An Algorithmic Theory of Simplicity in Mechanism Design

Diodato Ferraioli[1] [0000−0002−7962−5200] and Carmine Ventre[2] [0000−0003−1464−1215]

[1] Università degli Studi di Salerno, Fisciano (SA) 84016, Italy
dferraioli@unisa.it
[2] King's College London, Strand London WC2R 2LS, United Kingdom
carmine.ventre@kcl.ac.uk

Abstract. A growing body of work in economics and computation focuses on the trade-off between implementability and simplicity in mechanism design. The goal is to develop a theory that not only allows to design an incentive structure easy to grasp for imperfectly rational agents, but also understand the ensuing limitations on the class of mechanisms that enforce it. In this context, OSP mechanisms have assumed a prominent role since they provably account for the absence of contingent reasoning skills, a specific cognitive limitation. For single-dimensional agents, it is known that OSP mechanisms need to use certain greedy algorithms. In this work, we introduce a notion that interpolates between OSP and SOSP, a more stringent notion where agents only plan a subset of their own future moves. We provide an algorithmic characterization of this novel class of mechanisms for single-dimensional domains and binary allocation problems, that precisely measures the interplay between simplicity and implementability. We further show how mechanisms based on reverse greedy algorithms (a.k.a., deferred acceptance auctions) are algorithmically more robust to imperfectly rationality than those adopting greedy algorithms.

1 Introduction

A set of agents need to determine an outcome that will affect each of them. In these cases, a mechanism is designed to interact with the agents and compute an outcome based on the decisions made during the interaction. In its more general form, the mechanism can be modelled as an extensive-form game. To guarantee properties on the quality of the solution computed (e.g., optimality for an objective function of interest) mechanisms are required to provide incentives for the agents to behave in a predictable and desirable way. For example, in dominant-strategy incentive-compatible mechanisms, it is pointwise optimal for the agents to behave correctly and truthfully report their private information to the mechanism. This property can however be too weak for certain agents, as observed experimentally [7, 20]. In particular, different extensive-form implementations of the mechanism can lead to different degrees of strategic confusion. Consider, for example, software agents that take decisions during the execution of the mechanism. These agents could take actions that are irrational from the economic point of view when they have been "badly" programmed, either because the programmer misunderstood the incentive structure in

place or due to computational barriers preventing from comparing payoffs for each strategy adopted by the other agents [26]. A less complex decision process would be guaranteed by associating one outcome/payoff to each possible action taken by the agent, since she would simply need to rank her own actions independently of the behavior of the other agents. This is the idea behind obviously strategyproof (OSP) mechanisms [23]. OSP is a stronger notion of incentive-compatibility that takes a conservative view and requires honesty to be an obviously dominant strategy: the worst payoff that can be achieved with it is not worse than the best possible payoff obtained with a different strategy, where worst and best are chosen over the possible strategies of the remaining agents.

OSP is shown to capture the incentives of a specific form of imperfect rationality — absence of contingent reasoning skills. From the computational point of view, OSP is known to be intimately linked with greedy algorithms for single-dimensional agents: OSP is equivalent to certain well-defined combinations of greedy algorithms that are suitably monotone in the agent's private information [13, 18]. This algorithmic lens allows to focus on the quality of the solutions output by OSP mechanisms, measured in terms of the approximation guarantee to a given objective function, and conclude that these monotone greedy algorithms can perform well for some binary allocation problems [13] but, in general, less well for richer solution spaces [18].

Does OSP encapsulate strategic simplicity in mechanism design? Let us consider agents with limited foresight, who can only devise a plan for their next k moves (their *planning horizon*) but cannot anticipate what they will do after that. It may not be simple enough for agents to go beyond that horizon in their reasoning; in chess, for example, if white can always win, any winning strategy is obviously dominant but requires to look at many moves in the future meaning that the strategic choices in chess are far from obvious [28]. Similarly, software agents may need more data to be "retrained" and get a more granular picture of the scenarios that go beyond their k-th future move and plan accordingly. What is simplicity in mechanism design for these agents? OSP would result simple for agents with infinite planning horizons (e.g., agents who need not retrain and have all the data available from the beginning). A solution concept called Strong OSP (SOSP) has been recently introduced in the literature to define simplicity for agents who cannot forecast any of their own future moves [28] (e.g., those who need to retrain after each move). Can we interpolate between these two extremes? Is, in case, the class of implementable mechanisms larger the longer the agents' planning horizons?

Our Contribution. We introduce the notion of k-step OSP mechanisms to capture the incentive compatibility of agents who decide their next move by only planning for the subsequent k steps and leave the remaining moves undecided. Thus, OSP corresponds to the case in which $k = \infty$ since, when determining their next move, agents reason about all their subsequent actions in the extensive-form mechanism. SOSP, on the contrary, corresponds to the case in which $k = 0$ in that no extra future action is planned for in addition to the one under consideration. We fully characterize k-step OSP mechanisms as follows:

Main Theorem 1 (informal). *A mechanism is k-step OSP for single-parameter agents and binary allocation problems if and only if each agent receives at most $k+2$ "OSP queries", the $(k+2)$-th (if any) being "payoff-neutral up to one type".*

In the informal statement above, "OSP query" means that the extensive-form mechanism must satisfy the ensuing OSP constraints, whereas "payoff-neutral up to one type query" means that the payoff for the queried agent i is essentially determined for all her moves but potentially one; this move corresponds to a single type of i for which her outcome can still be undecided at that point. Whilst it is expected that the queries must be OSP, it is surprising that there is clean connection between the number of possible queries that each agent can be asked and her planning horizon, with up to $k+1$ "unrestricted" queries to each agent i and one closing query that not only can affect the outcome that the other players will receive but also i's (albeit very limitedly)[3]. Thus, our notion fully captures the trade-off between simplicity and implementability. The more sophisticated the decision making of the agents is, the more mechanisms are incentive compatible. In particular, our notion gives rise to a fully nested class of mechanisms since every k-step OSP mechanism is also k'-step OSP for $k' > k$.

To prove this characterization, we develop the theoretical foundations to study k-step OSP mechanisms. We prove that it is w.l.o.g. to focus on a certain family of implementations and find a structural property of k-OSP constraints that limits the class of queries that the mechanism can make. We subsequently give a version of the Taxation Principle for k-OSP mechanisms by proving that from a certain point of the extensive-form game, outcomes (and payments) need to be constant for the agent being queried. Few additional structural properties are then proved to bound the number of queries to $k+2$ and determine the power of the last query.

This result complements the results by Pycia and Troyan [28] for binary allocation problems from two perspectives. Firstly, our characterization says that each agent has two payoff relevant queries in an SOSP mechanism. Under an assumption of rich domains, Pycia and Troyan [28] prove instead that agents only take one payoff-relevant move in SOSP mechanisms. Thus, our result highlights that for general single-dimensional type domains, there can be a second query which affects the outcome of the queried agent in a very specific and limited way. Secondly, Pycia and Troyan [28] consider a notion of k-step dominance, which is very related to ours, in that agents think about k future self moves when taking a decision. It turns out that their definition is technically different, with consequences on the boundaries between simplicity and implementability. In [28], although agents take into account their subsequent k moves they do not commit to them but rather follow the principle of "cross[ing] that bridge when you come to it" (i.e., make choices as they arise) [30]. It is thus assumed that the agents optimistically complete the "honest" path of play beyond their planning horizon when taking a decision. To some extent, our notion formalizes the "look before you leap" (i.e., create a complete contingent plan for the possible k future decisions one may have) model of decision making

[3] This connection is neither an "innate" nor a straightforward property of k-step OSP mechanisms, since for larger outcome spaces there may be k-step OSP mechanisms with more than $k+2$ queries.

