Journal of Artificial Intelligence Research 62 (2018) 459-488 Submitted 2/18; published 7/18

The Power of Verification for Greedy Mechanism Design

Dimitris Fotakis FOTAKIS@CS.NTUA.GR
National Technical University of Athens, Greece

Piotr Krysta PKRYSTA@LIVERPOOL.AC.UK
University of Liverpool, UK

Carmine Ventre C.VENTRE@ESSEX.AC.UK
University of Essex, UK

Abstract

Greedy algorithms are known to provide, in polynomial time, near optimal approximation guarantees for Combinatorial Auctions (CAs) with multidimensional bidders. It is known that truthful greedy-like mechanisms for CAs with multi-minded bidders do not achieve good approximation guarantees.

In this work, we seek a deeper understanding of greedy mechanism design and investigate under which general assumptions, we can have efficient and truthful greedy mechanisms for CAs. Towards this goal, we use the framework of priority algorithms and weak and strong verification, where the bidders are not allowed to overbid on their winning set or on any subset of this set, respectively. We provide a complete characterization of the power of weak verification showing that it is sufficient and necessary for any greedy fixed priority algorithm to become truthful with the use of money or not, depending on the ordering of the bids. Moreover, we show that strong verification is sufficient and necessary to obtain a 2-approximate truthful mechanism with money, based on a known greedy algorithm, for the problem of submodular CAs in finite bidding domains. Our proof is based on an interesting structural analysis of the strongly connected components of the declaration graph.

1. Introduction

Algorithmic mechanism design (AMD) is the study of optimization problems where part of the input data is private data of selfish agents. We are interested in a *truthful mechanism*; the goals here are to incentivize the agents to truthfully report their private data to the mechanism and to optimize the objective function of the problem under consideration. The incentives are typically adjusted by means of *payments* (i.e., monetary transfers between mechanism designer and agents). Therefore truthfulness requires the design of a suitable pair of algorithm and payments, computing the outcome to the optimization problem and the transfers respectively. AMD problems have numerous real-world applications which are chiefly facilitated by the use of large-scale Internet systems in areas like, e.g., e-commerce, online auctions and shopping systems.

Combinatorial Auctions (CAs) is a paradigmatic AMD problem. In a CA, we have a set U of m goods and a set of n bidders. Each bidder i has a *private* valuation function (also termed *type*) v_i mapping subsets of goods to nonnegative numbers. Valuations are monotone, i.e., for $S \supseteq T$, $v_i(S) \geq v_i(T)$, and normalized, i.e., $v_i(\emptyset) = 0$. The goal is to find a partition $S_1, \ldots, S_n$ of U such that the *social welfare* $\sum_{i=1}^{n} v_i(S_i)$ is maximized. Even the simplest versions of CAs are known to be **NP**-hard and even hard to approxi-

mate in polynomial time. For pointers to the vast literature on the design of approximate polynomial-time mechanisms for various versions of CAs (and, more generally, mechanism design, see Nisan, Roughgarden, Tardos, & Vazirani, 2007).

We study CAs with bidders' valuation functions represented by *value oracles*, i.e., each bidder i submits such an oracle that given any subset of goods $S \subseteq \mathsf{U}$ outputs the value $v_i(S)$. This is weaker than the other standard model, demand queries, used in the literature to tackle the issue of exponentially (in n and m) large valuations (Blumrosen & Nisan, 2009). We aim at the design of truthful mechanisms that only use a polynomial number of calls to such value oracles and return solutions with best possible approximation guarantees with respect to social welfare. For CAs with single-minded bidders[1] a simple greedy truthful mechanism achieves a $\sqrt{m}$-approximation (Lehmann, O'Callaghan, & Shoham, 2002), the best possible guarantee in polynomial-time unless $P = NP$.

However, as soon as the valuation functions become more general than single-minded, greedy truthful mechanisms perform much worse. Borodin and Lucier (2016) studied limitations of a general class of greedy algorithms, *priority algorithms*, for truthful CAs with non single-minded bidders. They proved that any such truthful priority mechanism (i.e., mechanism using a priority algorithm to compute its outcome) has an approximation ratio $\Omega(m)$. The results of Borodin and Lucier (2016) imply that no truthful priority mechanism achieves a good approximation ratio.

A widely studied class of CAs with non single-minded, thus multidimensional, bidders assumes submodular valuation functions, modeling economies of scale. A well known greedy approximation algorithm for submodular CAs gives an approximation guarantee of 2 (Lehmann, Lehmann, & Nisan, 2006). This algorithm simply processes the goods in any fixed order and awards them to the bidder who has maximum marginal value for each good. However, there are no payments that make this algorithm truthful (Lehmann et al., 2006). More generally, Borodin and Lucier (2016) considered the whole class of such greedy mechanisms (defined by a fixed order of the goods used) for submodular CAs and proved that no mechanism in this class is truthful!

Motivated by the dichotomy between a powerful algorithmic technique and its poor compatibility with bidders' incentives, we investigate herein under which general circumstances, we can have truthful multi-dimensional CAs that use greedy algorithms. Inspired by Krysta and Ventre (2015), we focus on relaxations of truthfulness wherein some form of overbidding can be detected and prevented. This line of research, generally dubbed mechanisms with *verification*, is motivated by a number of real-life applications and has been considered by economists in the context of CAs (see, e.g., Celik, 2006). We want to understand what kind of relaxed notions of incentive-compatibility, i.e., verification, we can adopt to happily marry greedy algorithms with good approximation guarantees and truthfulness. A relaxation of truthfulness is necessary in general for efficient CAs with multidimensional bidders for otherwise no approximation better than m can be achieved, as recently proved (Daniely, Schapira, & Shahaf, 2015).

1. Bidder i is single-minded if there exists a fixed subset $S_i \subseteq \mathsf{U}$ and a number $w_i \geq 0$ such that $v_i(S) = w_i$ for any S such that $\mathsf{U} \supseteq S \supseteq S_i$, and $v_i(S) = 0$ otherwise.

1.1 Our Contribution

CAs with verification are perhaps best illustrated by means of the following motivating scenario, discussed first by Krysta and Ventre (2015). Consider a government (auctioneer) auctioning business licenses for a set U of cities under its administration. A company (bidder) wants to get a license for some subset of cities (subset of U) to sell her product stock to the market. Consider the bidder's profit for a subset of cities S to be equal to a unitary *publicly known* product price (e.g., for some products, such as drugs, the government could fix a social price) times the number of product items available in the stocks that the company possesses in the cities comprising S. In this scenario, the bidder is strategic on her stock availability. As noted, e.g., by Celik (2006), a simple inspection on the stock consistency implies that bidders cannot overbid their profits: the concealment of existing items in stock is costless but disclosure of unavailable ones is prohibitively costly. Our assumption is a kind of verification *a posteriori* where the inspection is carried out only for the solution actually implemented; therefore, each bidder cannot overstate her valuation for the set she gets allocated, if any. It is important to notice that bidders can misreport sets and valuations for unassigned sets in an *unrestricted* way. In what follows we will call this model a *weak verification* model.

Priority algorithms comprise a widely accepted general model of greedy-like algorithms in combinatorial optimization (Borodin, Nielsen, & Rackoff, 2003). In the context of CAs and for our purposes, we consider *greedy priority algorithms* that sequentially process *elementary bids*, namely triples consisting of the identity i of a bidder, a subset S of goods, and bidder i's value $v_i(S)$ for S.

A priority algorithm fixes an order on all possible elementary bids, without looking at them, and processes them in this order deciding whether the subsequent elementary bid is accepted or not (Borodin & Lucier, 2016). Greediness implies that the current elementary bid is accepted if the requested set does not conflict with any previously allocated sets. For a *fixed priority algorithm*, the order of the elementary bids is determined before the algorithm starts processing the bids and does not depend on any elementary bids accepted by the algorithm throughout its execution. An algorithm that processes all the bids received from the bidders from the largest to the smaller is an example of a fixed priority algorithm; variants of this approach (where, e.g., the ordering also considers the size of the corresponding set) are studied by Lehmann et al. (2002) and Krysta and Ventre (2015); Algorithm 1 is a further example of a fixed priority algorithm. For *adaptive priority algorithms*, instead, the order of the elementary bids depends on the elementary bids already accepted and may change throughout the execution of the algorithm. An example of such an approach is Algorithm 2. In our model, fixed priority algorithms can accept at most one elementary bid from the same bidder. The reason is that for nonadditive valuation functions (e.g., for submodular or subadditive valuations), accepting an elementary bid from some bidder i may cause the bids of i on other subsets of goods to change. Such changes should affect the actual order in which the algorithm considers the elementary bids, which is not compatible with the idea of a fixed priority algorithm. Adaptive priority algorithms may accept any number of elementary bids from the same bidder.

Extending and generalizing previous results proved by Krysta and Ventre (2015) and Fotakis, Krysta, and Ventre (2014), we first prove that weak verification, namely verification

that does not allow the bidders to overbid on their winning sets, is both sufficient and necessary for any greedy fixed priority algorithm to be truthful without money (i.e., even in absence of payments), if the elementary bids are processed in nonincreasing order of their values, or with money, if the order of the elementary bids is determined by, possibly bidder dependent, increasing functions of their values. We highlight that these results are not specific to only CAs but actually apply to any AMD problem Π and greedy fixed priority algorithm for Π that uses triples as elementary bids (S being an output component pertinent to Π), see Section 3. We remark that such general characterization-type results in AMD are scarce in the literature.

We also look further at the algorithmic characterization of truthful mechanisms with money that use a greedy fixed priority algorithm. In detail, under some assumptions on the ordering functions used to rank the elementary bids, we give an explicit formula for suitable payments that these mechanisms can use. Interestingly, the ordering functions play a role in both the approximation quality of the algorithm and the amount of money used to enforce truthfulness. We then study whether money can be traded off with approximation; in particular, we ask the following question: can we obtain better approximate truthful CAs by investing more in terms of payments? We suggest that the answer to this question is negative by showing that, rather surprisingly, money and (an upper bound to the) approximation guarantee go in the same direction (i.e., higher payments can only lead to worse approximations for CAs) already for single-minded bidders. We also provide an instance where the approximation increases with the payments. To the best of our knowledge, this is one of the first attempts in the literature to relate approximation and frugality of truthful mechanisms. Another recent result in this vein is given in (Serafino, Vidali, & Ventre, 2017) and pertains to mechanisms with monitoring (Kovács, Meyer, & Ventre, 2015), a paradigm closely related to verification.

All the results on the power and the limitations of fixed priority algorithms assume that the algorithm can accept at most one elementary bid from the same bidder. On the other hand, adaptive priority algorithms can change the bid order, again without looking at the actual bids, after they process each elementary bid (Borodin & Lucier, 2016). Moreover, they can accept any number of elementary bids from the same bidder. To deal with the additional power of adaptive priority algorithms (cf. Borodin & Lucier, 2016), we introduce the concept of *strong verification*, where the bidders are not allowed to overbid on *any* subset of their winning set. (We stress that the very same motivation scenario used above for weak verification justifies also this stronger model.) We first characterize all the mechanisms that are truthful without money and with strong verification. We observe that such mechanisms exhibit highly non-monotone behavior, thus suggesting that designing such mechanisms with good approximation guarantee is difficult. This motivates our study of strong verification with money.

As our main technical result, we show that the 2-approximate (adaptive priority) greedy algorithm of Lehmann et al. (2006) admits payments leading to a truthful mechanism in finite bidding domains if and only if strong verification is used. We prove the existence of payments that make this algorithm truthful by using the well known cycle-monotonicity technique (Rochet, 1987; Vohra, 2011), which basically requires to study the cycles of the so-called *declaration graph*. For a fixed agent, this graph contains a node for each possible declaration in the agent's domain; edge from declaration a to declaration b exists when

(strong) verification allows to misreport a as b. The weight of such an edge measures the loss that the agent would make by reporting b when her true type is a. We unveil a surprising structural property of the declaration graph associated to this algorithm. It is well known that if the valuation domain is finite, a mechanism is truthful with money if and only if the corresponding declaration graph does not contain any negative-weight cycle (see, e.g., Rochet, 1987; Vohra, 2011). Here, by carefully exploiting the greedy selection rule and strong verification, we prove that all cycles of the declaration graph consist of cliques of nodes linked by 0-weight edges. Thus, by contracting the nodes of all such cliques, we obtain an acyclic declaration graph. Then, we can define payments for any given declaration (node of this graph) by computing essentially the shortest paths from this node to all the sinks. We also show that strong verification is necessary for the existence of the payments, even if it is allowed to execute all possible orders of goods.

