n-\lceil\tau\rceil-k}\cdot t = (m-\alpha+\lambda)\theta$, where λ is the smallest integer such that $m-\alpha+\lambda$ is a multiple of $n-\lceil\tau\rceil-k$. Hence, we have $\alpha \ge \frac{(m+\lambda)\theta}{\nu+\theta}$, and, consequently, $MS(\mathbf{a},\mathbf{y}) \ge \frac{(m+\lambda)\theta\nu}{\nu+\theta} \ge \frac{m\theta\nu}{\nu+\theta}$, since $\lambda \ge 0$.

If $MS(\mathbf{a},\mathbf{y}) = \left\lceil \frac{m-\alpha}{n-\lceil\tau\rceil-k} \right\rceil \cdot t = \frac{m-\alpha+\lambda}{n-\lceil\tau\rceil-k} \cdot t = (m-\alpha+\lambda)\theta$, then, $(m-\alpha+\lambda)\theta \geq \alpha\nu$. Hence, $\alpha \leq \frac{(m+\lambda)\theta}{\nu+\theta}$, and, consequently,

$$MS(\mathbf{a},\mathbf{y}) \geq \left((m+\lambda) - \frac{(m+\lambda)\theta}{\nu+\theta}\right)\theta = \frac{(m+\lambda)\nu\theta}{\nu+\theta} \geq \frac{m\nu\theta}{\nu+\theta}.$$

□

Lemma 6 *It holds* $\hat{w} \leq 2w$.

Proof Throughout this proof, let us denote with γ the number of jobs that the optimal allocation for $\mathbf{x}$ assigns to machines in B.

Consider first the simpler case that $\gamma = m$, i.e., the optimal allocation for $\mathbf{x}$ assigns all the jobs to the machines in B. Clearly, in this case $OPT_i(\mathbf{x}) = OPT_i(\hat{\mathbf{z}})$ for every i, and thus $\gamma = \overline{\beta}$. Moreover, by definition of β, $\left\lceil \frac{\gamma}{|B|} \right\rceil = \frac{\beta}{|B|}$. Finally, if j^* is the index of the machine that achieves the makespan in the optimal assignment for type profile $\mathbf{x}$, i.e., $j^* = \arg\max_j (x_j \cdot OPT_j(\mathbf{x}))$, then $j^* \in B$ and

$$\begin{aligned}\hat{w} &= \left(\frac{\beta}{|B|}\right)^{-1} \max_{j\in B}(x_j \cdot OPT_j(\hat{\mathbf{z}})) = \left(\frac{\beta}{|B|}\right)^{-1} (x_{j^*} \cdot OPT_{j^*}(\hat{\mathbf{z}})) \\ &= \left\lceil \frac{\gamma}{|B|} \right\rceil^{-1} \max_{j\in A}(x_j \cdot OPT_j(\mathbf{x})) \leq 2 \cdot \frac{\alpha|B|}{|A|\gamma} \cdot w,\end{aligned} \tag{2}$$

where the inequality follows since $\left\lceil \frac{\gamma}{|B|} \right\rceil^{-1} \leq \frac{|B|}{\gamma}$.

Suppose now that $\gamma < m$. Note that in this case we can write $\beta = c\gamma + d$, where $c \geq 1$ and $d \in \{0,\ldots,\gamma-1\}$. Let j^* be as defined above, and let us, similarly, define j° as the machine in B achieving the highest load in the optimal allocation for $\hat{\mathbf{z}}$, i.e., $j^\circ = \arg\max_{j\in B}(x_j \cdot OPT_j(\hat{\mathbf{z}}))$. Finally, we denote with $j^\dagger$ a machine in B that achieves less than $c+1$ times the number of jobs that it receives in the optimal allocation for $\mathbf{x}$, i.e., $OPT_{j^\dagger}(\hat{\mathbf{z}}) \leq c \cdot OPT_{j^\dagger}(\mathbf{x})$. Clearly, such a machine must necessarily exist. Note also that, by our tie-breaking rule about optimal assignments, we have that $OPT_{j^\dagger}(\mathbf{x}) \geq 1$.