[30]. Agents take a pessimistic perspective by considering the worst possible way in which they can complete their "honest" path of play beyond their planning horizon. Consequently, our approach is more restrictive but more robust to decisions made outside the agents' planning horizons. (Please see Example 1 for an illustration of the differences between the two notions.) Importantly, within their modeling of decision making, Pycia and Troyan [28] prove that one future move is enough for 1-step simple mechanisms, for which honesty is one-step dominant, to collapse to OSP mechanisms. In fact, agents can at each step reconsider whether the plan that was 1-step dominant at the previous decision point is still the right way to go; intuitively, this freedom allows the agents to reconstruct an obviously dominant strategy step by step. Our notion instead draws a more fine grained boundary between the mechanisms that can be implemented and the behavioral model of the agents interacting with them.

Example 1 (Ascending price auctions). Consider an ascending auction for a single good, where at each non-terminal information set, an agent is called to play and has two actions, Stay In or Drop Out. The payoff of an agent i is equal to the agent's valuation v_i minus her payment if the agent is allocated the good and 0 otherwise. The price for the good weakly increases along each path of play. The auction ends when there is only one agent who has not dropped out; she wins the good and pays the price associated to the last time she moved. We remark that the ascending auction is OSP because the strategy of staying in as long as the current price is below the agent's valuation is obviously dominant, as shown in [23].

A one-step dominant strategy at information set I^* is the following. If $p(I^*)$, the price at I^*, is not higher than v_i, then Stay In and Drop Out at the subsequent information set I along a path of play. If the price at I^* is higher than v_i, then Drop Out at I^* and the subsequent information set I. This is one-step dominant because in both cases, the minimum from following the strategy is 0 (dropping out eventually) which is not worse than other strategies at I^* which can only guarantee a payoff of 0. Importantly, in her decision process, agent i is ignoring future paths of play wherein the agent will not drop out. However, player i must not necessarily drop out at I but can still stay in if the price at I is not higher than v_i. Agent i can thus eventually have inconsistent plans (cf. Remark 1 in [28]). Moreover, there are paths where i could, beyond her planning horizon, stay in even for prices higher than v_i.

A 1-step obviously dominant strategy instead mimics the aforementioned obviously dominant strategy but limits it to a planning horizon of two moves. So the agent would look at both $p(I^*)$ and $p(I)$, prices at I^* and I, and Stay In (Drop Out, resp.) at $I' \in \{I^*, I\}$ if $p(I') \leq v_i$ ($p(I') > v_i$, resp.). Incidentally, and differently from above, our notion leads to consistent strategies since the decision of agent i at I will follow the same recommendation dictated by the strategy at I^*. This strategy is 1-step obviously dominant only when the minimum payoff that can be reached by completing it in any possible way would be better than a deviation. So, for example, if the type domain of agent i were \$1, \$2, \$3, \$4 and \$5, and her type were \$4 then, by following the strategy, agent i would have stayed in for prices \$2 and \$3 (for her first two moves). To evaluate the 1-step obvious dominance of the strategy, agent i would then take the worst possible path of play that her future self could take, e.g., the decision to stay in

for a price of \$5 leading to a payoff of -1. The best possible payoff by deviating is 0 (e.g., dropping out).

To further explore the boundary between implementabilty and simplicity, we study the extent to which the approximation guarantee of OSP deteriorates when future self moves are not fully accounted for in decision making. We build on our first theorem to characterize what social choice functions can be implemented by k-step OSP mechanisms. Towards this end, we define a version of cycle monotonicity for k-step OSP; as in the cases of SP and OSP (amongst others) cycle monotonicity is a useful tool in that it allows to separate the allocation function (whose approximation guarantee we want to bound) from the payment function. The construction of this useful toolkit requires some further work to conveniently rewrite k-step OSP constraints. We are then able to algorithmically characterize k-step OSP mechanisms, by relating the cycles of the OSP graph [12] with those of the k-step OSP graph we define.

Main Theorem 2 (informal). *There is a k-step OSP mechanism implementing binary social choice function f if and only if f is two-way greedy with k-limitable priority functions.*

The algorithmic nature of OSP for binary allocation problems can be explained in terms of two-way greedy algorithms [13]. These are algorithms that build the eventual solution by either greedily including agents with low cost (high valuation, resp.) – termed forward greedy algorithm – or by greedily excluding agents that have a high cost (low valuation, resp.) – termed reverse greedy algorithm – for minimization (maximization, respectively) problems.[4] These algorithms adopt adaptive priority functions (that is, priority functions that depend on the past choices made by the algorithm) to define the greedy order in which agents have to be processed. The restriction from OSP to k-step OSP limits the class of priority functions that can adopted – it must be the case that each agent i is at the top of the priority list for no more than $k+2$ times excluding the instances in which i is consecutively at the top. These are k-limitable priority functions. Loosely speaking, this means that the greedy algorithm must be ready to decide whether an agent is included or excluded from the solution in no more than $k+2$ steps, thus clearly demarcating the algorithmic limitations that k-step OSP imposes.

We conclude this work by exploring the power of this class of algorithms. We focus here on p-systems, a class of problems for which it is known that greedy algorithms can compute a solution with approximation at most p. To what extent can this result be replicated for k-step OSP mechanisms? The size of the type domain of the agents and the format of the greedy algorithm play a crucial role. Let d denote half the size of the agents' type domain (for simplicity of exposition assume that d is an integer); we obtain the following tight result.

[4] In general, different agents can face different directions; in specific circumstances, it is possible to move from one direction to the other even for single agents, cf. Section 5.

Main Theorem 3 (informal). *There is a p-approximate k-limitable reverse greedy algorithm for p-systems whenever $k \geq d-2$. There is an instance of a 1-system (i.e., a weighted matroid) for which no k-limitable two-way (forward, respectively) greedy algorithm has bounded approximation for $k < d-2$ ($k < 2d-3$, respectively).*

We can then conclude that, in the worst case, there is no gradual degradation of the performances of k-step OSP mechanisms; as the value of k decreases, there is a stark dichotomy. Essentially, the two-way greedy algorithm needs to traverse the domain of each agent to compute good solutions when the types in the domain are sufficiently far apart, meaning that k must be large enough. It is conceivable to imagine that were it possible to cluster the domain of the agents around k centres without losing too much granularity, then better results would be possible. Interestingly, reverse greedy needs half the number of priority functions of forward greedy. It was already known that forward greedy is less robust than reverse greedy to economic desiderata (such as, individual rationality) of OSP mechanisms, due to the characterization of OSP payments in [17]. Our result further shows that reverse greedy is not only economically but also algorithmically more robust than forward greedy to imperfect rationality.

Related Work. OSP mechanisms have been extensively studied in various contexts, including stable matchings and single-peaked domains [6, 32, 5]; settings without monetary transfers [27]; binary allocation and single-dimensional problems [13, 18]; and, combinatorial auctions and machine scheduling [21, 10]. Extensions and variants of OSP include monitoring and verification in OSP mechanisms [15, 14]; k-OSP, interpolating between OSP and classical strategyproofness [16]; and, Non-Obviously Manipulable (NOM) mechanisms [33, 3]. These studies have provided characterizations, impossibility results, and approximation bounds for various OSP mechanism design problems.

2 Preliminaries

Let N be a set of *n selfish agents* and let $\mathscr{S}$ be a set of feasible *outcomes*. Each agent i has a *type* $t_i \in D_i$, where D_i is the *domain* of i. The type t_i is assumed to be *private knowledge* of agent i. We let $t_i(X) \in \mathbb{R}$ denote the *cost* of agent i with type t_i for the outcome $X \in \mathscr{S}$. When costs are negative, it means that the agent has a profit from the solution, called *valuation*.