We finally turn our attention to the complexity of computing the payments that are guaranteed to exist by our proof. The only general approach known to date to prove that computing payments is not harder than essentially executing the algorithm (once), is the technique of Babaioff, Kleinberg, and Slivkins (2013, 2015). We show, however, that this approach does not lead to the required payments in our case, indicating that the payment computation has to carefully take into account the structure of the declaration graph. The problem of computing payments can be equivalently formulated as computing a feasible solution to an appropriate linear program (of exponential size) defined on the declaration graph. It is easy to observe that shortest paths are only one of the feasible solutions to this linear program (the minimal one), but any feasible solution can be used as a payment function. In fact, we are not interested in an entire feasible solution, but just on its component that corresponds to (and reveals the payment for) some given node of the declaration graph. We further prove that if the domain is represented succinctly, the problem of computing the payment corresponding to a shortest path solution to this linear program is **NP**-hard. The proof shows that computing such payments requires (and reveals) crucial information about a succinctly represented declaration graph of exponential size. This is true even for CAs with $n = 2$ submodular bidders and $m = 2$ goods.

We complement this result by showing that we can always find efficiently (non-shortest path) payments in our declaration graph in the restricted case with $m = 2$ goods and any number of bidders. We use the structure of the declaration graph to prove that the associated outcome graph (i.e., the declaration graph with all the nodes leading to the same outcomes contracted to a single node) does not have any negative-weight cycle. We are able to actually close the picture completely, by proving that there are instances with $n = 2$ and $m = 3$ for which the outcome graph has a negative weight cycle, meaning that for such instances the payment-per-outcome approach fails.

1.2 Significance of Results and Related Work

Our work is among the very first studies which dig deeper into the mostly unexplored structure of the declaration graphs associated to interesting mechanisms. We are aware of only very few such similar studies (e.g., Lavi & Swamy, 2009; Krysta & Ventre, 2015).

To put our results in a broader mechanism design context, it is in place to compare them with the celebrated VCG mechanism (Vickrey, 1961; Clarke, 1971; Groves, 1973).

Although we do not know if payments for our greedy mechanism for submodular CAs can be computed in time independent of the size of the domain, our results have significant implications. If one wants to use an auction mechanism in practice, the mechanism to choose would arguably be greedy, for its simplicity over VCG, which requires an optimal solution to a computationally difficult problem (VCG does not need verification though). The crucial point is that in practice CAs define in fact the setting in which we *are able* to compute our payments efficiently. Think about submodular CAs where each bidder can declare valuation functions from a set of size polynomial in n and m. In this case, the declaration graph associated to the greedy mechanism has polynomial size and thus payments are computable in time polynomial in n and m, i.e, in the combinatorial size of the allocation problem. For instance, valuations are numbers from $\{0, 1, 2, \ldots, M\}$, where M is a constant and, moreover, each bidder submits explicit bids only on subsets of goods of constant size. Interestingly, for another such practically relevant setting of CAs with constant-size submodular valuations, Dobzinski and Vondrák (2013) prove that the social welfare maximization problem is not only **NP**-hard but also **NP**-hard to approximate within any factor better than 1.225. Thus, we cannot run VCG in this setting in polynomial time. But since the declaration graph of the greedy algorithm has size polynomial in n and m, we can compute the payments in polynomial time, see Theorem 9.

CAs with submodular bidders were widely studied both from an algorithmic perspective and in an AMD framework (see, e.g., Mirrokni, Schapira, & Vondrák, 2008; Khot, Lipton, Markakis, & Mehta, 2005; Vondrák, 2008; Lehmann et al., 2006; Dobzinski, Nisan, & Schapira, 2005; Dobzinski, 2011; Dobzinski & Vondrák, 2012, 2013). As an optimization problem (without strategic consideration), CAs with submodular bidders given by value oracles are NP-hard to solve optimally and **NP**-hard to approximate with factors better than $e/(e-1)$ (Khot et al., 2005); such factors cannot be obtained using only polynomially many value oracles (Mirrokni et al., 2008). On the other hand, there are the following polynomial time algorithms: a deterministic greedy 2-approximation (Lehmann et al., 2006) and a randomized $e/(e-1)$-approximation algorithm (Vondrák, 2008). Now, considering truthful mechanisms for this problem with payments, very strong non-constant lower bounds on the approximation ratios are known. Dobzinski (2011) proved that every universally truthful randomized mechanism for CAs with submodular valuations with an approximation ratio of $m^{1/2-\varepsilon}$, for some $\varepsilon > 0$, must use exponentially many value oracles. There is a deterministic polynomial time truthful mechanism with an approximation ratio of $\sqrt{m}$ (see Dobzinski et al., 2005), which is best possible due to this hardness result. Moreover, Dobzinski and Vondrák (2012) show that no deterministic polynomial time truthful mechanism provides an $m^{1/2-\varepsilon}$-approximation for a constant $\varepsilon > 0$ to this problem, even with succinctly represented submodular valuations, unless $\mathbf{NP} = \mathbf{RP}$. Similarly, a strong non-constant lower bound is proved even for randomized polynomial time truthful in expectation mechanisms for CAs with succinctly represented submodular valuations (Dobzinski & Vondrák, 2012).[2] In light of such strong hardness results, our 2-approximate truthful mechanism is a valuable result (also considering its applicability to practical auction domains).

The concept of ex-post verification (i.e., verification that depends on the output computed) in AMD dates back to the seminal work of Nisan and Ronen (2001); subsequently, it

2. A valuation function is called succinctly represented if it can be encoded in a bit string of length polynomial in the problem size.

has been extensively studied in machine scheduling settings (see Penna & Ventre, 2014 and references therein) and introduced to CAs (Krysta & Ventre, 2015). This work shows how to obtain (essentially) best possible approximation guarantees for CAs with multi-minded bidders, by means of polynomial-time truthful mechanisms using verification and money. We consider their model, dubbed here weak verification, and in Section 3 generalize (some of) their results to characterize verification for truthful greedy CAs. The feasibility of trading verification with money in mechanism design for CAs has been studied in (Fotakis et al., 2014); specifically, the paper shows how weak verification can be leveraged to design truthful CAs that do not charge bidders. We here complement their results by showing in Section 3 that no other verification notion can be used for incentive-compatibility without money as long as greedy algorithms are concerned.

A very recent related work considers value-maximizing bidders (Wilkens, Cavallo, & Niazadeh, 2017). These bidders disregard the payments and only care to maximize their valuation for the computed outcome, as long as their quasi-linear utility is non-negative. On a technical level, their model is pretty similar to mechanisms with verification and without money built upon greedy algorithms. In fact, some of our results and techniques in Section 3 can be reinterpreted to explain some of the results therein.

2. Model and Preliminaries

This section introduces our model and gives some preliminaries.

2.1 CAs and Verification Paradigms

For definition of CAs and basic notation, we refer to Section 1. Assuming that the values $v_i(S)$ are private knowledge of the bidders accessed by value oracles, we seek to design an *allocation algorithm* A and a payment function P. The auction (also called mechanism) (A, P) for a given input of bids from the bidders, outputs an assignment and charges the bidder i a payment P_i. Allocations and payments should be defined so that no bidder has an incentive to misreport her preferences and in order to maximize the social welfare. More formally, we let b_i be a valuation function of agent i, i.e., $b_i : 2^{\mathsf{U}} \to \mathbb{R}^+$. We call b_i a *declaration* of bidder i. We let v_i be the type of agent i and D_i denote the set of all the possible declarations of agent i; we call D_i the *declaration domain* of bidder i. Let $\mathbf{b}_{-i}$ be the declarations of all the agents but i. For any $\mathbf{b}_{-i}$ and declaration $b_i \in D_i$, we let $A_i(b_i, \mathbf{b}_{-i})$ be the set in 2^{U} that A on input $\mathbf{b} = (b_i, \mathbf{b}_{-i})$ allocates to bidder i. If no set is allocated to i then we naturally set $A_i(b_i, \mathbf{b}_{-i}) = \emptyset$. We say that (A, P) is a truthful mechanism (or simply that A is a truthful algorithm) if for any i, $b_i \in D_i$ and $\mathbf{b}_{-i}$:

$$v_i(A_i(v_i, \mathbf{b}_{-i})) - P_i(v_i, \mathbf{b}_{-i}) \geq v_i(A_i(\mathbf{b})) - P_i(\mathbf{b}). \tag{1}$$

In the *(weak) verification model* (Krysta & Ventre, 2015; Fotakis et al., 2014) each bidder can only declare lower valuations for the set she is awarded. More formally, bidder i whose type is v_i can declare a type b_i if and only if:

$$b_i(A_i(b_i, \mathbf{b}_{-i})) \leq v_i(A_i(b_i, \mathbf{b}_{-i})) \tag{2}$$

whenever $A_i(b_i, \mathbf{b}_{-i}) \neq \emptyset$.

The stronger variant of verification we consider here allows each bidder to underbid on the set awarded and its subsets (other declarations are unrestricted). Formally, in the *strong verification model*, bidder i can declare a type b_i if and only if :

$$b_i(T) \leq v_i(T) \; \forall T \subseteq A_i(b_i, \mathbf{b}_{-i}), \tag{3}$$

whenever $A_i(b_i, \mathbf{b}_{-i}) \neq \emptyset$.

When (2) or (3) is not satisfied, the bidder is caught lying by the relevant verification step and the mechanism punishes her in order to make this behavior very undesirable (i.e., for simplicity we can assume that in such a case the bidder will have to pay a fine of infinite value). This way (1) is satisfied directly when the verification does catch lies – i.e., when (2) (respectively, (3)) does not hold for the weak (respectively, strong) verification – as in such a case a lying bidder would have an infinitely bad utility because of the punishment/fine. Thus in this model, truthfulness with weak (strong, respectively) verification of a mechanism is fully captured by (1) holding only for any i, $\mathbf{b}_{-i}$ and $b_i \in D_i$ such that (2) (respectively, (3)) is fulfilled.

When $P_i(\mathbf{b}) = 0$ for all bidders i and bid vectors $\mathbf{b}$, we say that the mechanism is *without money*. All the concepts above can be adapted to the setting without money so that when, for example, (1) is true for null payments we talk about truthful mechanisms without money.

In CAs with submodular bidders, we have that D_i is comprised of submodular valuations, i.e., for all i, $b_i \in D_i$ and $S, T \subseteq \mathsf{U}$, $b_i(S) + b_i(T) \geq b_i(S \cup T) + b_i(S \cap T)$.

2.1.1 A Graph Theoretic Approach to Truthfulness

The technique we use to study truthful mechanisms with verification is the so-called cycle monotonicity. Consider an algorithm A. We set up a weighted graph for each bidder i depending on A, i's domain D_i, verification paradigm, and the declarations $\mathbf{b}_{-i}$. Non-existence of negative-weight cycles (respectively, edges) in this graph guarantees the truthfulness with money (respectively, without money) of A.

More formally, fix algorithm A, bidder i and declarations $\mathbf{b}_{-i}$. Let V denote the verification paradigm at hand; if a type a can declare to be of type b obeying V, we say that $a \rightarrow_V b$ (e.g., $a \rightarrow_V b$ if and only if (2) is true whenever V is weak verification). The *declaration graph* associated to algorithm A has a node for each possible declaration in the domain D_i. For the verification V, we add an edge between a and b in D_i whenever $a \rightarrow_V b$. For example, in the case of strong verification, from (3), edge (a, b) belongs to the graph if and only if $a(T) \geq b(T)$ for all $T \subseteq A_i(b, \mathbf{b}_{-i})$.[3] The weight of the edge (a, b) is defined as $a(A_i(a, \mathbf{b}_{-i})) - a(A_i(b, \mathbf{b}_{-i}))$ and thus encodes the loss that a bidder whose type is a incurs into by declaring b. Here $a(S)$, for a subset of goods S, simply denotes the valuation of S for an agent of type a. The following known result relates the edges/cycles of the declaration graph to the existence of payments leading to truthful auctions.

3. Formally, for strong verification, an edge (a, b) belongs to the graph if $a(T) \geq b(T)$, $\forall T \subseteq A_i(b, \mathbf{b}_{-i})$, only whenever $A_i(b, \mathbf{b}_{-i}) \neq \emptyset$ as this set (and its subsets) would be needed to verify. However, because of the monotonicity and normalization of valuations, $a(A_i(b, \mathbf{b}_{-i})) \geq b(A_i(b, \mathbf{b}_{-i}))$ holds also whenever $A_i(b, \mathbf{b}_{-i}) = \emptyset$, since $a(\emptyset) = b(\emptyset) = 0$.