Observe that

$$\begin{aligned} x_{j^\circ} \cdot OPT_{j^\circ}(\hat{\mathbf{z}}) &\leq x_{j^\dagger} \cdot (OPT_{j^\dagger}(\hat{\mathbf{z}}) + 1) && \text{(by optimality)} \\ &\leq x_{j^\dagger} \cdot \left[\left(c \cdot OPT_{j^\dagger}(\mathbf{x}) + OPT_{j^\dagger}(\mathbf{x}) - 1\right) + 1\right] && \text{(by def. of } j^\dagger) \\ &\leq 2 \cdot c \cdot x_{j^\dagger} \cdot OPT_{j^\dagger}(\mathbf{x}) \\ &\leq 2 \cdot c \cdot x_{j^*} \cdot OPT_{j^*}(\mathbf{x}). && \text{(by def. of } j^*) \end{aligned}$$

Moreover, $\beta \geq c \cdot \gamma$, from which we have that $\frac{|B|}{\beta} \leq \frac{|B|}{c\gamma}$. Hence we have that $\hat{w} \leq 2 \cdot \frac{\alpha|B|}{|A|\gamma} \cdot w$.

Then, for both cases, it is sufficient to prove that $\frac{\alpha|B|}{|A|\gamma} \leq 1$, or alternatively, that $\gamma \geq \frac{|B|}{|A|}\alpha$. However this immediately follows from the fact that the machines in B

are the fastest ones, and our tie-breaking rule among optimal allocations chooses the allocation that maximizes the number of jobs on these machines.

$$\sum_{i \in B} OPT_i(\hat{\mathbf{z}}) < \frac{k+1}{\lceil \tau \rceil + k} \beta.$$

□

According to Lemma 4, and Lemma 5, we then have that

$$\begin{aligned} \frac{MS(\hat{\mathbf{a}}, \hat{\mathbf{y}})}{MS(\mathbf{a}, \mathbf{y})} &\leq \frac{m + 2k + \lceil \tau \rceil - 1}{m} \cdot \frac{\hat{\nu}\hat{\theta}}{\hat{\nu} + \hat{\theta}} \cdot \frac{\nu + \theta}{\nu\theta} \\ &= \frac{m + 2k + \lceil \tau \rceil - 1}{m} \cdot \frac{\hat{w}}{w} \cdot \frac{w(n - k - \lceil \tau \rceil) + t(k + \lceil \tau \rceil)}{\hat{w}(\lceil \tau \rceil - 1) + t(k+1)}. \end{aligned} \tag{3}$$

Observe that

$$\begin{aligned} \frac{\hat{w}}{w} \cdot \frac{w(n - k - \lceil \tau \rceil) + t(k + \lceil \tau \rceil)}{\hat{w}(\lceil \tau \rceil - 1) + t(k+1)} &= \frac{\hat{w}}{w} \cdot \frac{w(\tau(k+\tau) - (k + \lceil \tau \rceil)) + t(k + \lceil \tau \rceil)}{\hat{w}(\lceil \tau \rceil - 1) + t(k+1)} \\ &\leq (k + \lceil \tau \rceil) \cdot \frac{\hat{w}}{w} \cdot \frac{w(\lceil \tau \rceil - 1) + t}{\hat{w}(\lceil \tau \rceil - 1) + t(k+1)}. \end{aligned} \tag{4}$$