We would like to run a *mechanism* in order to select an outcome and assign opportune payments. I.e., the mechanism implements a pair (f, p), where f (termed *social choice function* or, simply, algorithm) maps the actions taken by the agents to a feasible solution in $\mathscr{S}$, and p maps the actions taken by the agents to *payments*. Note that payments need not be positive. Each selfish agent i is equipped with a *quasi-linear utility function* $u_i \colon D_i \times \mathscr{S} \to \mathbb{R}$: for $t_i \in D_i$ and for an outcome $X \in \mathscr{S}$ returned by a mechanism $\mathscr{M}$, $u_i(t_i, X)$ is the utility that agent i has for the implementation of outcome X when her type is t_i, i.e., $u_i(t_i, X) = p_i - t_i(X)$. In this work

we will focus on *single-parameter* settings, that is, the case in which the private information of each bidder i is a single real number t_i and $t_i(X)$ can be expressed as $t_i w_i(X)$ for some publicly known function w_i.

In order to implement (f, p), we design a game Γ for the agents to play. Specifically, Γ is an imperfect-information, extensive-form game with perfect recall, defined in the standard way: $\mathscr{H}$ is a finite collection of partially ordered *histories* (i.e., sequences of moves). At every non-terminal history $h \in \mathscr{H}$, one agent $i \in N$ is called to play and has a finite set of *actions* $A(h)$ from which to choose. At some history h it may be also possible that the mechanism M plays by taking a random choice: specifically, we denote with $\omega(h)$ the realization of the mechanism's random choice at history h, and with $\omega = (\omega(h))_{h \in \mathscr{H}:\, M \text{ plays at } h}$ the mechanism's random choices along the entire game. Each terminal history is associated with an outcome $X \in \mathscr{S}$, and agents receive utility $u_i(t_i, X)$. We let $h' \subseteq h$ denote that h' is a subhistory of h (equivalently, h is a continuation history of h'), and write $h' \subset h$ when $h' \subseteq h$ but $h \neq h'$. When useful, we sometimes write $h = (h', a)$ to denote the history h that is reached by starting at history h' and following the action $a \in A(h)$.

An *information set* I of agent i is a set of histories such that for any $h, h' \in I$ and any subhistories $\tilde{h} \subseteq h$ and $\tilde{h}' \subseteq h'$ at which i moves, at least one of the following conditions holds: (i) there is a history $\tilde{h}^* \subseteq \tilde{h}$ such that $\tilde{h}^*$ and $\tilde{h}'$ are in the same information set, $A(\tilde{h}^*) = A(\tilde{h}')$, and i makes the same move at $\tilde{h}^*$ and $\tilde{h}'$; (ii) there is a history $\tilde{h}^* \subseteq \tilde{h}'$ such that $\tilde{h}^*$ and $\tilde{h}$ are in the same information set, $A(\tilde{h}^*) = A(\tilde{h})$, and i makes the same move at $\tilde{h}^*$ and $\tilde{h}$. We denote by $I(h)$ the information set containing history h. Roughly speaking, an information set collects all the histories that an agent is unable to distinguish.

A *strategy* for a player i in game Γ is a function S_i that specifies an action at each one of her information sets. When we want to refer to the strategies of different types t_i of agent i, we write $S_i(t_i)$ for the strategy followed by agent i of type t_i; in particular, $S_i(t_i)(I)$ denotes the action chosen by agent i with type t_i at information set I. We use $\mathbf{S}(\mathbf{t}) = (S_i(t_i))_{i \in N}$ to denote the strategy profile for all of the agents when the type profile is $\mathbf{t} = (t_i)_{i \in N}$. An *extensive-form mechanism* is an extensive-form game together with a profile of strategies $\mathbf{S}$.

Given an agent i, an information set I is an *earliest point of departure* between strategy S_i and S_i' in game Γ if I is on the path of play under both S_i and S_i' and both strategies choose the same action at all earlier information sets, but choose a different action at I. I.e., I is the earliest information set at which these two strategies diverge. Note that for two strategies, there will in general be multiple earliest points of departure.

For an agent i with preference t_i, strategy S_i *k-step obviously dominates* strategy S_i' in game Γ if, starting at any earliest point of departure I between S_i and S_i', the outcome that maximize the utility of i among the ones reachable following S_i' at I is weakly worse than the outcome that achieves the minimum utility among the ones reachable by following S_i at I, where worst (best, resp.) cases are determined by considering any future play by other agents (including random choices of the mechanism) and any future play of agent i that coincides with strategy S_i (S_i', resp.) in all information sets I' following I such that i plays at most k times between I (ex-

cluded) and I' (included). In particular we simply say that S_i *obviously dominates* S'_i when $k = \infty$, and that it *strongly obviously dominates* S'_i when $k = 0$. If a strategy S_i (k-step / strongly) obviously dominates all other S'_i, then we say that S_i is *(k-step / strongly) obviously dominant*. If a mechanism implements f by guaranteeing that each player has a (k-step / strongly) obviously dominant strategy, we say that it is *(k-step / strongly) obviously strategy-proof* ((k-step / S) OSP).

Round-Table Mechanisms. Mackenzie [24] proved that for OSP mechanisms, it is without loss of generality to consider mechanisms $\mathcal{M}$ with a specific format, named *round table mechanisms*, defined as follows. $\mathcal{M}$ is defined by a triple $(f, p, \mathcal{T})$ where, as above, the pair (f, p) determines the outcome of the mechanism, and $\mathcal{T}$ is a tree, called *implementation tree*, such that:

– Every leaf ℓ is labeled with a possible outcome of the mechanism $(X(\ell), p(\ell))$, where $X(\ell) \in \mathcal{S}$ and $p(\ell) \in \mathbb{R}^n$;
– Each internal node u in the implementation tree $\mathcal{T}$ defines the following:
  • An agent $i = i(u)$ to whom the mechanism makes some query. Each possible answer to this query leads to a different child of u.
  • A subdomain $D(u) = (D_i(u), D_{-i}(u))$ containing all types profiles that are *compatible* with u (or available at u), i.e., with all the answers to the queries from the root down to node u. Specifically, the query at node u defines a partition of the current domain of i, $D_i(u)$ into $k \geq 2$ subdomains, one for each of the k children of node u. Thus, the domain of each of these children will have, as for the domain of i, the subdomain of $D_i(u)$ corresponding to a different answer of i at u, and an unchanged domain for the other agents. We also say that action at u *signals* the associated subdomain.

Observe that, according to the definition above, for every type profile $\mathbf{b} = (b_i \in D_i)_{i \in N}$ there is only one leaf $\ell = \ell(\mathbf{b})$ such that $\mathbf{b}$ belongs to $D^{(\ell)}$. Similarly, to each leaf ℓ there is at least a profile $\mathbf{b}$ that belongs to $D^{(\ell)}$. This allows as to simplify the notation: indeed, we can define $\mathcal{M}(\mathbf{b}) = (X(\ell), p(\ell))$ for $\ell = \ell(\mathbf{b})$. Similarly, we can simply write $f(\mathbf{b}) = (f_1(\mathbf{b}), \ldots, f_n(\mathbf{b})) = X(\ell)$ and $p(\mathbf{b}) = (p_1(\mathbf{b}), \ldots, p_n(\mathbf{b})) \in \mathbb{R}^n = p(\ell)$, and $u_i(t_i, \mathcal{M}(b_i, \mathbf{b}_{-i})) = p_i(b_i, \mathbf{b}_{-i}) - t_i(f(b_i, \mathbf{b}_{-i}))$. For the single-parameter setting, considered in this work, we can further simplify the notation, by setting $t_i(X) = t_i f_i(\mathbf{b})$ when we want to express the cost of a single-parameter agent i of type t_i for the output of social choice function f when the actions taken by the agent lead to the leaf ℓ associated with bid vector $\mathbf{b}$.

Two type profiles $\mathbf{b}, \mathbf{b}'$ are said to *diverge* (or to be separated) at a node u of $\mathcal{T}$ if this node has two children v, v' such that $\mathbf{b} \in D(v)$, whereas $\mathbf{b}' \in D(v')$. For every such node u, we say that $i(u)$ is the *divergent agent* at u. We sometimes abuse notation and we say that two types t_i and b_i of the agent $i = i(u)$ diverge (are separated) at u.