Proposition 1. *If each D_i is finite and each declaration graph associated to algorithm A does not have negative-weight cycles with verification V then there exists a payment function P such that (A, P) is a truthful mechanism with verification V. Similarly, if each (possibly infinite) graph does not have a negative-weight edge with verification V, then A is a truthful mechanism without money and with verification V.*

The proposition above is adapted from Rochet (1987) and Vohra (2011) to the verification setting V similarly to Ventre's (2014) approach for a specific notion of verification. Note that Proposition 1 requires the technical hypothesis of finite domains for mechanisms with money: this is because the payments P are defined as lengths of certain shortest paths on the declaration graph; shortest paths are well-defined only on finite graphs. This seems to be a limitation of the cycle-monotonicity technique for the design of mechanisms with verification in general (Ventre, 2014).

A variant of the cycle-monotonicity technique described above, defines payments as a function of the outcome and the declarations of the others, using the following tool.

Definition 1 (Ventre, 2014)**.** *For an algorithm A, verification paradigm V, every i and $\mathbf{b}_{-i} \in D_{-i}$, the* outcome graph *has a node for each possible outcome output of A. For every pair of outcomes, X, Y, $X \neq Y$, add an edge (X, Y) if there exist $a, b \in D_i$ such that $A(a, \mathbf{b}_{-i}) = X$, $A(b, \mathbf{b}_{-i}) = Y$ and $a \to_V b$. The weight of edge (X, Y) (if any) is $\inf_{a \in \mathcal{R}_X^Y} \{a(X) - a(Y)\}$, where $\mathcal{R}_X^Y = \{a \in \mathcal{R}_X | \exists b \in \mathcal{R}_Y \text{ s.t. } a \to_V b\}$ with $\mathcal{R}_Z = \{a \in D_i | A(a, \mathbf{b}_{-i}) = Z\}$ for any outcome Z.*

A theorem similar in spirit to Proposition 1 holds and then the absence of negative-weight cycles in the outcome graph is a necessary and sufficient condition for the existence of payments-per-outcome implementing algorithm A. Contrarily to classical mechanism design, in the context of verification, the outcome graph (and the payments-per-outcome implementation) is with loss of generality (Ventre, 2014).[4]

2.2 Greedy Priority Algorithms

To define greedy priority algorithms, we are going to consider a general setting. Here, we let $\mathcal{O}$ denote the set of possible *output components* (e.g., subsets of the universe for CAs); as from above, for $a \in D_i$ and $S \in \mathcal{O}$, we let $a(S)$ denote the valuation of an agent of type a for the output component S (i.e., how much the bidder likes the output component). We are now ready to briefly introduce the class of priority algorithms, following Borodin et al. (2003). The input of a *priority algorithm* is a finite subset I of the class $\mathcal{I}$ of all permissible input atomic items. We consider (elementary) bids as input atomic items. Namely $\mathcal{I}$ consists of all possible triples $(i, S, b_i(S))$, where i is the bidder, S is an output component (e.g., subset of U for CAs) of the problem at hand, and $b_i(S)$ is i's valuation for S. For simplicity, we denote such an elementary bid as $b_i(S)$, or $b(S)$, if i is clear from the context. The output of a priority algorithm consists of an irrevocable decision for each elementary bid processed. A (possibly adaptive) priority algorithm A receives as input a finite set of elementary bids I and proceeds in rounds, processing a single bid in each round. While there are unprocessed bids in I, A selects a total order $\mathcal{T}$ on $\mathcal{I}$ without looking at the set of

4. Theorem 8 below reinforces this observation for stronger notions of verification.

unprocessed bids. It is important that $\mathcal{T}$ can be any total order on $\mathcal{I}$, and that for adaptive priority algorithms, the order may be different in each round. If the order is fixed before starting processing the bids, then the algorithm is called fixed priority. In each round, A receives the first (according to $\mathcal{T}$) unprocessed bid $x \in I$ and irrevocably accepts or rejects it. Then, x (and all bids preceding x in $\mathcal{T}$) are removed from I, and A proceeds to the next round. *Greedy priority algorithms* accept the current bid $b_i(S)$ if granting S to bidder i is feasible for the problem at hand (e.g., S is disjoint to the previously allocated subsets of U).

E.g., the family of greedy algorithms studied by Lehmann et al. (2002) and Krysta and Ventre (2015) belong to the class of greedy fixed priority algorithms. In these algorithms, elementary bids are sorted according to some criterion, which is non-decreasing in bid $b_i(S)$ and non-increasing in the cardinality of the set of goods S. Typical examples of such criteria are the bid $b_i(S)$, the average bid per good $b_i(S)/|S|$ and $b_i(S)/\sqrt{|S|}$. Then, the algorithm considers the elementary bids in instance I in this order, accepts the next elementary bid $(i, b_i(S), S)$ (i.e., irrevocably allocates the set of goods S to bidder i), removes from I all other bids from bidder i and all bids for sets S' with nonempty intersection with S, and proceeds with the next available bid. A typical example of a greedy adaptive priority algorithm is Algorithm 2 (see the discussion in Section 4.2).

Throughout this work, we assume that a fixed priority algorithm can accept at most one elementary bid from the same bidder. We note that for nonadditive valuation functions (e.g., for submodular or subadditive valuations), accepting a bid $(i, S, b_i(S))$ may lead to updated bids $(i, S', b_i(S'))$ from bidder i on other subsets S' of goods. This, in turn, may change the order in which the priority algorithm considers the remaining elementary bids. Since any change in the order of elementary bids is not compatible with the idea of fixed priority algorithms, the assumption that a fixed priority algorithm can accept at most one bid from each bidder is essentially without loss of generality. On the other hand, adaptive priority algorithms may accept any number of elementary bids from the same bidder.

3. Weak Verification and Greedy Fixed Priority Algorithms

We want to understand what kind of verification we can use with greedy fixed priority algorithms to implement them truthfully (with or without money). It turns out that the concept of weak verification (2) is sufficient and necessary for that. The role of money in truthfulness is strictly linked with the ordering of the bids used by the algorithm.

To establish these results, we are going to consider a general mechanism design setting. Note that, as done above for D_i when defining greedy priority algorithms, we can import all the notation defined above for CAs. Thus, for example, $A_i(\mathbf{b})$ denotes the output component that bidder i gets allocated by the algorithm A on input bid vector $\mathbf{b}$. We call $A_i(\mathbf{b})$ the *winning component* of bidder i.

The proofs in this section (and the proofs of Theorems 4 and 5 in the next section) are build upon cycle monotonicity in the declaration graph, and share the same underlying structure. By the verification notion considered, the existence of an edge in the declaration graph gives an extra algorithmic property that coupled with greediness leads to the desired results.

Remark 1. *Theorems 1 and 2 below hold for any AMD problem* Π *and greedy priority algorithm for* Π *that can be cast in our framework (i.e., input items are elementary bids). Recall that a greedy priority algorithm treats each input item as if it was the last one in the sequence and then optimizes the objective function of the algorithm on each item (Borodin et al., 2003).*

3.1 Mechanisms without Money

Our first result concerns truthfulness without money. In this context, we use the term algorithm and mechanism interchangeably.

Theorem 1. *A greedy fixed priority algorithm processing the bids in non-increasing order of their value (and with a fixed bid-independent tie-breaking rule) is truthful with verification and without money if and only if the verification prevents overbidding on the (non-empty) winning component.*

Proof. Fix i and $\mathbf{b}_{-i}$ and consider the declaration graph associated to the greedy priority algorithm A with a verification paradigm V. By Proposition 1, A is truthful without money if and only if the graph has no negative-weight edges. We prove that if V is weak verification (2), then there is no negative-weight edge, and conversely, if V is not weak verification (2), then there exist instances with a negative edge.

For the first claim, let (a, b) be an edge of the graph. By hypothesis, this edge exists if and only if $a(Y) \geq b(Y)$, $Y = A_i(b, \mathbf{b}_{-i})$. The edge weight is $a(A_i(a, \mathbf{b}_{-i})) - a(Y)$. Consider the case in which the edge is negative, i.e., $a(A_i(a, \mathbf{b}_{-i})) < a(Y)$ (thus yielding $Y \neq A_i(a, \mathbf{b}_{-i})$). In such a case, due to the ordering of the bids, $a(Y)$ is considered by A before $a(A_i(a, \mathbf{b}_{-i}))$. But since A is greedy and since Y was not allocated to i, there must exist bids $b_j(Z)$, for $j \neq i$, and some other output components Z, conflicting with Y, such that $b_j(Z) \geq a(Y)$. Let $b^*_j(Z)$ be the highest of these bids. But then, since $Y = A_i(b, \mathbf{b}_{-i})$, we have $b(Y) \geq b^*_j(Z) \geq a(Y)$, with at least one of the inequalities being strict (by the fixed tie-breaking rule). Thus, a contradiction.

For the second claim, since V is not (2), some bidder i is allowed to overbid on some winning component Y. We can, therefore, build an instance in which bidder i by declaring a does not win Y, with $a(A_i(a, \mathbf{b}_{-i})) < a(Y)$, while, she wins Y by declaring b. The construction amounts to define $\mathbf{b}_{-i}$ in a way similar to the above (i.e., bid $b^*_j(Z)$ in between $b(Y)$ and $a(Y)$). But then (a, b) would have negative weight. □

3.2 Mechanisms with Money

Let $f_i : D_i \times \mathcal{O} \to \mathbb{R}_{\geq 0}$ [5] be a function that maps input declarations from the domain of agent i and an output component to a non-negative real number. Assume that for fixed outcome, f_i is strictly increasing in the valuation, i.e., for any $S \in \mathcal{O}$, $a, b \in D_i$, $a(S) > b(S)$ if and only if $f_i(a, S) > f_i(b, S)$.

Consider now the greedy fixed priority algorithm that uses these functions f_i to rank the bid b_i of bidder i for the various components $S \in \mathcal{O}$. Examples of algorithms belonging

5. We remark that Theorem 2 works also when f_i has co-domain $\mathbb{R}$; the non-negativity assumption is only needed for the results in Section 3.2.1.

to this family, where $f_i(a,S) = \frac{a(S)}{\sqrt{|S|}}$, are studied in the literature for single-parameter bidders (Lehmann et al., 2002) and multi-parameter agents (Krysta & Ventre, 2015). Both these algorithms (essentially) return the best possible approximation guarantees for CAs, even in a non-strategic computational sense. Such a generalization of greedy algorithms, however, can add negative-weight edges (cf. Krysta & Ventre, 2015, Example 2) but no negative-weight cycle, as shown next. Below, we say that a greedy (fixed priority) algorithm A is truthful with money (and verification) if there exist payments P such that (A,P) is a truthful mechanism (with verification).

Theorem 2. *A greedy fixed priority algorithm that orders elementary bids using functions f_i as above (and breaks ties in a fixed manner independent on the bids) is truthful with verification and with money if and only if the verification prevents bidder i from overbidding on the (non-empty) winning component.*

Proof. For the "if" direction, fix i and $\mathbf{b}_{-i}$ and consider the declaration graph associated to the greedy priority algorithm A. (Accordingly, $\mathbf{b}_{-i}$ is dropped from the notation herein to increase readability.) We show that the cycles of this graph have non-negative weights. Consider a generic cycle $C := a^0 \to a^1 \to \ldots \to a^k \to a^{k+1} = a^0$.