Moreover, the function $\frac{w(\lceil \tau \rceil - 1) + t}{\hat{w}(\lceil \tau \rceil - 1) + t(k+1)}$ turns out to be non-increasing in t, given that, by Lemma 6, $\hat{w} \leq 2 \cdot w \leq (k+1) \cdot w$, where the last inequality follows since $k \geq 1$. Hence, since $t \geq w$,

$$\begin{aligned} \frac{\hat{w}}{w} \cdot \frac{w(\lceil \tau \rceil - 1) + t}{\hat{w}(\lceil \tau \rceil - 1) + t(k+1)} &\leq \frac{\hat{w}}{w} \cdot \frac{w(\lceil \tau \rceil - 1) + w}{\hat{w}(\lceil \tau \rceil - 1) + w(k+1)} \\ &= \frac{\lceil \tau \rceil \hat{w}}{\hat{w}(\lceil \tau \rceil - 1) + w(k+1)} \\ &\leq \frac{\lceil \tau \rceil \hat{w}}{\hat{w}(\lceil \tau \rceil - 1) + \hat{w}(k+1)/2} \\ &= \frac{\lceil \tau \rceil}{\lceil \tau \rceil + k/2 - 1/2} \leq \frac{2\lceil \tau \rceil}{\lceil \tau \rceil + k}, \end{aligned} \tag{5}$$

where the last inequality follows since $\lceil \tau \rceil + k/2 - 1/2 \geq \frac{1}{2}(\lceil \tau \rceil + k)$. By merging (3), (4), and (5), we achieve the desired bound. □

The next corollary follows by simple algebraic manipulations from Proposition 2.

Corollary 1 *Mechanism $\mathcal{M}_k$, for $1 \leq k \leq n-2$, is $2(\lceil \tau \rceil + 1)$-approximate for $m > \lceil \tau \rceil (2k + \lceil \tau \rceil)$.*

4 The Case of Facility Location

Let us now consider the facility location problem. Here, the type t_i of each agent consists of her position on the real line. The algorithm f must choose a position $z \in \mathbb{R}$ for the facility. The cost that agent i pays for having the facility in position z is

$t_i(z) = d(t_i, z) = |t_i - z|$. So, $t_i(z)$ denotes the distance between t_i and the location of the facility.

We focus on mechanisms implementing a social choice function f^* that optimizes the *social cost*, i.e.,

$$f^*(\mathbf{b}) \in \arg\min_{z\in\mathbb{R}} \texttt{cost}(z, \underline{\ }), \quad \texttt{cost}(z, \underline{\ }) = \sum_{i=1}^{n} b_i(z).$$

As above, we say that the mechanism $\mathcal{M}$ is α-approximate if the solution z returned by the mechanism has a social cost that is at most a factor α away from the cost of the solution returned by f^* for any input.

Let $\tau = \tau_k(n) = \max\left\{1, \frac{n-k-1}{k+1}\right\}$. Observe that $\tau_0(n) = n-1$ and $\tau_{n-1}(n) = 1$. More specifically, $\tau = 1$ for every $k \geq \frac{n}{2} - 1$ and $\tau = O(1)$ if and only if $k = \Omega(n)$.

We now show that there is no k-OSP mechanism for the facility location problem with an approximation ratio better than τ. The proof extends a similar argument given for OSP mechanisms in [7]. Let us first prove the following lemma.

Lemma 7 *Fix $k \in \left\{0, \ldots, \left\lceil \frac{n}{2} \right\rceil - 2\right\}$. For every α, β, with $\alpha < \beta$, no ρ-approximate mechanism $\mathcal{M}$ for the facility location problem, with $\rho < \tau_k(n)$, sets $f(\boldsymbol{b}) \leq \alpha$, if $b_j = \alpha$ for $k+1$ agents j and $b_j = \beta$ for every remaining agent, where f is the social choice function used by $\mathcal{M}$.*

Proof The optimal facility location for the given setting consists in placing the facility in position β. The total cost in this case is $(k+1)(\beta - \alpha)$.

If $f(\mathbf{b}) \leq \alpha$, then the total cost is larger than $(n-k-1)(\beta-\alpha)$, and thus the approximation ratio of the mechanism would be $\frac{n-k-1}{k+1} = \tau_k(n) > \rho$. □

With the same argument one can prove also the following lemma.