A round-table mechanism $\mathcal{M}$ is OSP if for every agent i with real type t_i, for every vertex u such that $i = i(u)$, for every $\mathbf{b}_{-i}, \mathbf{b}'_{-i}$ (with $\mathbf{b}'_{-i}$ not necessarily different from $\mathbf{b}_{-i}$), and for every $b_i \in D_i$, with $b_i \neq t_i$, such that $(t_i, \mathbf{b}_{-i})$ and $(b_i, \mathbf{b}'_{-i})$ are compatible with u, but diverge at u, it holds that $u_i(t_i, \mathcal{M}(t_i, \mathbf{b}_{-i})) \geq u_i(t_i, \mathcal{M}(b_i, \mathbf{b}'_{-i}))$. Roughly speaking, OSP requires that each time agent i is asked to take a decision that depends on her type, the worst utility that she can get if she henceforth behaves according to her true type is at least the best utility she can get by deviating.

We will extend this definition as follows. Let us first start with some useful definitions. For each agent i, and for each vertex u in the implementation tree $\mathcal{T}$ such that $i = i(u)$, and every u' along the path from u to a leaf, we say that u' belongs to the *k-step limit* $\mathscr{L}_k(u)$ of u if u' is the k-th node along this path (u excluded) in which i is queried or, if i is queried in less than k nodes along this path, then u' is the leaf. Moreover, for each agent i, and for each vertex u in the implementation tree $\mathcal{T}$ such that $i = i(u)$, the *k-step neighborhood* $\mathscr{N}_k(u)$ of u consists of each node of $\mathcal{T}$ that appears in the path between u and some $u' \in \mathscr{L}_k(u)$. Then, for each agent i and for each vertex u in the implementation tree $\mathcal{T}$ such that $i = i(u)$, two (not necessarily different) profiles $\mathbf{a}$ and $\mathbf{b}$ are said to be *k-step unseparated* at u if $\mathbf{a}$ and $\mathbf{b}$ do not diverge either at u or in every node in the k-step neighborhood of u. Finally, for each agent i and profile $\mathbf{a}$, and for each vertex u in the implementation tree $\mathcal{T}$ such that $i = i(u)$ and $\mathbf{a}$ is available at u, the *k-step equivalence class* of $\mathbf{a}$ at u is $\Gamma^k_u(\mathbf{a}) = \{\mathbf{b} \colon \mathbf{b}$ is k-step unseparated from $\mathbf{a}$ at $u\}$.

Then, a round-table mechanism $\mathscr{M}$ is *k-step OSP* if for every agent i with real type t_i, for every vertex u such that $i = i(u)$, for every $\mathbf{b}_{-i}, \mathbf{b}'_{-i}, \mathbf{b}''_{-i}$ (with $\mathbf{b}'_{-i}, \mathbf{b}''_{-i}$ not necessarily different from $\mathbf{b}_{-i}$ and different from each other), and for every $b_i, b'_i \in D_i$ such that $(t_i, \mathbf{b}_{-i}), (b_i, \mathbf{b}'_{-i}) \in \Gamma^k_u(t_i, \mathbf{b}_{-i})$, while $(t_i, \mathbf{b}_{-i})$ and $(b'_i, \mathbf{b}''_{-i})$ diverge at u, it holds $u_i(t_i, \mathscr{M}(b_i, \mathbf{b}'_{-i})) \geq u_i(t_i, \mathscr{M}(b'_i, \mathbf{b}''_{-i}))$. Roughly speaking, k-step OSP requires that, at each time step agent i is asked to take a decision, the worst utility that she can get if she now behaves according to her true type is at least the best utility she can get by behaving differently, *even if she later (i.e., after the next k queries received by the mechanism) changes her mind and plays as if her type was an untruthful but still available one.*

Mackenzie [24] proved that if there is an OSP mechanism implementing a social choice function f, then this can be also implemented by an OSP round-table mechanism. We show that this is true also for k-step OSP. We will thus focus on round-table mechanisms in this paper.

Theorem 1. *There is a k-step OSP mechanism implementing a social choice f if and only if f can be also implemented by a k-step OSP round-table mechanism.*

3 Characterizing the Implementation Tree

$\mathscr{M} = (f, \cdot, \mathcal{T})$ is said to be *almost-ordered*, if for every agent i, every node u of $\mathcal{T}$ such that $i = i(u)$, and every pair of profiles $\mathbf{a}$ and $\mathbf{b}$ separated at u such that $f_i(\mathbf{a}) > f_i(\mathbf{b})$ we have that $c_i < d_i$ where $c_i = \max\{x_i \mid \exists \mathbf{x}_{-i} \colon (x_i, \mathbf{x}_{-i}) \in \Gamma^k_u(\mathbf{a})\}$ and $d_i = \min\{x_i \mid \exists \mathbf{x}_{-i} \colon (x_i, \mathbf{x}_{-i}) \in \Gamma^k_u(\mathbf{b})\}$. Roughly speaking, $\mathscr{M}$ is almost ordered if for every pair L and R of the subsets in which $D_i(u)$ is partitioned at u, either $\mathscr{M}$ assigns the same outcome to all types in $L \cup R$ or these two sets are ordered (i.e., the types in one set are all smaller than the types in the other). Due to page limit, proofs are deferred to full version.

Lemma 1. *An extensive-form mechanism $\mathscr{M} = (f, p)$ with implementation tree $\mathcal{T}$ is k-step OSP if and only if for all i, all vertices u of $\mathcal{T}$ such that $i = i(u)$, and every pair*

of profiles $\mathbf{a}$ *and* $\mathbf{b}$ *separated at* u*, the following* k-step OSP constraint *holds for all* $(c_i, \mathbf{c}_{-i}) \in \Gamma_u^k(\mathbf{a})$*:*

$$p_i(b_i, \mathbf{b}_{-i}) - p_i(a_i, \mathbf{a}_{-i}) \leq c_i \left(f_i(b_i, \mathbf{b}_{-i}) - f_i(a_i, \mathbf{a}_{-i}) \right). \tag{1}$$

Theorem 2. *A* k*-step OSP mechanism* $\mathscr{M}$ *implementing social function* f *with implementation tree* $\mathscr{T}$ *is almost-ordered.*

Let us focus on settings with binary outcomes, i.e., $\mathscr{S} = \{0,1\}^n$ and consider a mechanism $\mathscr{M}$ implementing (f, p) with implementation tree $\mathscr{T}$. We next provide some useful definitions.

Definition 1 (Suffix/Prefix of Type Domains). *Given a node* $u \in \mathscr{T}$ *and agent* $i = i(u)$ *we say that the domain* $D_i(u)$ *is* prefix *if* $\max\{t \in D_i(u)\} < \min\{t \in D_i \setminus D_i(u)\}$*, i.e., it only contains the smaller types in the domain of* u *(with larger types already removed in the queries preceding* u*). Similarly, we say that* $D_i(u)$ *is* suffix *if* $\min\{t \in D_i(u)\} > \max\{t \in D_i \setminus D_i(u)\}$.