Claim 1. *If for some $j \in \{0,\ldots,k\}$, we have $A_i(a^j) = \emptyset$ then the cycle weights 0.*

Proof. Assume that for some $a = a^j$ in C, $A_i(a) = \emptyset$, meaning that the winning component of bidder i declaring a is empty. The claim is that for $b = a^{j+1}$, $A_i(b) = \emptyset$, thus implying, by reiterating the argument, that all the cycle has weight 0. Assume by sake of contradiction that $A_i(b) \neq \emptyset$ and note that since $A_i(a) = \emptyset$ then, for all non-empty output components X, there must exist bidder $j_X \neq i$ such that $Y_X = A_{j_X}(b_{j_X}, \mathbf{b}_{-j_X})$ conflicts with X and $f_{j_X}(b_{j_X}, Y_X) \geq f_i(a,X)$. But then since $A_i(b) \neq \emptyset$ and as the edge (a,b) belongs to the graph, we have by the definition of verification, that $a(A_i(b)) \geq b(A_i(b))$ and then $f_i(b, A_i(b)) \leq f_i(a, A_i(b))$. However, since $Z = A_i(b)$ is the winning component of i when declaring b it must be that $f_i(b,Z) \geq f_{j_Z}(b_{j_Z}, A_{j_Z}(b_{j_Z}, \mathbf{b}_{-j_Z})) \geq f_i(a,Z)$. Therefore, $f_i(b, A_i(b)) = f_i(a, A_i(b))$ which in turns yields, by the bid-independent fixed tie-breaking rule, that $A_i(a) = A_i(b) \neq \emptyset$, a contradiction. □

We can then focus on the case in which $A_i(a^j) \neq \emptyset$ for all $j \in \{0,\ldots,k\}$. By the existence of edge (a^j, a^{j+1}) we obtain $a^j(A_i(a^{j+1})) \geq a^{j+1}(A_i(a^{j+1}))$ and then, by monotonicity of f_i,

$$f_i(a^j, A_i(a^{j+1})) \geq f_i(a^{j+1}, A_i(a^{j+1})). \tag{4}$$

We now prove that, for $0 \leq j \leq k$, we have

$$f_i(a^j, A_i(a^j)) \geq f_i(a^j, A_i(a^{j+1})). \tag{5}$$

Assume by contradiction that this is not the case; by the definition of priority algorithms, this means that it is not feasible to allocate $A_i(a^{j+1})$ to bidder i when she bids a^j meaning that there must exist some other bidder $k \neq i$ whose output component $A_k(a^j)$ conflicts with $A_i(a^{j+1})$ and is considered before in the ordering of the algorithm. On the other hand, it is feasible to allocate $A_i(a^{j+1})$ to bidder i when she bids a^{j+1}, meaning that $A_i(a^{j+1})$

must be considered before (i.e., have a higher priority than) $A_k(a^j)$ when bidder i bids a^{j+1}. We can then conclude that $f_i(a^j, A_i(a^{j+1})) < f_i(a^{j+1}, A_i(a^{j+1}))$, a contradiction with (4).

Unfolding the two inequalities above for the cycle, we get

$$f_i(a^j, A_i(a^{j+1})) = f_i(a^{j+1}, A_i(a^{j+1})),$$

which implies that $a^j(A_i(a^{j+1})) = a^{j+1}(A_i(a^{j+1}))$. Then the weight of the cycle C is

$$\sum_{j=1}^{k} a^j(A_i(a^j)) - a^j(A_i(a^{j+1})) = \sum_{j=1}^{k} a^j(A_i(a^j)) - a^{j+1}(A_i(a^{j+1})) = 0.$$

This concludes the "if" part of the proof by Proposition 1.

For the "only if" part, let us assume by contradiction that a fixed priority algorithm from the statement is truthful, yet the verification allows some bidder i to overbid on some non-empty winning component X. We will prove that there exist f_i's for which such an algorithm cannot be truthful. Specifically, we are going to focus on $f_i(a, X) = f_i(a(X))$, i.e., f_i depends only on $a(X)$ rather than the whole tuple $(a(S))_{S\in\mathcal{O}}$ and $X \in \mathcal{O}$, and f_i that is unbounded from above.

Consider the instance where X is the most preferred component of i and assume that X is the only component maximizing i's utility; call a_i the corresponding type. Define $\mathbf{b}_{-i}$ in such a way that X is not allocated to i when she is truthtelling. Since A is greedy, there exist bidders $j \neq i$, such that $f_j(b_j(Y)) > f_i(a_i(X))$, $Y = A_j(b_j, \mathbf{b}_{-j})$ and Y conflicting with X. Let j be such a bidder with the highest $f_j(b_j(Y))$. We are going to show a 2-cycle of negative weight thus showing that there are no payments for the algorithm (cf. Proposition 1). Consider the bid b defined as a_i except that

(i) $b(X) > a_i(X)$ in such a way that $f_i(b(X)) > f_j(b_j(Y))$; and

(ii) $b(A_i(a_i, \mathbf{b}_{-i})) > b(X) - a(X) + a_i(A_i(a_i, \mathbf{b}_{-i}))$.

Since f_i is strictly increasing and unbounded from above such a b exists. Note that by (i) and (ii), X is allocated to i in this case (in particular, since $a_i(A_i(a_i, \mathbf{b}_{-i})) - a_i(X) < 0$ then $b(A_i(a_i, \mathbf{b}_{-i})) < b(X)$). Moreover, since by contradiction the verification allows this lie, then the declaration graph has the edge (a_i, b). But the edge (b, a_i) belongs to the graph as well as there is no overbidding on winning component (i.e., by (i) and (ii) $b(A_i(a, \mathbf{b}_{-i})) - a_i(A_i(a, \mathbf{b}_{-i})) > b(X) + a(X) > 0$). The cycle $a_i \to b \to a_i$ has then weight $b(X) - b(A_i(a_i, \mathbf{b}_{-i})) + a_i(A_i(a_i, \mathbf{b}_{-i})) - a(X) < 0$ by (ii). □

3.2.1 Explicit Payment Functions

Let V be the verification that prevents bidder i from overbidding on the (non-empty) winning component. We now explore the possibility to define explicit payment functions, mainly based upon (5), leading to truthfulness with verification V.

The first observation is that truthful CAs using greedy algorithms as those considered in Theorem 2 might need to pay agents rather than charging them (Krysta & Ventre, 2015) for otherwise bidders would be better off by not participating to the auction (i.e., the auction would not satisfy voluntary participation). Therefore, to simplify the notation and

be consistent with our model above, in this section we define $p_i(\mathbf{b}) = -P_i(\mathbf{b})$ for all agents i and bid vectors $\mathbf{b}$; this way the utility of agent i becomes $p_i(b_i, \mathbf{b}_{-i}) + v_i(S)$, S denoting the set that the auction allocates to i in input $\mathbf{b}$.

The second observation is that by the arguments in the proof of Theorem 2 we know that for $t \rightarrow_V b$, it must be either the case that $A_i(t, \mathbf{b}_{-i}) = A_i(b, \mathbf{b}_{-i}) = \emptyset$ or $A_i(t, \mathbf{b}_{-i}) \neq \emptyset$. This immediately implies that when the mechanism satisfies voluntary participation we can achieve truthfulness by simply setting $p_i(b, \mathbf{b}_{-i}) = 0$ whenever $A_i(b, \mathbf{b}_{-i}) = \emptyset$. The only remaining case is then the case $A_i(t, \mathbf{b}_{-i}) \neq \emptyset$ and $A_i(b, \mathbf{b}_{-i}) \neq \emptyset$. Our objective is to define payment functions $p_i(t, \mathbf{b}_{-i})$ and $p_i(b, \mathbf{b}_{-i})$ such that when agent can misreport t as b according to V, the following holds true for some constant $c \geq 0$:[6]

$$\begin{aligned} p_i(t, \mathbf{b}_{-i}) + t(A_i(t, \mathbf{b}_{-i})) &\geq c f_i(t, A_i(t, \mathbf{b}_{-i})) && (6) \\ &\geq c f_i(t, A_i(b, \mathbf{b}_{-i})) \\ &\geq p_i(b, \mathbf{b}_{-i}) + t(A_i(b, \mathbf{b}_{-i})). && (7) \end{aligned}$$

By applying the same reasoning to all pairs of declarations, by (6) and (7) we get the following system of inequalities: for all b in the domain with $A_i(b, \mathbf{b}_{-i}) \neq \emptyset$:

$$\begin{aligned} \inf_{t \rightarrow_V b} \{c f_i(t, A_i(b, \mathbf{b}_{-i})) - t(A_i(b, \mathbf{b}_{-i}))\} &\geq p_i(b, \mathbf{b}_{-i}) \\ p_i(b, \mathbf{b}_{-i}) &\geq c f_i(b, A_i(b, \mathbf{b}_{-i})) - b(A_i(b, \mathbf{b}_{-i})). \end{aligned}$$

The system admits solutions if and only if for all $t \rightarrow_V b$, the following holds

$$c f_i(t, A_i(b, \mathbf{b}_{-i})) - t(A_i(b, \mathbf{b}_{-i})) \geq c f_i(b, A_i(b, \mathbf{b}_{-i})) - b(A_i(b, \mathbf{b}_{-i})). \quad (8)$$

When $A_i(b, \mathbf{b}_{-i}) \neq \emptyset$, define

$$p_i(b, \mathbf{b}_{-i}) = c f_i(b, A_i(b, \mathbf{b}_{-i})) - b(A_i(b, \mathbf{b}_{-i})).$$

In general, there is no guarantee on the sign of these payments.

Theorem 3. *Let A be a greedy fixed priority algorithm that orders elementary bids using strictly increasing functions f_i of their value (and breaks ties in a fixed manner independent on the bids). If (8) holds then the mechanism (A, p) is truthful with verification V and runs in polynomial-time if A does.*

Proof. Fix i and $\mathbf{b}_{-i}$; to simplify the exposition we drop i and $\mathbf{b}_{-i}$ from the notation in the sequel of the proof. Let t be i's true type and take any $b \neq t$ in the domain of player i such that the agent can misreport t as b according to V.

Let us first consider the case in which $A_i(t) = \emptyset$. As noted in the proof of Theorem 2, it then must be the case that $A_i(b) = \emptyset$ for otherwise (t, b) would not belong to the graph. In this case truthfulness follows from having the utility from truthtelling equal 0 which is also the utility from declaring b since the payment is 0 for both declarations.

6. The constant c might be needed to make the system feasible; for some f_i, e.g., $f_i(a, S) = \frac{a(S)}{\sqrt{|S|}}$, this is in fact needed (Krysta & Ventre, 2015).

Next, consider the case in which $A_i(t) \neq \emptyset$ and $A_i(b) = \emptyset$. Since $t(\emptyset) = 0$ then the utility of agent i from declaring b is 0. The utility that agent i experiences from being truthful is

$$\begin{aligned} t(A_i(t)) + p_i(t) &= t(A_i(t)) + cf_i(t, A_i(t)) - t(A_i(t)) \\ &= cf_i(t, A_i(t)) \geq 0, \end{aligned}$$

and therefore we have that the mechanism is truthful (and satisfies voluntary participation).

Finally, consider the case in which $A_i(t) \neq \emptyset$ and $A_i(b) \neq \emptyset$. The utility of agent i when declaring b is then

$$\begin{aligned} t(A_i(b)) + p_i(b) &= t(A_i(b)) + cf_i(b, A_i(b)) - b(A_i(b)) \\ &\leq cf_i(t, A_i(b)) \\ &\leq cf_i(t, A_i(t)) = t(A_i(t)) + p_i(t), \end{aligned}$$

where the first inequality follows from (8) and the second by (5). □

3.2.2 Trade-Off between Frugality and Approximation

Theorems 2 and 3 characterize a family of truthful mechanisms in terms of both algorithms and payments. The question we want to address here is to what extent we can trade the amount of money paid to the agents (also known as *frugality*) with the approximation guarantee of the truthful mechanism.

Let us define a couple of useful notions. Assume that for f_i there exists a constant c such that

$$cf_i(b, A_i(b, \mathbf{b}_{-i})) \geq b(A_i(b, \mathbf{b}_{-i}))$$

for all $i, t \rightarrow_V b$. We call such functions f_i *c-positive*. Notice, that for the payments of Theorem 3 we have $p_i(b, \mathbf{b}_{-i}) \geq 0$ for c-positive function f_i. The *steepness* of function f_i is simply $\max_{b,S\neq\emptyset} \frac{f_i(b,S)}{b(S)}$. Observe that if f_i is c-positive then its steepness is at least $1/c$. From the definition of payments in Theorem 3 it follows that if the charged payments are higher then functions f_i can only be steeper. A very natural question now is the following: if our mechanism charges more, which may lead to steeper functions f_i, does this improve its approximation guarantee? We will show that the answer to this question is negative even in a very simple setting of CAs with single-minded bidders. We will first prove an interesting connection between steepness of functions f_i and therefore the payments and a bound on the approximation guarantee of the mechanism. This shows that if the payments, i.e., steepness, decrease, then this bound on the approximation ratio also decreases, see Proposition 2. Then we will use this proposition and its proof to show in Proposition 3 that indeed it is possible that the payments increase and at the same time, the approximation increases, that is, gets strictly worse.

Suppose that each bidder $i = 1, 2, \ldots, n$ is single-minded, that is, there exists a set of goods $S_i \subseteq \mathsf{U}$ that bidder i demands and a number $w_i > 0$ such that: $v_i(S) = w_i$ for any $S \subseteq \mathsf{U}$ with $S_i \subseteq S$, and $v_i(S) = 0$ otherwise. Assume also for simplicity that the demanded sets have sizes at most a positive integer d, that is, $d = \max\{|S_1|, \ldots, |S_n|\}$.