Lemma 8 *For every α, β, with $\alpha < \beta$, no ρ-approximate mechanism $\mathcal{M}$ for the facility location problem, with $\rho < n-1$, sets $f(\boldsymbol{b}) \geq \beta$, if $b_j = \beta$ for one single agent and $b_j = \alpha$ for every remaining agent, where f is the social choice function used by $\mathcal{M}$.*

Theorem 2 *For every $\varepsilon > 0$, there is no $(\tau - \varepsilon)$-approximate mechanism for the facility location problem that is k-OSP.*

Proof For $k \geq \left\lceil \frac{n}{2} \right\rceil - 1$, we have $\tau = 1$, and thus the claim is obvious.

Suppose then $k \leq \left\lceil \frac{n}{2} \right\rceil - 2$ and let $\mathcal{M} = (f, \mathcal{T}, p)$ be a $(\tau - \varepsilon)$-approximate mechanism for the facility location problem that is k-OSP. Let us consider the domain of every agent to be $D = \{a, a+\delta, \ldots, b-\delta, b\}$, where $\delta \leq \frac{(k+1)\varepsilon}{n-2k-2} \cdot \frac{b-a}{2}$. The proof essentially works by proving that there are instances in which any k-OSP mechanism must place the facility either in x or in $x+\delta$, for some $x \in D$. Setting δ to be a small value, as described above, implies that the choice among x and $x+\delta$ essentially does not influence the approximation ratio of the mechanism. Note also that it is w.l.o.g. to

consider only this domain, since any k-OSP mechanism returning a ρ-approximation in a larger domain must necessarily have the same properties when applied on a smaller domain.

Then, let i be the first divergent agent of $\mathcal{M}$ (such a divergent agent must exist, otherwise the mechanism must return the same solution when all agents are in a, and when they are all in b, which is impossible without violating the approximation guarantee of the mechanism) and let u_i be the node of $\mathcal{T}$ at which it diverges. Note that, by definition of divergent agent, there must be two types t_i, t_i' of agent i such that $t_i' = t_i + \delta$ and i takes an action in $\mathcal{M}$ when her type is t_i that is different from the action taken when her type is t_i'. We denote as c and d the smallest t_i and the largest t_i', respectively, for which this occurs, i.e., c is the smallest type in D' such that i diverges on c and $c + \delta$, and d is the largest type in D' such that i diverges on d and $d - \delta$.

Note that either $c < \frac{b+a}{2}$ or $d > \frac{b+a}{2}$. Indeed, if $c \geq \frac{b+a}{2}$, then $d \geq c + \delta > \frac{b+a}{2}$. In the rest of the proof we will assume that $c < \frac{a+b}{2}$. The proof for the case that $d > \frac{a+b}{2}$ simply requires to replace c with d, $c + \delta$ with $d - \delta$, and b with a, and invert the direction of the inequalities in the next claims.

The proof fixes an agent i and uses two profiles $\mathbf{x}$ and $\mathbf{y}$, that are defined as follows:

- $x_i = c + \delta$, and $x_j = c$ for every $j \neq i$;
- $y_i = c$, $y_j = c$ for $j \in Ł(u_i)$, and $y_k = b$ for every remaining agent j.

We begin by using k-OSP to relate payments and outcomes of the mechanism $\mathcal{M}$ on input $\mathbf{x}$ and $\mathbf{y}$. Specifically, we note that if the real location of i is $t_i = x_i = c+\delta$ then $u_i(\mathbf{x}) = p_i(\mathbf{x}) - d(c + \delta, f(\mathbf{x}))$, and $u_i(\mathbf{y}) = p_i(\mathbf{y}) - d(c + \delta, f(\mathbf{y}))$.