Definition 2 (Revelation, Extremal and (In)Effective Queries). *Given a node* $u \in \mathscr{T}$ *and a type* $t \in D_i(u)$*,* $i = i(u)$*, we say that the query at this node is a(n)*

Revelation query *if agent* i *is asked to reveal her type.*

Extremal query *if agent* i *is asked to separate one extreme (i.e., the minimum or maximum type) from the rest of her current domain* $D_i(u)$*.*

Ineffective query *if for every* $\mathbf{x}_{-i}$ *available at* u*, and every* $t', t'' \in D_i(u)$*,* $f_i(t', \mathbf{x}_{-i}) = f_i(t'', \mathbf{x}_{-i})$ *and* $p_i(t', \mathbf{x}_{-i}) = p_i(t'', \mathbf{x}_{-i})$*. In words, regardless of how types are partitioned by this query,* i *receives the same outcome and payment for every given profile* $\mathbf{x}_{-i}$*.*

Strongly Ineffective query *if for all* $\mathbf{x}_{-i}, \mathbf{x}'_{-i} \in D_{-i}(u)$ *and all* $t', t'' \in D_i(u)$*, we have* $f_i(t', \mathbf{x}_{-i}) = f_i(t'', \mathbf{x}'_{-i})$ *and* $p_i(t', \mathbf{x}_{-i}) = p_i(t'', \mathbf{x}'_{-i})$*. In words, in such a query agent* i *receives the same outcome and payment for each possible profile available at that history.*

Only-t **effective query** *if for each* $\mathbf{x}_{-i}$ *available at* u *and every* $t', t'' \in D_i(u)$ *with* $t', t'' \neq t$*,* $f_i(t', \mathbf{x}_{-i}) = f_i(t'', \mathbf{x}_{-i})$ *and* $p_i(t', \mathbf{x}_{-i}) = p_i(t'', \mathbf{x}_{-i})$*, and there is* $\mathbf{y}_{-i}$ *available at* u *such that* $f_i(t, \mathbf{y}_{-i}) \neq f_i(t', \mathbf{y}_{-i})$*. In words, for each profile of other agents' actions, the outcome and payment received by* i *is the same except for the type* t*.*

Strongly only-t **effective query** *if for each* $\mathbf{x}_{-i}, \mathbf{x}'_{-i}$ *available at* u *and every* $t', t'' \in D_i(u)$ *such that* $t', t'' \neq t$*,* $f_i(t', \mathbf{x}_{-i}) = f_i(t'', \mathbf{x}'_{-i})$ *and* $p_i(t', \mathbf{x}_{-i}) = p_i(t'', \mathbf{x}'_{-i})$*, and there is* $\mathbf{y}_{-i}$ *available at* u *such that* $f_i(t, \mathbf{y}_{-i}) \neq f_i(t', \mathbf{y}_{-i})$*. That is,* i *receives exactly the same outcome and payment in each profile available at* u*, except for the ones in which she has type* t*.*

We concentrate on (strongly) only-maximum (minimum, resp.) effective queries, that is, the (strongly) only-t *queries for which* $t = \max\{t \in D_i(u)\}$ *(*$t = \min\{t \in D_i(u)\}$*, resp.).*

Definition 3 (k**-limited mechanism).** *A mechanism* $\mathscr{M}$ *implemented by a tree* $\mathscr{T}$ *is* k-limited *if for each agent* i*, and for every path* P *from the root of* $\mathscr{T}$ *to a leaf, one of the following holds:*

- i *is queried at most* $k+1$ *times along* P*;*

– *i is queried at most $k+2$ times along P* and *at the node u corresponding to the $(k+2)$-th query: (i) either $|D_i(u)| = 2$ or $D_i(u)$ is prefix; and (ii) the query at u is either a strongly ineffective revelation query, or a strongly only-maximum effective revelation query or an only-maximum effective extremal query that separates the maximum type in $D_i(u)$ from the rest of the domain;*
– *i is queried at most $k+2$ times along P* and *at the node u corresponding to the $(k+2)$-th query we have that: (i) $D_i(u)$ is suffix; and (ii) the query at u is either a strongly ineffective revelation query, a strongly only-minimum effective revelation query or it is an only-minimum effective extremal query that separates the minimum type in $D_i(u)$ from the rest of the domain.*

Theorem 3. *If there is a k-step OSP mechanism $\mathscr{M}$ that implements (f,p) with implementation tree $\mathscr{T}$, then there is a k-limited k-step OSP mechanism $\mathscr{M}'$ implementing (f,p).*

The rest of this section proves Theorem 3. We first prove some preliminary properties of a k-step OSP mechanism $\mathscr{M}$, which amount to a version of the taxation principle for k-step OSP.

Lemma 2 (Taxation Principle for k-step OSP). *Let $\mathscr{M}$ be a k-step OSP mechanism that implements (f,p) with implementation tree $\mathscr{T}$. For all i and for each vertex u in $\mathscr{T}$ such that $i = i(u)$, take any three profiles $\mathbf{a} = (a_i, \mathbf{a}_{-i})$, $\mathbf{c} = (c_i, \mathbf{a}_{-i})$, and $\mathbf{d} = (d_i, \mathbf{a}_{-i})$ such that (i) $a_i < c_i < d_i$, (ii) $\mathbf{a}, \mathbf{c}, \mathbf{d} \in \Gamma_u^k(\mathbf{a})$, and (iii) there is $u' \notin \mathscr{N}_k(u)$ such that $i = i(u')$, $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{d}$ are available at u', and two among $\mathbf{a}$, $\mathbf{c}$, and $\mathbf{d}$ are separated at u'. We have that:*

1. *if there are $\mathbf{b}$ and $\mathbf{b}'$ separated from $\mathbf{a}, \mathbf{c}, \mathbf{d}$ by i along the path from u to u' such that $b_i > d_i$ and $b'_i < a_i$, then $f_i(\mathbf{a}) = f_i(\mathbf{c}) = f_i(\mathbf{d})$ and $p_i(\mathbf{a}) = p_i(\mathbf{c}) = p_i(\mathbf{d})$ (Outer-sandwich separations);*
2. *if there is $\mathbf{b}$ separated from $\mathbf{a}, \mathbf{c}, \mathbf{d}$ by i along the path from u to u' such that $a_i < b_i < d_i$, then $f_i(\mathbf{a}) = f_i(\mathbf{c}) = f_i(\mathbf{d})$ and $p_i(\mathbf{a}) = p_i(\mathbf{c}) = p_i(\mathbf{d})$ (Inner-sandwich separation);*
3. *if there is $\mathbf{b}$ separated from $\mathbf{a}, \mathbf{c}, \mathbf{d}$ by i along the path from u to u' such that $b_i > d_i$ then $f_i(\mathbf{a}) = f_i(\mathbf{c})$ and $p_i(\mathbf{a}) = p_i(\mathbf{c})$ (Top separation);*
4. *if there is $\mathbf{b}$ separated from $\mathbf{a}, \mathbf{c}, \mathbf{d}$ by i along the path from u to u' such that $b_i < a_i$ then $f_i(\mathbf{c}) = f_i(\mathbf{d})$ and $p_i(\mathbf{c}) = p_i(\mathbf{d})$ (Bottom separation).*

Mechanism $\mathscr{M}$ with implementation tree $\mathscr{T}$ is *almost k-limited* if $\mathscr{T}$ is such that for each agent i and every path P from the root of $\mathscr{T}$ to a leaf, the following conditions are satisfied:

– for $u \in P$ corresponding to the $(k+2)$-th query to i, if we denote with a_i and d_i the minimum and the maximum of the types of i available at u respectively, we must have that
  • if $D_i(u)$ contains at least three types and it is neither prefix nor suffix (i.e., in the previous $k+1$ queries to i, either i separated from $D_i(u)$ types b_i and b'_i such that $b_i > d_i$ and $b'_i < a_i$, or i separated from $D_i(u)$ type b_i such that $a_i < b_i < d_i$), then the $(k+2)$-th query is ineffective;

- if $D_i(u)$ contains at most two types or it is prefix (i.e., in the previous $k+1$ queries to i, i separated from $D_i(u)$ only types b_i such that $b_i > d_i$), then the $(k+2)$-th query is either ineffective or it is only-maximum effective;
- if $D_i(u)$ contains at least three types and it is suffix (i.e., in the previous $k+1$ queries to i, i separated from the domain only types b_i such that $b_i < a_i$), then the $(k+2)$-th query is either ineffective or it is only-minimum effective.

– every $u \in P$ corresponding to the q-th query to i with $q \geq k+3$ are ineffective.

Considering the first query along a path P, Lemma 2 restricts the $(k+2)$-th query on P as in the first bullet point (the three cases corresponding to sandwich, top, and bottom separations, respectively). Moreover, given these properties, the subsequent queries must be ineffective (as requested by the second condition of almost k-limited mechanisms). Thus, Lemma 2 can be restated as follows.