We will use a primal-dual analysis of Krysta and Ventre (2015). Observe first that Algorithm 1 (Krysta & Ventre, 2015) is a greedy fixed priority algorithm that orders elementary

Algorithm 1: The greedy fixed priority algorithm for single-minded CAs.

1 Let $b_1, b_2, \ldots, b_n$ be the bids and $S_1, \ldots, S_n$ be the sets ordered by non-increasing f_i-values, i.e., $f_1(b_1, S_1) \geq f_2(b_2, S_2) \geq \cdots \geq f_n(b_n, S_n)$. In case of ties between different bidders consider first the bid of the lexicographically bigger bidder.
2 $\mathcal{P} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$.
3 For $i = 1, \ldots, n$ do
4 If $i \notin \mathcal{B} \wedge S_i \cap S = \emptyset$ for all $S \in \mathcal{P}$ then (a) $\mathcal{P} \leftarrow \mathcal{P} \cup \{S_i\}$, (b) $\mathcal{B} \leftarrow \mathcal{B} \cup \{i\}$.
5 Return $\mathcal{P}$.

bids using strictly increasing c-positive functions f_i, where each $f_i(b, S) = \frac{b(S)}{\sqrt{|S|}}$ and $c = \sqrt{d}$. Our proof will closely follow the proof of Lemma 10 from Krysta and Ventre (2015) (in fact the proof of Claim 1 from Krysta & Ventre, 2015). We will prove that any greedy fixed priority algorithm A that orders elementary bids using functions f_i has an approximation ratio of at most $d \cdot c \cdot \tau$, where τ is an upper bound on the steepness of all functions f_i. Let us denote by $\mathcal{S} = \{S_1, \ldots, S_n\} \subseteq 2^U$ the family of the demanded sets. For a given set $S_i \in \mathcal{S}$ we will denote by $b_i = b_i(S_i)$ the bid of bidder i for that set. For the purpose of our analysis, we formally describe algorithm A in Algorithm 1. Let us assume for simplicity that the indices of sets in $\mathcal{S} = \{S_1, \ldots, S_n\}$ coincide with the order of the sets in algorithm A, that is, $f_1(b_1, S_1) \geq f_2(b_2, S_2) \geq \cdots \geq f_n(b_n, S_n)$.

Proposition 2. *Consider the social welfare maximization problem in single-minded CAs with bundles of size at most d. Let A be a greedy fixed priority algorithm for this problem that orders elementary bids using strictly increasing c-positive functions f_i of their value (and breaks ties in a fixed manner that is independent on the bids). Assume that τ is the largest value of the steepness of all the functions f_i, $i = 1, \ldots, n$. Then algorithm A has an approximation ratio of at most $d \cdot c \cdot \tau$.*

Before we proceed with the proof, let us observe that if the bidders' payments decrease, the steepness, as well as the (upper bound on the) approximation may only decrease.

Our proof will employ the classical dual-fitting approach, see, e.g., Chapter 13 in the book by Vazirani (2001). Chapter 13 uses the dual-fitting approach to analyse an approximation ratio of the greedy algorithm for a minimization vertex cover problem. This approach uses linear programming (LP) relaxation of the primal problem together with its dual linear programming problem. In our case the primal problem is a maximization problem and thus its dual is a minimization problem. The linear programming (LP) relaxation of our primal problem is as follows:

$$\begin{array}{lll} \max & \sum_{i=1}^{n} b_i x_i & \\ \text{s.t.} & \sum_{S_i : S_i \in \mathcal{S},\, e \in S_i} x_i \leq 1 & \forall e \in \mathsf{U} \\ & x_i \geq 0 & \forall S_i \in \mathcal{S}. \end{array} \tag{9}$$

The corresponding dual linear program is then the following:

$$\begin{array}{lll} \min & \sum_{e \in \mathsf{U}} y_e & \\ \text{s.t.} & \sum_{e \in S_i} y_e \geq b_i & \forall S_i \in \mathcal{S} \\ & y_e \geq 0 & \forall e \in \mathsf{U}. \end{array} \tag{10}$$

The high level outline of the dual-fitting approach is a follows. As the greedy algorithm proceeds and gradually builds an integral primal solution x, we appropriately define a corresponding fractional dual solution y which however might be infeasible for the dual LP. To make it feasible, we will appropriately scale it to $d \cdot c \cdot \tau \cdot y$ (see the proof below), which is the dual-fitting ingredient of the approach. Because the scaled dual solution is feasible for the dual LP, by weak LP duality its dual objective value provides an upper bound on the value of any feasible primal solution. This fact is used to prove a bound on the approximation ratio.

Proof of Proposition 2. Suppose that Algorithm 1 has terminated and output solution $\mathcal{P}$, that is, we define the following integral primal solution: $x_i = 1$ if $S_i \in \mathcal{P}$ and $x_i = 0$ otherwise. Let $SAT_{\mathcal{P}} = \cup_{S \in \mathcal{P}} S$. Notice that for each set $S \in \mathcal{S}$ that was not chosen in the final solution $\mathcal{P}$, there is an element $e \in SAT_{\mathcal{P}} \cap S$ which was the *witness* of that event during the execution of the algorithm.

For each set $S \in \mathcal{S} \setminus \mathcal{P}$ we keep in $SAT_{\mathcal{P}}$ one (arbitrary) witness for S. We discard the remaining elements from $SAT_{\mathcal{P}}$.

A fractional dual solution y to be used in the analysis is defined during the execution of Algorithm 1. In line 2 of Algorithm 1 we set $y_e := 0$ for all $e \in \mathsf{U}$. We also add the following in line 4(a): $y_e := \frac{b_i}{|S_i|}$, for all $e \in S_i$. Whenever for some $e \in \mathsf{U}$, the value of y_e has not been defined, it assumes value zero. It is now an easy observation that the dual solution provides a lower bound on the cost of the output solution, that is,

$$\sum_{e \in \mathsf{U}} y_e \leq \sum_{S_i \in \mathcal{P}} b_i. \tag{11}$$

We now turn our attention to showing that the scaled dual solution $d \cdot c \cdot \tau \cdot y$ is feasible for the dual linear program, that is, constraints (10) are fulfilled for all sets $S \in \mathcal{S}$. Towards this goal, we have to show that, for each set $S_i \in \mathcal{S}$,

$$d \cdot c \cdot \tau \cdot \sum_{e \in S_i} y_e \geq b_i. \tag{12}$$

Suppose first that $S_r \in \mathcal{S} \setminus \mathcal{P}$. The reason that set S_r has not been included in the solution $\mathcal{P}$ is that there must be an element $e \in SAT_{\mathcal{P}}$ such that $e \in S_r$. In that case adding set S_r to solution $\mathcal{P}$ would violate constraint (9). Let $S_j \in \mathcal{P}$ be the set in the solution that contains element e. Recall that $e \in S_r \cap S_j$, thus

$$\sum_{e' \in S_r} y_{e'} \geq y_e = \frac{b_j}{|S_j|} = \frac{b_j}{|S_j| f_j(b_j, S_j)} \cdot f_j(b_j, S_j) \geq$$

$$\frac{b_j}{|S_j| f_j(b_j, S_j)} \cdot f_r(b_r, S_r) \geq \frac{1}{|S_j| \frac{f_j(b_j, S_j)}{b_j}} \cdot b_r \cdot \frac{1}{c} \geq \frac{b_r}{d \cdot c \cdot \tau},$$

where the second inequality follows by the greedy selection rule, and the third inequality uses the c-positive property.

Notice that claim (12) follows immediately from the definition of y if set $S_r \in \mathcal{P}$ was chosen by the algorithm. In this case we have the stronger inequality:

$$\sum_{e \in S_r} y_e \geq b_r.$$

This proves (12).

We have shown that the dual solution $d \cdot c \cdot \tau \cdot y$ is feasible for the dual linear program, which by weak LP duality implies that $d \cdot c \cdot \tau \cdot \sum_{e \in \mathsf{U}} y_e$ is an upper bound on the value of the optimal integral solution to our problem. We have also shown in (11), that $\sum_{e \in \mathsf{U}} y_e \leq \sum_{S_i \in \mathcal{P}} b_i$. Therefore, we obtain that

$$opt \leq d \cdot c \cdot \tau \cdot \sum_{e \in \mathsf{U}} y_e \leq d \cdot c \cdot \tau \cdot \sum_{S_i \in \mathcal{P}} b_i,$$

which proves the proposition. □

Inspired by the above proposition and its proof we will now show that indeed there exists an instance of CA where if the payments to the bidders are higher, the approximation ratio of the greedy mechanism gets provably worse. Recall here that functions f_i are called *c-positive* if there exists a constant $c > 0$ such that $cf_i(b, A_i(b, \mathbf{b}_{-i})) \geq b(A_i(b, \mathbf{b}_{-i}))$ for all $i, t \rightarrow_V b$.

Proposition 3. *There exists an instance of the social welfare maximization problem in CA with three single-minded bidders $N = \{1, 2, 3\}$ and with 2 goods, and two collections of c-positive functions f_i^1, f_i^2, $i \in N$, such that a greedy fixed priority algorithm for this problem that orders elementary bids using functions f_i^1 on this instance outputs an optimal solution with payments zero, and when using f_i^2 on this instance, it only outputs a $\sqrt{2}$-approximate solution with strictly positive payments.*

Proof. The instance of CA has two goods $\mathsf{U} = \{1, 2\}$ and the three bidders are: $S_1 = \{1\}$, $v_1 = 1 - \epsilon$, $S_2 = \{2\}$, $v_2 = \epsilon$, $S_3 = \{1, 2\}$, $v_3 = \sqrt{2}$, where $\epsilon > 0$ is a very small constant. Now we define the functions f: $f_1^1(v) = v$, $f_2^1(v) = v$, $f_3^1(v) = v/\sqrt{2}$, $f_1^2(v) = \delta \cdot v$, $f_2^2(v) = v$, $f_3^2(v) = v/\sqrt{2}$, where δ is some constant such that $\delta > 1/(1 - \epsilon)$. Observe that in the two collections of functions only the function f_1^1 for the first bidder has changed to f_1^2, otherwise $f_i^1 = f_i^2$ for $i = 2, 3$.

Because $f_3^1(v_3) = \sqrt{2}/\sqrt{2} > f_1^1(v_1) = 1 - \epsilon > f_2^1(v_2) = \epsilon$, if we run the greedy fixed priority algorithm on this instance with functions f_i^1, the solution output has only set S_3 and its social welfare is $v_3 = \sqrt{2}$ and it is optimal in this instance. Thus the approximation ratio of the algorithm is 1 and observe that the payment of bidder 3 is $cf_3^1(v_3) - v_3 = 0$, because we can just take $c = \sqrt{2}$ (payments of the other two bidders are zero anyway).

On the other hand, because $f_1^2(v_1) = \delta \cdot v_1 > 1 = f_3^2(v_3) > f_2^2(v_2) = \epsilon$, if we run the greedy fixed priority algorithm on this instance with functions f_i^2, the solution output has sets S_1, S_2 and its social welfare is $v_1 + v_2 = 1$, but the optimal solution is as before S_3 and has social welfare $\sqrt{2}$. Thus the approximation ratio of the algorithm is only $\sqrt{2}$.

Now, to see what the payment is in this case we need to find the constant c as the largest ratio $v_i/f_i^2(v_i)$ for $i = 1, 2$ (payment of bidder 3 is zero), see the definition of c-positive.

We have that $v_1/f_1^2(v_1) = \frac{1-\epsilon}{\delta(1-\epsilon)} = 1/\delta < 1$ and $v_2/f_2^2(v_2) = \frac{\epsilon}{\epsilon} = 1$, which gives $c = 1$ (because $1/\delta < 1$). Therefore, the payments are: $p_1 = cf_1^2(v_1) - v_1 = c\delta(1-\epsilon) - (1-\epsilon)$, and $p_2 = cf_2^2(v_2) - v_2 = c\epsilon - \epsilon = 0$. Observe that the payment p_1 of bidder 1 is strictly positive because $\delta > 1/(1-\epsilon) > 1$. □

4. Strong Verification

For adaptive priority algorithms, things can be more complex at least in the multidimensional case. Since bidders control more than one bid, they could lie on one of their bids to change the ordering of the adaptive priority algorithm and get an advantage. Therefore, it is not enough in general to prevent lies on the winning component, but a more stringent notion ought to be used. Next, we show that this is indeed the case for an adaptive priority algorithm for CAs with submodular bidders given by Lehmann et al. (2006). The concept of verification used to implement this algorithm turns out to be the strong verification introduced above.

4.1 Truthfulness without Money

We begin by characterizing the algorithms that are truthful without money in the setting of strong verification. Interestingly, the characterizing property is algorithmic only.