Since i diverges at u_i on c and $c + \delta$, agents in $Ł(u_i)$ play the same actions in x and y, and $\mathcal{M}$ is k-OSP, we have that $u_i(\mathbf{x}) \geq u_i(\mathbf{y})$. Hence, it follows that

$$p_i(\mathbf{x}) \geq p_i(\mathbf{y}) - d(c + \delta, f(\mathbf{y})) + d(c + \delta, f(\mathbf{x})). \tag{6}$$

If the real location of i is $t_i' = y_i = c$, then $u_i(\mathbf{x}) = p_i(\mathbf{x}) - d(c, f(\mathbf{x}))$, and $u_i(\mathbf{y}) = p_i(\mathbf{y}) - d(c, f(\mathbf{y}))$. As above, since i diverges on c and $c + \delta$, agents in $Ł(u_i)$ play the same actions in x and y, and $\mathcal{M}$ is k-OSP, we have that $u_i(\mathbf{y}) \geq u_i(\mathbf{x})$. Hence, it follows that

$$p_i(\mathbf{x}) \leq p_i(\mathbf{y}) - d(c, f(\mathbf{y})) + d(c, f(\mathbf{x})). \tag{7}$$

Therefore, in order to satisfy both (6) and (7), we need that

$$d(c + \delta, f(\mathbf{y})) - d(c, f(\mathbf{y})) \geq d(c + \delta, f(\mathbf{x})) - d(c, f(\mathbf{x})). \tag{8}$$

Using (8) above, we first show that $f(\mathbf{x})$ must be at least c and then that $f(\mathbf{y}) \leq c+\delta$. Finally, we prove how this last fact contradicts the desired approximation ratio.

Let us first show that $f(\mathbf{x}) \geq c$. Suppose instead that $f(\mathbf{x}) < c$. Since $f(\mathbf{x}) < c$, then the r.h.s. of (8) is δ. As for the l.h.s., we distinguish two cases. If $f(\mathbf{y}) \leq c + \delta$, then, since $f(\mathbf{y}) > c$ according to Lemma 7, we have $(c + \delta - f(\mathbf{y})) - (f(\mathbf{y}) - c) = \delta - 2(f(\mathbf{y}) - c) < \delta$. If $f(\mathbf{y}) > c + \delta$, we have $(f(\mathbf{y}) - (c + \delta)) - (f(\mathbf{y}) - c) = -\delta$. Hence, in both cases we reach a contradiction.

We now show that $f(\mathbf{y}) \leq c + \delta$. Assume by contradiction that $f(\mathbf{y}) > c + \delta$. Since $f(\mathbf{x}) \geq c$, and $f(\mathbf{x}) < c + \delta$ by Lemma 8, we can rewrite (8) as

$$(f(\mathbf{y})-(c+\delta))-(f(\mathbf{y})-c) \geq ((c+\delta)-f(\mathbf{x}))-(f(\mathbf{x})-c) \Rightarrow -\delta \geq \delta-2(f(\mathbf{x})-c).$$

However, this is impossible since $f(\mathbf{x}) < c + \delta$.

Finally, we prove that, given that $f(\mathbf{y}) \leq c+\delta$, then the mechanism is not $(\tau-\varepsilon)$-approximate. Indeed, since by Lemma 7 $f(\mathbf{y}) > c$, the total cost of mechanism $\mathcal{M}$ on input $\mathbf{y}$ is

$$\begin{aligned}
&(k+1)(f(\mathbf{y})-c)+(n-k-1)(b-f(\mathbf{y})) \\
&\qquad = (n-k-1)b-(k+1)c-(n-2k-2)f(\mathbf{y}) \\
&\qquad = (n-k-1)(b-c)-(n-2k-2)(f(\mathbf{y})-c) \\
&\qquad \geq (n-k-1)(b-c)-(n-2k-2)\delta \\
&\qquad \geq (n-k-1)(b-c)-(n-2k-2)\frac{(k+1)\varepsilon}{n-2k-2}\cdot\frac{b-a}{2} \\
&\qquad > (n-k-1-(k+1)\varepsilon)(b-c),
\end{aligned}$$

where we used that $b-c > b-\frac{b+a}{2} = \frac{b-a}{2}$. However, this is absurd, since $\mathcal{M}$ is $(\tau-\varepsilon)$-approximate, where $\tau = \frac{n-k-1}{k+1}$ and the optimal mechanism on input $\mathbf{y}$ places the facility in b and has total cost $(k+1)(b-c)$. □

5 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 [4] to allow for more fine grained notions of rationality. For machine scheduling and facility location, 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. However, we highlight that no approximation result is known even for OSP mechanisms.