Corollary 1. *If a mechanism $\mathcal{M}$ that implements (f,p) with implementation tree $\mathcal{T}$ is k-step OSP, then $\mathcal{T}$ is almost k-limited.*

We further limit the effectiveness of the $(k+2)$-th and following queries: if a query is ineffective for a pair of separated types then it is strongly ineffective for those types.

Lemma 3 (Strong Ineffectiveness upon Separation). *If a mechanism $\mathcal{M}$ that implements (f,p) with implementation tree $\mathcal{T}$ is k-step OSP, then for every i, and every path P of $\mathcal{T}$, every node $u \in P$ such that $i(u) = i$, if there are $t, t' \in D_i(u)$ such that for every $\mathbf{x}_{-i}$ available at u we have that $f_i(t,\mathbf{x}_{-i}) = f_i(t',\mathbf{x}_{-i})$, and t, t' are separated at u, then for every $\mathbf{x}_{-i}, \mathbf{x}'_{-i}$ available at u we have that $f_i(t,\mathbf{x}_{-i}) = f_i(t',\mathbf{x}'_{-i})$ and $p_i(t,\mathbf{x}_{-i}) = p_i(t',\mathbf{x}'_{-i})$.*

Lemma 3 states that if in path P of $\mathcal{T}$ the $(k+2)$-th query to i is ineffective, then it must be strongly ineffective. Similarly, if in some path P of $\mathcal{T}$ the $(k+2)$-th query to i is only-maximum (only-minimum, resp.) effective, and the types of i different from the maximum (minimum, resp.) are separated at the $(k+2)$-th or successive query, then the $(k+2)$-th query is strongly only-maximum (only-minimum, resp.) effective. In order to stress that in $\mathcal{T}$ this property must be satisfied, we say that the corresponding mechanism is *strongly almost k-limited*. Note that the $(k+2)$-th query of a strongly almost k-limited mechanism can still be extremal if this is the last query.

Lemmas 2 and 3 imply that a k-step OSP mechanism must be strongly almost k-limited. However, in such a mechanism we are still allowed to query agent i more than $k+2$ times, or to have a $(k+2)$-th query that is neither a revelation nor an extremal query. However we can show that it is possible to transform $\mathcal{T}$ in order to achieve the desired reduction to a k-limited mechanism. In particular, we will see that we can always guarantee that the $(k+2)$-th is a revelation query. Theorem 3 then follows from the repeated application of these transformations, since for each path, agent i is either queried at most $k+1$ times, or she is queried $k+2$ times, the last being a revelation or an extremal query.

4 Cycle-Monotonicity for k-step OSP

For each i, for each path P from the root of $\mathcal{T}$ to the node u corresponding to the $(k+2)$-th query to i, if it exists, or to a leaf otherwise, we partition the domain D_i of

types of i in classes $D^1_{i,P},\dots,D^\ell_{i,P}$, where $\ell=\min\{k+2,q+1\}$, q denoting the number of queries to i along this path; $D^j_{i,P}$, for $j<\ell$ contains all types of i available at the j-th query at i, but not at the $(j+1)$-th query to i along P, and $D^\ell_{i,P}=D_i\setminus\bigcup_{j=1}^{\ell-1}D^j_{i,P}$.

Suppose that, along the path P, i is queried at least $k+2$ times, and let u be the node corresponding to the $(k+2)$-th query at i. Let $D^{k+2,\overline{E}}_{i,P}$ be a maximal set of types of i available at u that receive the same outcome for each possible fixed profile of the remaining agents available at u, i.e., $D^{k+2,\overline{E}}_{i,P}\subseteq D^{k+2}_{i,P}$ such that for every $t,t'\in D^{k+2,\overline{E}}_{i,P}$ and every $\mathbf{x}_{-i}$ available at u we have that $f_i(t,\mathbf{x}_{-i})=f(t',\mathbf{x}_{-i})$, and for every $t\in D^{k+2,\overline{E}}_{i,P}$ and every $t''\in D^{k+2,E}_{i,P}=D^{k+2}_{i,P}\setminus D^{k+2,\overline{E}}_{i,P}$ there is $\mathbf{y}_{-i}$ available at u for which $f_i(t,\mathbf{y}_{-i})\neq f_i(t'',\mathbf{y}_{-i})$. Note that if the $(k+2)$-th query is (strongly) ineffective, we have that $D^{k+2,\overline{E}}_{i,P}=D^{k+2}_{i,P}$. Moreover, if the $(k+2)$-th query is (strongly) only-maximum or only-minimum effective, then $D^{k+2,E}_{i,P}$ contains only the one extremal type for which the $(k+2)$-th query is effective.

Given this partition of the type domain of agent i, we then partition the profiles in equivalence classes as follows: for each agent i, for each path P from the root of $\mathcal{T}$ to a node u corresponding to the $(k+2)$-th query to i, if it exists, or to a leaf otherwise, we define the equivalence classes $\Lambda^j_{i,P}(b)=\{(x_i,\mathbf{x}_{-i})\mid x_i\in D^j_{i,P},\ \mathbf{x}_{-i}\ \text{available at}\ u,\ f_i(x_i,\mathbf{x}_{-i})=b\}$, for $b\in\{0,1\}$ and $j=q+1$ if in P agent i is queried at most $q\leq k+1$ times, and $j\in\{(k+2,E),(k+2,\overline{E})\}$ otherwise. Let Λ_i be the set that contains all the equivalence classes defined for agent i. Moreover, we abuse notation and we set $f_i(\Lambda^j_{i,P}(0))=0$ and $f_i(\Lambda^j_{i,P}(1))=1$.

Definition 4. *(k-step OSP-graph) Let f be a social choice function and $\mathcal{T}$ be an implementation tree. We define for every agent i, the* k-step OSP-graph $\mathcal{O}^k_{i,f,\mathcal{T}}$ *with $\Lambda_{\mathbf{i}}$ as the set of vertices, and an edge e between $\Lambda,\Lambda'\in\Lambda_{\mathbf{i}}$ exists if there are $\mathbf{x}\in\Lambda$ and $\mathbf{x}'\in\Lambda'$ such that $\mathbf{x}$ and $\mathbf{x}'$ have been separated in $\mathcal{T}$ by i. Moreover, we set $w(e)=\min_{x_i\mid\exists\mathbf{x}_{-i}:\,(x_i,\mathbf{x}_{-i})\in\Lambda}x_i(f_i(\Lambda')-f_i(\Lambda))$.*

We say that the k-step OSP cycle monotonicity (k-step OSP CMON) property holds if, for all i, the graph $\mathcal{O}^k_{i,f,\mathcal{T}}$ does not have negative weight cycles.

Theorem 4. *A mechanism $\mathcal{M}$ with implementation tree $\mathcal{T}$ is k-step OSP for a social function f on finite domains if and only if it is k-limited and k-step OSP CMON holds.*

5 Algorithmic Characterization

We start by recalling the characterization for $k=\infty$ [13]: there is an OSP mechanism implementing f if and only f is *two-way greedy implementable*. Let us define this notion. Agent i is *revealable* at node u under social choice function f if either $f_i(\mathbf{x})=1$ for every $\mathbf{x}$ such that $x_i\in D_i(u)$ and $x_i<\max D_i(u)$, and $\mathbf{x}_{-i}$ is compatible with u, or $f_i(\mathbf{x})=0$ for every $\mathbf{x}$ such that $x_i\in D_i(u)$ and $x_i>\min D_i(u)$, and $\mathbf{x}_{-i}$ is compatible with u. Algorithm $f\colon\times_{i=1}^n D_i\to\mathcal{S}$ is *two-way greedy implementable* if it can be implemented by a round-table mechanism with implementation tree $\mathcal{T}$ such that:

– at each node u the agent $i = i(u)$ separates her domain $D_i(u)$ in two ordered subsets $L_i(u)$ and $R_i(u)$, i.e., $\max L_i(u) < \min R_i(u)$, such that at least one of the two is a singleton, i.e., it contains a single type, being the minimum in $D_i(u)$ (if $L_i(u)$ is a singleton) or the maximum (if $R_i(u)$ is a singleton). If $L_i(u)$ is a singleton, then the agent i receives outcome 1 in every profile compatible with $L_i(u) \times D_{-i}(u)$, and we say that i interacts with the mechanism in a greedy fashion. If $R_i(u)$ is a singleton, then i receives outcome 0 in every profile compatible with $R_i(u) \times D_{-i}(u)$, and we say that i interacts with the mechanism in a reverse greedy fashion;
– each agent i is not allowed to interleave interaction in a greedy fashion with interaction in a reverse greedy fashion until she is revealable. Specifically, agent i at node u is either revealable, or can interact with the mechanism in a greedy fashion if and only if she i interacted with the mechanism in a greedy fashion at every node u' along the path between the root of $\mathcal{T}$ and u such that $i = i(u')$.