Definition 2. *An algorithm A is* strongly monotone *if the following holds for any i, any $\mathbf{b}_{-i}$, any $a \in D_i$: if $A_i(a, \mathbf{b}_{-i}) = S$ then for all $b \in D_i$ such that $b(T) \geq a(T)\ \forall T \subseteq S$ it holds $b(A_i(b, \mathbf{b}_{-i})) \geq b(S)$.*

Theorem 4. *An algorithm A is truthful without money and with strong verification for bidders given by value oracles if and only if A is strongly monotone.*

Proof. Fix i, $\mathbf{b}_{-i}$ and consider the declaration graph associated to algorithm A. Take any edge of the graph (b, a) and let S denote $A_i(a, \mathbf{b}_{-i})$. By definition, the edge exists if and only if $b(T) \geq a(T)$ for all $T \subseteq S$.

Now if the algorithm is strongly monotone, we also have that $b(A_i(b, \mathbf{b}_{-i})) \geq b(S)$ and then the weight of the edge (b, a), that is, $b(A_i(b, \mathbf{b}_{-i})) - b(S)$, is non-negative. Vice versa, assume that the weight of (b, a) is non-negative: this means that whenever $b(T) \geq a(T)\ \forall T \subseteq S$ then it must be $b(A_i(b, \mathbf{b}_{-i})) \geq b(S)$ and therefore A is strongly monotone. The theorem follows from Proposition 1. □

This characterization relates to those given by Fotakis et al. (2014) for moneyless mechanisms with weak verification and mechanisms with money and no verification (Mu'Alem & Nisan, 2008; Lehmann et al., 2002). However, while in these cases the mechanisms are monotone (i.e., "better" sets must correspond to higher valuations), in our case the resulting mechanisms can be highly non-monotone. For example, it can be the case that the set $S_1 \subseteq S_2$ is won for a known double-minded bidder[7] with valuation (v_1, v_2) but not won for $b_i = (v_1 + \epsilon, v_2 - \epsilon)$, i.e., there is a discontinuity in bids winning S_1. This makes the class of

7. This terminology indicates a bidder who is only interested in two subsets of the universe; the two sets (denoted above as S_1 and S_2) are known to the mechanism and the bidders declare only their two valuations to the mechanism.

Algorithm 2: Greedy algorithm

1 Set $S_1, \ldots, S_n = \emptyset$
2 For each good $e \in \mathsf{U}$ do
3 Let j be the bidder with highest $v_j(S_j \cup \{e\}) - v_j(S_j)$
4 $S_j = S_j \cup \{e\}$
5 Return $S = (S_1, S_2, \ldots, S_n)$

these mechanisms hard to design and then the hope to find one with good approximation guarantee appears slim. Therefore, we add money to our mechanisms in order to make the task of obtaining constant approximations possible.

4.2 Strong Verification with Money

We focus on the greedy algorithm introduced by Lehmann et al. (2006) (cf. Algorithm 2). We assume that the selection of the bidder with maximum marginal valuation in line 3 uses a fixed bid-independent tie-breaking rule. Algorithm 2 is a greedy adaptive priority algorithm. In particular, given the allocation $S_1, \ldots, S_n$ of goods considered so far and the current good, it orders first the elementary bids $v_j(S_j \cup \{e\})$ in nonincreasing order of their marginal valuations $v_j(S_j \cup \{e\}) - v_j(S_j)$ and next all other bids in an arbitrary order (note that Borodin & Lucier, 2016, Section 4 uses a slightly different way of describing Algorithm 2 as a priority algorithm using goods as input atomic items). Since Algorithm 2 is an adaptive priority algorithm, it can accept any number of bids from the same bidder (i.e., it can allocate many different items to the same bidder).

Theorem 5. *There exists a payment function which paired with Algorithm 2 gives rise to a truthful mechanism with strong verification for CAs with submodular bidders, bidding from finite domains, given by value oracles.*

Proof. By Proposition 1, Algorithm 2 (denoted as A in the rest of the proof) is part of a truthful mechanism as long as no declaration graph associated to it has negative weight cycles. We will prove that this is indeed the case.

Fix i and $\mathbf{b}_{-i}$ and consider the declaration graph associated to A. Consider a generic cycle $C := a^0 \to a^1 \to \ldots \to a^k \to a^{k+1} = a^0$ of this graph. By existence of edges, for all $j = 0, \ldots, k$, we have:

$$a^j(T) \geq a^{j+1}(T) \; \forall T \subseteq A_i(a^{j+1}, \mathbf{b}_{-i}). \tag{13}$$

This yields that for all $j = 0, \ldots, k$

$$a^j(U) = a^{j+1}(U) \; \forall U \subseteq \cap_{l=1}^{k} A_i(a^l, \mathbf{b}_{-i}). \tag{14}$$

Now let r be the first good of U in the order considered by the algorithm in line 2 that is in $\cup_{l=1}^{k} A_i(a^l, \mathbf{b}_{-i})$ but not in $\cap_{l=1}^{k} A_i(a^l, \mathbf{b}_{-i})$; let A_r be the set assigned to bidder i up to the point in which r is considered. Without loss of generality, let $r \in A_i(a^1, \mathbf{b}_{-i})$. Since $r \in A_i(a^1, \mathbf{b}_{-i})$, then in $(a^1, \mathbf{b}_{-i})$ bidder i has the maximum marginal valuation for r. But then by (13) and (14), we have $a^0(A_r \cup \{r\}) \geq a^1(A_r \cup \{r\})$ and $a^0(A_r) = a^1(A_r)$, respectively. This yields $a^0(A_r \cup \{r\}) - a^0(A_r) \geq a^1(A_r \cup \{r\}) - a^1(A_r)$, which, in turn,

implies that bidder i has maximum marginal for r also in bid vector $(a^0, \mathbf{b}_{-i})$. We then have that $r \in A_i(a^0, \mathbf{b}_{-i})$ as well. By reiterating the argument, we have that $r \in A_i(a^l, \mathbf{b}_{-i})$ for $l = 0, \ldots, k$ and then $A_i(a^j, \mathbf{b}_{-i}) = A_i(a^{j+1}, \mathbf{b}_{-i})$ $\forall j = 0, \ldots, k$. All edges in C have thus weight 0. □

The theorem above guarantees the existence of payments that enforce truthfulness when the bidding domains are finite. We are left with the question of how to efficiently compute payments. The first immediate observation is that whenever the size of the graph is polynomial in the number of players and goods, then we can simply compute shortest paths as Rochet's (1987) theorem suggests.

Corollary 1. *There exists a 2-approximate truthful mechanism with strong verification for CAs with submodular bidders given by value oracles bidding from finite domains. The allocation is computable in time polynomial in $n+m$. If bidders' domains are of polynomial size in $m+n$, then payments can also be computed efficiently.*

To remove the assumption of Corollary 1 on the domains, one would need to depart from shortest paths and find a different way to define payments (e.g., via a closed formula) allowing efficient computation. A common approach is to understand the structure of the graph. The graph is very well structured. Indeed, by the proof of Theorem 5 any cycle is a clique of nodes corresponding to declarations to which the algorithm assigns the same set S, $a(T) = b(T)$ for all $T \subseteq S$ and declarations a and b in the clique. Since all its edges have weight 0, the clique can be contracted and the same payment assigned to all these declarations. We are left with a directed acyclic graph, holding out the hope to compute payments efficiently.

4.2.1 Weak Verification is not Sufficient for Truthfulness

We show that weak verification is not sufficient for the payments of Theorem 5 to exist. Namely, we give a domain for bidder 2 and a declaration for bidder 1 such that no matter what order Algorithm 2 uses in line 2, the declaration graph of bidder 2 has a negative-weight cycle, when weak verification is used.

We have two goods, named α and β, so that $\mathsf{U} = \{\alpha, \beta\}$. There are two submodular bidders $N = \{1, 2\}$. A valuation function v is represented as a triple (x, y, z), where $v(\{\alpha\}) = x$, $v(\{\beta\}) = y$, and $v(\{\alpha, \beta\}) = z$. We consider valuation function $v_1 = (x_1, y_1, z_1)$ for bidder 1 and domain $D_2 = \{b_2, b_2', b_2'', b_2'''\}$ for bidder 2. We let $b_2 = (x_2, y_2, z_2)$, $b_2' = (x_2', y_2', z_2')$, $b_2'' = (x_2'', y_2'', z_2'')$, and $b_2'' = (x_2''', y_2''', z_2''')$. We show that if v_1, b_2 and b_2' (respectively, v_1, b_2'' and b_2''') satisfy a set of linear constraints, there is a negative-weight 2-cycle in the declaration graph when weak verification is used and Algorithm 2 considers first good α and then good β (respectively, first good β and then good α).

We first note that if the following inequalities hold

$$x_2 > x_1\,, \quad z_2 - x_2 > y_1\,, \quad x_1 > x_2'\,, \quad y_2' > z_1 - x_1\,,$$

and Algorithm 2 considers first good α and then β, bidder 2 with declaration b_2 is allocated $\{\alpha, \beta\}$ and with declaration b_2' is allocated $\{\beta\}$. If weak verification is used and the following inequalities hold

$$y_2 \geq y_2'\,, \quad z_2' \geq z_2\,,$$

both edges (b_2, b_2') and (b_2', b_2) are present in the declaration graph of bidder 2, as implied by the allocations above and the definition of weak verification in (2). The weight of (b_2, b_2') in the declaration graph is $z_2 - y_2$ and the weight of (b_2', b_2) is $y_2' - z_2'$. So if Algorithm 2 considers first α and then β and the following inequality is also satisfied

$$z_2 - z_2' < y_2 - y_2' \,,$$

the declaration graph of bidder 2 has a negative-weight 2-cycle (b_2, b_2', b_2). It is straightforward to verify that the system of the five inequalities above is satisfied by infinitely many triples of submodular valuations. For a concrete example, the valuations $v_1 = (9.4, 6, 11)$, $b_2 = (11, 10, 18)$ and $b_2' = (9, 10, 19)$ are submodular and satisfy all the five inequalities above.

Similarly, if Algorithm 2 considers first β and then α and the following inequalities hold

$$y_2'' > y_1 \,, \quad z_2'' - y_2'' > x_1 \,, \quad y_1 > y_2''' \,, \quad x_2''' > z_1 - y_1 \,,$$

bidder 2 with declaration b_2'' is allocated $\{\alpha, \beta\}$ and with declaration b_2''' is allocated $\{\alpha\}$. If weak verification is used and the following inequalities hold

$$x_2'' \geq x_2''' \,, \quad z_2''' \geq z_2'' \,,$$

both edges (b_2'', b_2''') and (b_2''', b_2'') are present in the declaration graph of bidder 2. The weight of (b_2'', b_2''') is $z_2'' - x_2''$ and the weight of (b_2''', b_2'') is $x_2''' - z_2'''$. So if Algorithm 2 considers first β and then α and

$$z_2'' - z_2''' < x_2'' - x_2''' \,,$$

is also satisfied, the declaration graph of bidder 2 has a negative-weight 2-cycle (b_2'', b_2''', b_2''). Again, the system of the five inequalities above is satisfied by infinitely many triples of submodular valuations, including the valuations $v_1 = (9.4, 6, 11)$, $b_2'' = (11, 7, 16.5)$ and $b_2''' = (11, 5.8, 16.8)$.

So, we conclude that there is a valuation v_1 for bidder 1 and a domain D_2 for bidder 2 so that for any order of the goods, Algorithm 2 with weak verification results in a negative cycle in the declaration graph of bidder 2. Therefore, Algorithm 2 does not admit truthful payments for any order in which the goods are processed, unless strong verification is used. Strong verification, on the other hand, would eliminate edges (b_2', b_2) and (b_2''', b_2'') in the respective cases.

5. Computing Payments

The complexity of payment computation is usually no harder than complexity of the corresponding algorithm for the combinatorial problem at hand. This is because explicit formulas for payments are either known (e.g., VCG, single-dimensional domains) or derived from the cycle-monotonicity analysis (Lavi & Swamy, 2009; Krysta & Ventre, 2015). However, when this fails, no other source of hardness for payment computation is sought (Lavi & Swamy, 2009, Section 4) (Ventre, 2014). We initiate here a study of the complexity of payment computation by decoupling the latter from the computational complexity of the algorithm, using Algorithm 2 as a case study.