Funding Open access funding provided by Università degli Studi di Salerno within the CRUI-CARE Agreement.

References

1. Ferraioli, D., Ventre, C.: Obvious strategyproofness, bounded rationality and approximation: The case of machine scheduling. In: SAGT 2019 (2019)
2. Ferraioli, D., Ventre, C.: Obvious strategyproofness, bounded rationality and approximation. In: AAMAS 2019 (2019)
3. Ausubel, L.M.: An efficient ascending-bid auction for multiple objects. Amer. Econ. Rev. **94**(5), 1452–1475 (2004)
4. Li, S.: Obviously strategy-proof mechanisms. Amer. Econ. Rev. **107**(11), 3257–87 (2017)
5. Ashlagi, I., Gonczarowski, Y.A.: Stable matching mechanisms are not obviously strategy-proof. J. Econ. Theory **177**, 405–425 (2018)
6. Pycia, M., Troyan, P.: Obvious dominance and random priority. In: EC 2019 (2019)
7. Ferraioli, D., Ventre, C.: Obvious strategyproofness needs monitoring for good approximations. In: AAAI 2017, pp. 516–522 (2017)
8. Mackenzie, A.: A revelation principle for obviously strategy-proof implementation. Research Memorandum 014, (GSBE) (2017)
9. Zhang, L., Levin, D.: Bounded rationality and robust mechanism design: An axiomatic approach. Amer. Econ. Rev. **107**(5), 235–39 (2017)
10. Bade, S., Gonczarowski, Y.A.: Gibbard-Satterthwaite success stories and obvious strategyproofness. In: EC 2017, p. 565 (2017)
11. Ferraioli, D., Ventre, C.: Probabilistic verification for obviously strategyproof mechanisms. In: IJCAI 2018 (2018)
12. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: Obviously strategyproof mechanisms for machine scheduling. In: ESA 2019 (2019)
13. Kyropoulou, M., Ventre, C.: Obviously strategyproof mechanisms without money for scheduling. In: AAMAS 2019 (2019)
14. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: Automated optimal osp mechanisms for set systems: The case of small domains. In: WINE 2019 (2019)
15. Ferraioli, D., Penna, P., Ventre, C.: Two-way greedy: Algorithms for imperfect rationality. In: WINE 2021 (2021)
16. De Groot, A.: Thought and choice in chess. Mouton (1978)
17. Shannon, C.: Programming a computer for playing chess. Philos. Mag. **41**(314), 256–275 (1950)
18. Pearl, J.: Heuristics: Intelligent search strategies for computer problem solving. Addison-Wesley (1984)
19. Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: FOCS 2001, pp. 482–491 (2001)
20. Moulin, H.: On strategy-proofness and single-peakedness. Public Choice **35**, 437–455 (1980)
21. Sandholm, T., Gilpin, A.: Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. In: AMEC 2003, pp. 73–91 (2003)
22. Hartline, J., Roughgarden, T.: Simple versus optimal mechanisms. In: EC 2009, pp. 225–234 (2009)
23. Chawla, S., Hartline, J., Malec, D., Sivan, B.: Multi-parameter mechanism design and sequential posted pricing. In: STOC 2010, pp. 311–320 (2010)
24. 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)
25. Adamczyk, M., Borodin, A., Ferraioli, D., de Keijzer, B., Leonardi, S.: Sequential posted price mechanisms with correlated valuations. In: WINE 2015, pp. 1–15 (2015)
26. Feldman, M., Fiat, A., Roytman, A.: Makespan minimization via posted prices. In: EC 2017, pp. 405–422 (2017)
27. 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)
28. 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)
29. Brânzei, S., Procaccia, A.D.: Verifiably truthful mechanisms. In: ITCS 2015, pp. 297–306 (2015)
30. Glazer, J., Rubinstein, A.: An extensive game as a guide for solving a normal game. J. Econ. Theory **70**, 32–42 (1996)