While the definition of two-way greedy implementable mechanism appears to be quite involved, it establishes a very strong relationship between two-way greedy implementable algorithms and greedy algorithms [13]. Essentially every greedy algorithm and every reverse greedy algorithm (a.k.a., deferred-acceptance algorithm) is two-way greedy, and hence can be turned in an OSP mechanism. This continues to hold if the algorithm is allowed to greedily insert into the solutions some components (i.e., interact greedily with some agents) and reverse greedily remove from the solutions other components (i.e., interact reverse greedily with other agents). We will finally observe that whenever an agent is revealable at some node u, we can ask the agent to reveal her type without affecting the OSPness of the mechanism.

We can prove there is a useful relation between negative cycles in the k-step OSP-graph, and negative cycles in the ∞-step OSP-graph: there is no negative cycle in $\mathcal{O}^k_{i,f,\mathcal{T}}$ if and only if there is no negative cycle in $\mathcal{O}^\infty_{i,f,\mathcal{T}}$. However, two-way greedy implementation of f assumes that the implementation tree $\mathcal{T}$ is binary and makes only extremal queries. Clearly, each implementation tree $\mathcal{T}'$ for which there are no negative cycles in $\mathcal{O}^\infty_{i,f,\mathcal{T}'}$, regardless of the kind of queries that are performed in $\mathcal{T}'$, can be transformed in an implementation tree $\mathcal{T}$ with binary extremal queries that still has no negative cycle through a *serialization* procedure (see [13, Observation 3 and Theorem 4]). Similarly, a binary implementation tree $\mathcal{T}$ with extremal queries and no negative cycle in $\mathcal{O}^\infty_{i,f,\mathcal{T}}$ can be transformed in an implementation tree $\mathcal{T}'$ with generic queries but still no negative cycle through a simple *compression* procedure: every two consecutive nodes u and u' such that $i = i(u) = i(u')$ can be merged in a single node U with outgoing edges leading to $v_1, \ldots, v_x$, and to $v'_1, \ldots, v'_y$, where $v_1, \ldots, v_x$ are the children of u different from u', and $v'_1, \ldots, v'_y$ are the children of u'. However, these serialization/compression operations will change the number of queries done to each agent. While this does not matter for ∞-step OSPness, it turns out to be very relevant for k-step OSPness. Hence, we will introduce a way to keep the number of queries to a given agent unchanged after the operations of serialization/compression. Specifically, we say that a binary implementation tree $\mathcal{T}$ with extremal queries is *k-limitable* if and only if the implementation tree $\mathcal{T}'$ achieved through the compression procedure described above is *k-limited*. (For convenience

we are abusing a bit our terminology by calling k-limited the implementation tree $\mathscr{T}'$ rather than the mechanism $\mathscr{M}$ using $\mathscr{T}'$.)

Theorem 5. *There is a k-step OSP mechanism implementing f iff f is two-way greedy implementable and the corresponding implementation tree $\mathscr{T}$ is k-limitable.*

Essentially, Theorem 5 states that the OSP characterization in term of greedy algorithms continues to hold even for k-step OSPness. However, we here further require a constraint to be satisfied, i.e., that the implementation tree is k-limited (in its compressed version) or k-limitable (in the serialized version). The effect of this limitation is very heavy in the case of SOSP mechanisms.

6 Approximation Guarantee of k-step OSP Mechanisms

In this section we apply our characterization to quantify the restriction that k-step OSP imposes on the quality of the algorithmic solution that can be implemented. We will focus on maximization problems; agents' types will thus be a non-negative valuation (i.e., a non-positive cost) for each algorithmic allocation received.

To introduce our questions of interest, we start by discussing *single-item auctions*: n agents have a valuation $v_i \in D_i$ for the item, and we are willing to sell the item to the agent with the highest valuation. Note that in order to fully define the social function f for this problem, we need to specify how ties are broken. We will assume that ties are broken in favor of the agent with the smallest index. We will also assume for simplicity that every agent i has a domain $D_i = D$. The ascending price auction discussed in Example 1 is an OSP mechanism that solves this problem optimally. We can rephrase it in terms of two-way greedy implementation as follows: mark all agents as available; for t from the smallest type to the second largest type in D or until there is only one available agent, ask each available agent i in order of their index whether her type is t, and in case of positive answer, mark the agent as unavailable; assign the item to the agent with smallest index among the available ones. Actually, this is not the unique OSP auction implementing the social function of interest (e.g., we can query agents from the largest to the second smallest type and allocate the item upon a positive answer).

Our question is then: can we implement single-item auctions with a k-step OSP mechanism? If not, how large can be the *price of limited foresight*? To answer these questions, observe that the English auction described above queries agents $|D|-1$ times, the last query being a revelation strongly only-minimum effective query. Hence, this mechanism is k-limitable, and, by Theorem 5, k-step OSP for $k \geq |D|-3$.

However, the mechanism fails to be $(|D|-4)$-limited. Indeed, after the $(|D|-3)$-th query to agent i, her domain contains the three largest types, and for each action of other agents compatible with the $(|D|-3)$-th query, the outcome of this agent should be the same when agent i has the smallest and the second smallest type. But the ascending price auction described above does not provide such a guarantee. Can there be another k-step OSP mechanism implementing the social choice function f? Or some social function f' differing from f only in the tie-breaking rule (and hence, still returning an allocation that maximizes the social welfare)? We next address these questions in a more general setting than single-item auctions.

p-systems. We will now focus on the class of problems that satisfy the *downward closed property*. In these problems we are given a set E of elements, a weight $w(e) \in \mathbb{R}$ for each $e in E$, and a set $\mathscr{F}$, containing subsets $S \subseteq E$, named *feasible solutions*, that enjoy the following property: if $S \in \mathscr{F}$, then $T \in \mathscr{F}$ for every $T \subseteq S$, and we aim to select the feasible solution of maximum total weight, i.e. $S^* = \arg\max_{S \in \mathscr{F}} \sum_{e \in S} w(e)$.

Specifically, we focus on *p-systems*: these are special problems for which it is known that the *greedy* algorithm, that processes elements e in decreasing order of their weight, and includes them in the current solution S unless $S \cup \{e\} \notin \mathscr{F}$, and the *reverse greedy* algorithm, that processes elements e in increasing order of their weight, and remove any solution S containing e from the set $\mathscr{O}$ of maximal feasible solutions unless this empties $\mathscr{O}$, have approximation p. It is easy to check that the problem of social-welfare single-item auction is an example of 1-system (a.k.a., matroid). (For a formal definition of p-systems, we refer the reader to [19].) This, in turn, means that for all these problems, it is possible to design an OSP mechanism that is able to return a p-approximate feasible solution for the problem [13].

We defer to the full version the proof that it is always possible to implement the reverse greedy algorithm defined above as a k-limitable two-way greedy algorithm when $k \geq \left\lceil \frac{|D|}{2} \right\rceil - 2$, and thus, by Theorem 5, that a k-step OSP algorithm exists that returns a p-approximate solution for every p-system. We also show that, if no constraint is given on the values in D, then no k-step OSP mechanism exists that is able to return a bounded approximation of the optimal solution, whenever $k < \left\lceil \frac{|D|}{2} \right\rceil - 2$.