We begin by observing, via a connection with the implicit payment computation of Babaioff et al. (2015, 2013), that seemingly shortest-path payments cannot overlook the structure of the declaration graph. Informed by this, and in order to prove hardness of payment computation, we enrich the classical mechanism design model by including bidding domains as input to the mechanisms. We will adapt a standard black-box approach and assume that in addition to bidders' declarations, the bidders' domains are either given explicitly as part of the input encoded in binary or they are represented succinctly and the mechanism can access them by queries. The running time of a mechanism is measured as a function of $m + n$ and the size of domains' description or the number of queries, where each query takes constant time.

5.1 Impossibility of Implicit Payment Computation

We present an example indicating that we need to carefully take the structure of the declaration graph into account when we compute the payments for Algorithm 2. As noted above, by Theorem 5, if we fix the declarations of all other bidders to $\mathbf{b}_{-i}$, the declaration graph of bidder i reduces to a directed acyclic graph. Among the nodes of the DAG, we are particularly interested in the node, that we call 0, containing all the declarations mapped by the algorithm to $\emptyset$.

We observe that 0 is a sink of the DAG. Fix i, $\mathbf{b}_{-i}$ and let $a \in D_i$ be a declaration mapped by Algorithm 2 (denoted as A below) to $\emptyset$. Assume that a could lie, according to (3), and say b such that $A_i(b, \mathbf{b}_{-i}) = S \neq \emptyset$, and let r be the first good of S according to the order considered by the algorithm in line 2. By the greedy rule in line 3 (and the fixed tie-breaking rule within), we have that $b(\{r\}) \geq \text{marg}_{-i}(r) \geq a(\{r\})$, with at least one of the inequalities being strict, where $\text{marg}_{-i}(r)$ denotes the maximum marginal on r of bidders other than i at this point. The above inequality chain shows that b is a lie which contradicts (3).

Proposition 1 for acyclic graphs can be proved by setting the payment of bidder i in the declarations profile $(b_i, \mathbf{b}_{-i})$ to the equal to the length of the shortest path from b_i to 0, denoted as $SP(b_i, 0)$. Indeed, fixed i and $\mathbf{b}_{-i}$, for an edge (a, b) of the declaration graph for bidder i, $SP(a, 0) \leq SP(b, 0) + a(A_i(a, \mathbf{b}_{-i})) - a(A_i(b, \mathbf{b}_{-i}))$ which simply rewrites (1). We note that this payment can be negative, i.e., we pay the bidders so that they do not underbid on their allocated set. Moreover, as noted by Krysta and Ventre (2015), scaling down all the payments so that they become non-negative (while remaining truthful) may require that the utility of some bidders becomes negative, i.e., violate voluntary participation.

One could try to compute (the length of the shortest path from b_i to 0 and) the payment of bidder i in $(b_i, \mathbf{b}_{-i})$ by considering the path from b_i to 0 consisting of all declarations obtained by uniformly scaling down all the coordinates b_i. Such a path would go down through all declarations λb_i, for all $\lambda \in [0, 1]$. Then, similarly to Archer, Papadimitriou, Talwar, and Tardos (2003) and Babaioff et al. (2015, 2013), one can set the payment of bidder i as follows:

$$P_i(b_i, \mathbf{b}_{-i}) = b_i(A_i(b_i, \mathbf{b}_{-i}) - \int_0^1 b_i(A_i(\lambda b_i, \mathbf{b}_{-i}))d\lambda. \tag{15}$$

It would be particularly interesting if the payments for Algorithm 2 can be computed in this way because Archer et al. (2003) and Babaioff et al. (2015, 2013) prove that random

sampling w.r.t. λ yields an unbiased estimator of the integral in (15). Therefore, using the techniques of Archer et al. and Babaioff et al., we could estimate the payments in randomized polynomial time, thus turning Algorithm 2 into a truthful-in-expectation mechanism with strong verification. Our next example shows that (15) may not result in the right (shortest path length and) payments, thus indicating that this technique does not work with our declaration graph.

Our example uses 2 goods $\mathsf{U} = \{\alpha, \beta\}$ and 2 submodular bidders $N = \{1, 2\}$, and a valuation function v is a triple (x, y, z), where $v(\{\alpha\}) = x$, $v(\{\beta\}) = y$, and $v(\{\alpha, \beta\}) = z$. We let $v_1 = (11 - \varepsilon, 12, 12)$, for a small $\varepsilon > 0$, $b_2 = (23, 12, 23)$ and $b_2' = (11, 12, 12)$. Algorithm 2 considers first α and next β, and always breaks ties in favour of bidder 1. We first calculate the payment of (15) if bidder 2 bids b_2. Then, $A_2(b_2, v_1) = \{\alpha\}$ and $b_2(A_2(b_2, v_1)) = 23$. Moreover, for all $\lambda \in (\frac{11-\varepsilon}{23}, 1]$, $A_2(\lambda b_2, v_1) = \{\alpha\}$, because $23\lambda > 11-\varepsilon$; for all $\lambda \in (\frac{1+\varepsilon}{12}, \frac{11-\varepsilon}{23}]$, $A_2(\lambda b_2, v_1) = \{\beta\}$, because $23\lambda \leq 11 - \varepsilon$ and $12\lambda > 1 + \varepsilon$; and for all $\lambda \in [0, \frac{1+\varepsilon}{12}]$, $A_2(\lambda b_2, v_1) = \emptyset$. Therefore,

$$b_2(A_2(\lambda b_2, v_1)) = \begin{cases} 23 & \text{for all } \lambda \in (\frac{11-\varepsilon}{23}, 1] \\ 12 & \text{for all } \lambda \in (\frac{1+\varepsilon}{12}, \frac{11-\varepsilon}{23}] \\ 0 & \text{for all } \lambda \in [0, \frac{1+\varepsilon}{12}] \end{cases}$$

Applying (15), we get that

$$\begin{aligned} P_2(b_2, v_1) &= 23 - \left(23\left(1 - \tfrac{11-\varepsilon}{23}\right) + 12\left(\tfrac{11-\varepsilon}{23} - \tfrac{1+\varepsilon}{12}\right)\right) \\ &= 11\,\tfrac{11-\varepsilon}{23} + 1 + \varepsilon = \tfrac{144+12\varepsilon}{23} \end{aligned}$$

So, applying (15) to Algorithm 2 results in bidder 2 paying an amount of $\frac{144+12\varepsilon}{23}$ for declarations (b_2, v_1).

On the other hand, we have that $A_2(b_2, v_1) = A_2(b_2', v_1) = \{\alpha\}$, which implies that the weight of edge (b_2, b_2') in the declaration graph of bidder 2 is 0. Since $A_2(\frac{10}{11}b_2', v_1) = \{\beta\}$, the weight of edge $(b_2', \frac{10}{11}b_2')$ is -1. Therefore, for Algorithm 2 with declarations (b_2, v_1), we should pay an amount of at least 1 to bidder 2.

5.2 A Computational Criticism to Cycle-Monotonicity

We now show that if domains are represented succinctly, payment computation requires (and can reveal) crucial information about the declaration graph, and thus, it is **NP**-hard. Specifically, we show that, unless $\mathbf{P} = \mathbf{NP}$, we cannot efficiently compute minimum payments, i.e., payments defined as shortest paths on the declaration graph.

Theorem 6. *The problem of computing the shortest-path payments for Algorithm 2 with strong verification is* **NP***-hard, even for $n = 2$ bidders and $m = 2$ goods, where bidders' domains are accessed by queries.*

Proof. Suppose we are given an instance with two goods $\mathsf{U} = \{\alpha, \beta\}$ and two bidders $N = \{1, 2\}$ each with submodular valuation on the bundles of U. A valuation function v_i of bidder i is represented as above by a triple (x, y, z), where $v_i(\{\alpha\}) = x, v_i(\{\beta\}) = y, v_i(\{\alpha, \beta\}) = z$, and, because of monotonicity, we have that $x \leq z, y \leq z$, and, because of submodularity, we have that $y \geq z - x$.

We will define an instance of CA (and a particular domain) given an instance of the satisfiability problem which is specified by a CNF input Boolean formula ϕ with k Boolean variables. Let the domain of bidder 2 contain just one declaration $D_2 = \{(x, y, z)\}$ for some fixed positive numbers $x, y, z > 0$ such that $x < z, y < z$, and $y = z - x$. Let $S = \{s_j : j = 1, 2, \ldots, 2^k\}$ be the set of all possible $0-1$ truth assignments to the k Boolean variables of formula ϕ. The domain of bidder 1 is: $D_1 = \{(x + \varepsilon,\, y + \varepsilon,\, x + y + \varepsilon/2)\} \cup \{(x - \varepsilon/j,\, y + \varepsilon/2,\, x + y + \varepsilon/2 - \varepsilon/j) : s_j \text{ satisfies } \phi, j = 1, \ldots, 2^k\} \cup \{(x - \varepsilon/j,\, y + 2\varepsilon,\, x + y + 2\varepsilon - \varepsilon/j) : s_j \text{ does not satisfy } \phi, j = 1, \ldots, 2^k\}$, where $\varepsilon > 0$ is a small fixed constant, $\varepsilon \ll \min\{x, y\}$.

Since the declaration of bidder 2 is fixed to (x, y, z), the declaration graph for bidder 1 has D_1 as the set of its nodes, and before defining its edges we will make some observations about the greedy algorithm, denoted as A, when α is considered first and β second. When we run A with bidder 1 declaring $(x+\varepsilon,\, y+\varepsilon,\, x+y+\varepsilon/2) \in D_1$, then the outcome for bidder 1 is $\{\alpha\}$ and $\{\beta\}$ for bidder 2. For any declaration of bidder 1 from $D_1 \setminus \{(x+\varepsilon,\, y+\varepsilon,\, x+y+\varepsilon/2)\}$ the A's outcome for bidder 1 is $\{\beta\}$ and $\{\alpha\}$ for bidder 2.

Given any node $b_1 \in D_1 \setminus \{(x+\varepsilon,\, y+\varepsilon,\, x+y+\varepsilon/2)\}$ and $b_1' = (x+\varepsilon,\, y+\varepsilon,\, x+y+\varepsilon/2) \in D_1$, the declaration graph does not have the edge (b_1, b_1'). This follows from the fact that $b_1(\{\alpha\}) < x$ and $b_1'(\{\alpha\}) > x$, therefore if edge (b_1, b_1') were present this would contradict the no-overbidding verification assumption. Similarly, there is no edge present from b_1' to any node $b_1 \in \{(x - \varepsilon/j,\, y + 2\varepsilon,\, x + y + 2\varepsilon - \varepsilon/j) : s_j \text{ does not satisfy } \phi, j = 1, \ldots, 2^k\}$, as its presence would contradict the no-overbidding assumption for $\{\beta\}$: $b_1'(\{\beta\}) = y + \varepsilon < y + 2\varepsilon = b_1(\{\beta\})$.

Thus, node b_1' is connected to the rest of the declaration graph by edge (b_1', b_1), for some $b_1 \in \{(x - \varepsilon/j,\, y + \varepsilon/2,\, x + y + \varepsilon/2 - \varepsilon/j) : s_j \text{ satisfies } \phi, j = 1, \ldots, 2^k\}$. And, deciding if node b_1' is connected to the rest of the declaration graph (and thus, whether Algorithm 2 requires a non-zero payment for b_1') is equivalent to deciding if formula ϕ is satisfiable.

Therefore, the problem of computing the payment for node b_1' is **NP**-hard provided that we can represent the declaration graph succinctly in time polynomial in k and the size of ϕ. Such a representation has been shown for the satisfiability problem by small circuits by Galperin and Wigderson (1983, Definition 2.2 and Lemma 2.2). This representation essentially reduces to answering in polynomial time (in the size of ϕ) a question if given $b_1 \in D_1 \setminus \{(x+\varepsilon,\, y+\varepsilon,\, x+y+\varepsilon/2)\}$, does there exists edge (b_1', b_1), which is just checking if the appropriate given $0/1$ truth assignment s_j (defining node b_1) satisfies formula ϕ or not. Thus, our reduction represents the domain D_1 by presenting an encoding of formula ϕ to the mechanism and mechanism asks queries of the type "does b_1 satisfy ϕ?".

We will present a further explanation to as why this implies that the problem of computing the payment for node b_1' is **NP**-hard. Suppose we are given an instance ϕ of the satisfiability problem with k variables. Then we encode it as the declaration graph as above. But because this graph is of exponential size in the size of ϕ, we need to represent it succinctly by small circuits as above: formula ϕ is easily translated into a polynomial size circuit in polynomial time (see Galperin & Wigderson, 1983, proof of Lemma 2.2). This representation is now fed to a hypothetical polynomial time (in the size of ϕ) algorithm that computes payment for node b_1'. This algorithm accesses the declaration graph by querying the circuit and decides if node b_1' is connected to the rest of the graph. If yes, then we know that ϕ is satisfiable and if no then ϕ is not satisfiable. This gives a polynomial

time algorithm for the satisfiability problem and thus shows **NP**-hardness of the payment computation. □

We complement this result by showing that for $m = 2$, it is indeed possible to compute efficiently payments that are not defined as shortest paths on the declaration graph. Let α, β denote the two items and assume that Algorithm 2 considers α before β.