Theorem 6. *There is a two-way greedy algorithm, which (i) is k-limitable for $k \geq \left\lceil \frac{|D|}{2} \right\rceil - 2$; and, (ii) returns a p-approximation of the feasible set of maximum weight for every p-system. Conversely, for every $\rho > 0$, every $d = |D| \geq 5$, and every $k < \frac{d}{2} - 2$ ($k < d - 3$, resp.), there is a p-system $(E, \mathscr{F})$ such that no k-limitable two-way greedy (greedy, resp.) algorithm returns a feasible solution of total weight not larger than a $\frac{1}{\rho}$ fraction of the total weight of the optimal feasible solution.*

7 Conclusions

In this work, we have studied the algorithmic robustness of OSP to agents that are not able to perform contingent reasoning and think about their future actions. Specifically, we introduce a novel notion, termed k-step OSP, that smoothens the notions of OSP (where absence of contingent reasoning is the only cognitive limitation) and SOSP (where in addition agents are unable to think about any of their future actions) by maintaining the assumption that agents are not able to think contingently but allowing them a foresight of k self moves ahead. We provide an algorithmic characterization of these mechanisms for single-dimensional agents and binary outcomes, via the introduction of a new cycle monotonicity toolkit. We apply our characterization to downward-closed maximization problems and prove that the performance of OSP can deteriorate when k is small in comparison to the type space of the agents. En route, we prove that reverse greedy algorithms are more robust than greedy algorithms to this worsening of the performances.

A natural open problems left by this work is to understand the extent to which the findings above hold for non-binary allocation problems, and quantify the limitations of limited planning horizons for other optimization problems, such as scheduling related machines that is now fully understood for OSP [18]. We highlight that our framework can also be adopted for non-binary allocation problems to prove that queries from the $(k+2)$-th onward must be limited, but this does not necessarily lead to mechanisms with only $k+2$ queries since the Taxation Principle for k-step OSP would be less clean as the relative weight of types and outcomes would matter.

Acknowledgements. Diodato Ferraioli is supported by the PNRR project FAIR – Future AI Research (PE00000013) under the NRRP MUR program funded by NextGenerationEU. Carmine Ventre acknowledges funding from the UKRI Trustworthy Autonomous Systems Hub (EP/V00784X/1).

References

1. Archbold, T., De Keijzer, B., Ventre, C.: Non-obvious manipulability for single-parameter agents and bilateral trade. In: Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023) (2023)
2. Archbold, T., De Keijzer, B., Ventre, C.: Non-obvious manipulability in extensive-form mechanisms: the revelation principle for single-parameter agents. In: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023) (2023)
3. Archbold, T., De Keijzer, B., Ventre, C.: Willy wonka mechanisms. In: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024) (2024)
4. Arribillaga, R., Massó, J., Neme, A.: On obvious strategy-proofness and single-peakedness. JET (2020)
5. Arribillaga, R.P., Massó, J., Neme, A.: All sequential allotment rules are obviously strategy-proof. Theoretical Economics **18**(3), 1023–1061 (2023)
6. Ashlagi, I., Gonczarowski, Y.A.: Stable matching mechanisms are not obviously strategy-proof. Journal of Economic Theory **177**, 405–425 (2018)
7. Ausubel, L.M.: An efficient ascending-bid auction for multiple objects. American Economic Review **94**(5), 1452–1475 (2004)
8. Bade, S., Gonczarowski, Y.A.: Gibbard-satterthwaite success stories and obvious strategyproofness. In: Proceedings of the 2017 ACM Conference on Economics and Computation. p. 565. EC '17, Association for Computing Machinery, New York, NY, USA (2017)
9. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: Automated optimal OSP mechanisms for set systems - the case of small domains. In: Web and Internet Economics - 15th International Conference, WINE 2019, New York, NY, USA, December 10-12, 2019, Proceedings. pp. 171–185 (2019)
10. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: Obviously strategyproof mechanisms for machine scheduling. In: Bender, M.A., Svensson, O., Herman, G. (eds.) 27th Annual European Symposium on Algorithms, ESA 2019. vol. 144, pp. 46:1–46:15 (2019)
11. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: Obviously strategyproof mechanisms for machine scheduling. In: 27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany. pp. 46:1–46:15 (2019)
12. Ferraioli, D., Meier, A., Penna, P., Ventre, C.: New constructions of obviously strategyproof mechanisms. Math. Oper. Res. **48**(1), 332–362 (2023)

13. Ferraioli, D., Penna, P., Ventre, C.: Two-way greedy: Algorithms for imperfect rationality. In: Web and Internet Economics - 17th International Conference, WINE 2021, Potsdam, Germany, December 14-17, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13112, pp. 3–21. Springer (2021)
14. Ferraioli, D., Ventre, C.: Probabilistic verification for obviously strategyproof mechanisms. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 240–246 (2018)
15. Ferraioli, D., Ventre, C.: Approximation guarantee of OSP mechanisms: The case of machine scheduling and facility location. Algorithmica **83**(2), 695–725 (2021)
16. Ferraioli, D., Ventre, C.: Obvious strategyproofness, bounded rationality and approximation. Theory Comput. Syst. **66**(3), 696–720 (2022)
17. Ferraioli, D., Ventre, C.: Explicit payments for obviously strategyproof mechanisms. In: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023. pp. 21257–21336. ACM (2023)
18. Ferraioli, D., Ventre, C.: On the connection between greedy algorithms and imperfect rationality. In: Leyton-Brown, K., Hartline, J.D., Samuelson, L. (eds.) Proceedings of the 24th ACM Conference on Economics and Computation, EC 2023, London, United Kingdom, July 9-12, 2023. pp. 657–677. ACM (2023)
19. Hausmann, D., Korte, B., Jenkyns, T.A.: Worst case analysis of greedy type algorithms for independence systems. Combinatorial Optimization pp. 120–131 (1980)
20. Kagel, J.H., Harstad, R.M., Levin, D.: Information impact and allocation rules in auctions with affiliated private values: A laboratory study. Econometrica **55**(6), 1275–1304 (1987)
21. de Keijzer, B., Kyropoulou, M., Ventre, C.: Obviously strategyproof single-minded combinatorial auctions. In: ICALP. pp. 71:1–71:17 (2020)
22. Kyropoulou, M., Ventre, C.: Obviously strategyproof mechanisms without money for scheduling. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1574–1581 (2019)
23. Li, S.: Obviously strategy-proof mechanisms. American Economic Review **107**(11), 3257–87 (November 2017)
24. Mackenzie, A.: A revelation principle for obviously strategy-proof implementation. Games and Economic Behavior **124**, 512–533 (2020)
25. Mandal, P., Roy, S.: On obviously strategy-proof implementation of fixed priority top trading cycles with outside options. Economics Letters **211**, 110239 (2022)
26. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.): Algorithmic Game Theory (2017)
27. Pycia, M., Troyan, P.: Obvious dominance and random priority. In: Proceedings of the 2019 ACM Conference on Economics and Computation. p. 1. EC '19, Association for Computing Machinery, New York, NY, USA (2019)
28. Pycia, M., Troyan, P.: A theory of simplicity in games and mechanism design. Econometrica **91**(4), 1495–1526 (2023)
29. Ron, S.: Impossibilities for obviously strategy-proof mechanisms. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA) (2024)
30. Savage, L.: Foundations of Statistics. John Wiley and Sons (1954)
31. Thomas, C.: Classification of priorities such that deferred acceptance is osp implementable. In: EC. pp. 860–860 (2021)
32. Troyan, P.: Obviously strategy-proof implementation of top trading cycles. International Economic Review **60**(3), 1249–1261 (2019)
33. Troyan, P., Morrill, T.: Obvious manipulations. Journal of Economic Theory **185**, 104970 (2020)