Definition 3. *Let P_i^ω be the payment function defined as $P_i^\omega(b, \mathbf{b}_{-i}) = 0$ if $A_i(b, \mathbf{b}_{-i}) \in \{\{\alpha, \beta\}, \{\beta\}, \emptyset\}$;*

$$P_i^\omega(b, \mathbf{b}_{-i}) = -b_2(\beta)$$

otherwise, $b_2(\beta)$ being the maximum of what the other bidders declare for $\{\beta\}$.

Theorem 7. *The payment P_i^ω leads to a truthful CA with strong verification when coupled with Algorithm 2 (even for infinite declaration domains).*

Proof. Fix i and $\mathbf{b}_{-i}$ and consider an edge (a, b) of the resulting declaration graph associated to Algorithm 2, denoted A in this proof. Since the payment is independent of the declaration itself there is no problem when $A_i(a, \mathbf{b}_{-i}) = A_i(b, \mathbf{b}_{-i})$. Moreover, as noted above (see Section 5.1), when $A_i(a, \mathbf{b}_{-i}) = \emptyset \neq A_i(b, \mathbf{b}_{-i})$ then edge (a, b) does not exist. The very same argument shows that we cannot have an edge in the case in which $A_i(a, \mathbf{b}_{-i}) = \{\beta\}$ and $A_i(b, \mathbf{b}_{-i}) \supseteq \{\alpha\}$.

Let us now focus on the case in which $A_i(a, \mathbf{b}_{-i}) \supseteq \{\alpha\}$. This means that the algorithm first allocates α to i and then when allocating β, it looks at $a(\{\alpha, \beta\}) - a(\{\alpha\})$ in comparison with $b_j(\beta)$ for all bidders $j \neq i$. In the case in which $A_i(a, \mathbf{b}_{-i}) = \{\alpha, \beta\}$, it is then the case that

$$a(\{\alpha, \beta\}) - a(\{\alpha\}) \geq b_2(\beta), \tag{16}$$

whilst when $A_i(a, \mathbf{b}_{-i}) = \{\alpha\}$ we can conclude that

$$a(\{\alpha, \beta\}) - a(\{\alpha\}) \leq b_2(\beta). \tag{17}$$

We use (16) to prove truthfulness for the case in which $A_i(a, \mathbf{b}_{-i}) = \{\alpha, \beta\}$ and $A_i(b, \mathbf{b}_{-i}) = \{\alpha\}$. Specifically,

$$\begin{aligned} a(A_i(a, \mathbf{b}_{-i})) - P_i^\omega(a, \mathbf{b}_{-i}) &= a\{\alpha, \beta\} \\ &\geq a(\{\alpha\}) + b_2(\beta) \\ &= a(A_i(b, \mathbf{b}_{-i})) - P_i^\omega(b, \mathbf{b}_{-i}). \end{aligned}$$

Similarly, we can use (17) to prove the truthfulness when $A_i(a, \mathbf{b}_{-i}) = \{\alpha\}$ and $A_i(b, \mathbf{b}_{-i}) \in \{\{\alpha, \beta\}, \{\beta\}\}$.

The remaining cases, (i) $A_i(a, \mathbf{b}_{-i}) = \{\alpha, \beta\}$ and $A_i(b, \mathbf{b}_{-i}) = \{\beta\}$ and (ii) $A_i(a, \mathbf{b}_{-i}) = \{\beta\}$ and $A_i(b, \mathbf{b}_{-i}) = \emptyset$, follow from absence of payments and monotonicity of valuations. □

An important observation is that we use the outcome graph (i.e., same payment for all the nodes mapped by the algorithm to the same outcome) and therefore by Theorem 7, the outcome graph for $m = 2$ has no negative-weight cycles. There are many intuitive reasons for which the idea behind P_i^ω would not extend to three items α, β, γ considered in this order by the algorithm (e.g., marginals do not align neatly for all possible outcomes; hard

to handle "jumps" from $\{\alpha\}$ to $\{\alpha, \beta, \gamma\}$). We give a formal argument for these intuitions. This approach of payment-per-outcome fails because the outcome graph might have negative weight cycles.

Theorem 8. *There exists an instance of CAs with submodular bidder with $n = 2$ and $m = 3$, for which the outcome graph associated to Algorithm 2, defined upon strong verification, has a negative weight cycle.*

Proof. Let A denote Algorithm 2 and α, β, γ be the three goods in the universe. We are going to assume that A considers α first, β second and γ eventually. Define the bid b_2 of bidder 2 to be $b_2(X) = |X|$ for each $X \subseteq \{\alpha, \beta, \gamma\}$. Consider the domain of bidder 1 to be $D_1 = \{a, b, c, d\}$ where for $3/7 > \epsilon > 0$:

	a	b	c	d
$\{\alpha\}$	$1+\epsilon$	$1+\epsilon$	$1+\epsilon$	$1+\epsilon/3$
$\{\beta\}$	2	2	$1+\epsilon$	$1+\epsilon$
$\{\gamma\}$	2	2	$1+5/3\epsilon$	$1+5/3\epsilon$
$\{\alpha, \beta\}$	$2+2\epsilon$	2	$2+2/3\epsilon$	$2+2/3\epsilon$
$\{\alpha, \gamma\}$	$3+\epsilon$	$3+\epsilon$	$2+2\epsilon$	$2+2\epsilon$
$\{\beta, \gamma\}$	$3+\epsilon$	$3+\epsilon$	2	2
$\{\alpha, \beta, \gamma\}$	$3+\epsilon$	$3+\epsilon$	$2+2\epsilon$	$2+2\epsilon$

By inspection, the four bids are submodular. Moreover, given our assumption on the good ordering, we get that $A_1(a, b_2) = \{\alpha, \beta\}$, $A_1(b, b_2) = \{\alpha, \gamma\}$, $A_1(c, b_2) = \{\alpha, \gamma\}$ and $A_1(d, b_2) = \{\alpha, \beta\}$. It is not hard to check that the edges (a, b) and (c, d) belong to the declaration graph associated to A. Their weight is $-1 + \epsilon$ and $4/3\epsilon$, respectively. However, as $A_1(a, b_2) = A_1(d, b_2) = \{\alpha, \beta\}$ and $A_1(b, b_2) = A_1(c, b_2) = \{\alpha, \gamma\}$ then the outcome graph (cf. Definition 1) has a cycle $\{\alpha, \beta\} \to \{\alpha, \gamma\} \to \{\alpha, \beta\}$ of weight $-1 + 7/3\epsilon < 0$. □

5.3 Comparison with the VCG Mechanism

We now prove that there are settings of our model relevant to practical applications where we have a provable advantage over VCG. Specifically, we show that there are settings where VCG is computationally intractable, but the positive results in this paper apply and give a truthful constant approximation of the social welfare, in polynomial-time.

Theorem 9. *There is a setting of CAs with submodular bidders, for which the problem of computing payments for Algorithm 2 is solvable in polynomial time, but the problem of finding a social welfare maximizing allocation is* **NP***-hard to approximate within a factor $2e/(2e-1) - \varepsilon$ for any $\varepsilon > 0$.*

Proof. We will describe instances of CAs with submodular bidders for which Dobzinski and Vondrák (2013) prove that the social welfare maximization problem is **NP**-hard to approximate within any constant factor better than $2e/(2e-1) \approx 1.225$. Fix any small $\varepsilon > 0$ and we are given a universe U of $|\mathsf{U}| = m$ goods and a set of n bidders. Dobzinski and Vondrák use the following instances of the Max n-Cover problem (see Dobzinski & Vondrák, 2013, p. 9). Let $\mathcal{S} \subset 2^{\mathsf{U}}$ be a family of sets partitioned into subfamilies $\mathcal{S}_1, \ldots, \mathcal{S}_n$, such

that: every set in $\mathcal{S}$ has the same size s; $|\mathcal{S}_1| = |\mathcal{S}_2| = \cdots = |\mathcal{S}_n| = g$; for every two sets $S, T \in \mathcal{S}$, $|S \cap T| \leq \varepsilon \cdot s$. Note that s and g are absolute fixed constant integers.

The submodular valuation function v_i of any bidder $i \in \{1, \ldots, n\}$ is defined as function f in Definition 3.3 (see Dobzinski & Vondrák, 2013, p. 5), where we use the constant size family $\mathcal{S}_i \subset 2^{\mathsf{U}}$ in place of family $\mathcal{F}$ and $a = \frac{1}{2s}$, $b = \varepsilon \cdot s$. Observe that, given $\varepsilon > 0$ and the parameters m, n, s, g of this construction and the family $\mathcal{S}_i$, function v_i for bidder i is uniquely and succinctly defined. Suppose now that the declaration of any bidder $j \in \{1, \ldots, n\} \setminus \{i\}$ is fixed to his declared valuation v_j as defined above for his family $\mathcal{S}_j$. Then the set D_i of all possible declared valuations v_i of bidder i of the type defined above has size of at most the number of possible subfamilies $\mathcal{S}_i$. Subfamily $\mathcal{S}_i$ has g subsets of U each of size s, thus $|D_i| \leq \left(\binom{m}{s}\right)^g$, which is polynomial in m because s and g are fixed constants. Thus, the declaration graph of Algorithm 2 has polynomial size and payments are polynomial time computable. □

6. Conclusions

We considered the question of understanding the structure of truthful greedy mechanisms, by characterizing the kind of bidders' misbehavior that needs to be verified. In the context of CAs, we showed that *any* greedy fixed priority algorithm is truthful as long as bidders cannot overbid on the subset of goods they win (if any). We then proved that greedy adaptive priority algorithms require stronger assumptions, by proving that the 2-approximation algorithm for CAs with submodular bidders (Lehmann et al., 2006) has to also verify overbidding on the subsets of the awarded set of goods.

The main question left open by our work concerns the computational complexity of computing the payments for the aforementioned algorithm. Our results provide a first, meaningful case study for the problem of assessing the computational efficiency of the cycle-monotonicity technique. We conjecture that computing these payments is actually a hard problem. However, this appears hard to prove, given that the question is not about optimization but rather search (that is, the existence of a feasible solution is guaranteed and the question is about computing any feasible solution versus the shortest path solution). This is similar to the question about the computational complexity of computing Nash equilibria where a solution is always guaranteed to exist; therefore, for our problem, the tools adopted to prove the hardness of computing Nash equilibria seem to be required.

Acknowledgments

This work is partially supported by Greek NSRF grant Algorithmic Game Theory/THALES and EPSRC through grants EP/K01000X/1 and EP/M018113/1. A preliminary version of this work was published as a conference paper (Fotakis, Krysta, & Ventre, 2015).

References

Archer, A., Papadimitriou, C. H., Talwar, K., & Tardos, É. (2003). An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Mathematics*, *1*(2), 129–150.

Babaioff, M., Kleinberg, R. D., & Slivkins, A. (2015). Truthful mechanisms with implicit payment computation. *J. ACM*, *62*(2), 10:1–10:37.

Babaioff, M., Kleinberg, R., & Slivkins, A. (2013). Multi-parameter mechanisms with implicit payment computation. In *Proceeding of ACM Conference on Electronic Commerce, EC '13*, pp. 35–52.

Blumrosen, L., & Nisan, N. (2009). On the computational power of demand queries. *SIAM J. Comput.*, *39*(4), 1372–1391.

Borodin, A., & Lucier, B. (2016). On the limitations of greedy mechanism design for truthful combinatorial auctions. *ACM Trans. Economics and Comput.*, *5*(1), 2:1–2:23.

Borodin, A., Nielsen, M., & Rackoff, C. (2003). (Incremental) priority algorithms. *Algorithmica*, *37*(4), 295–326.

Celik, G. (2006). Mechanism design with weaker incentive compatibility constraints. *Games and Economic Behavior*, *56*(1), 37–44.

Clarke, E. (1971). Multipart Pricing of Public Goods. *Public Choice*, 17–33.

Daniely, A., Schapira, M., & Shahaf, G. (2015). Inapproximability of truthful mechanisms via generalizations of the VC dimension. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pp. 401–408.

Dobzinski, S. (2011). An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011*, pp. 139–148.

Dobzinski, S., Nisan, N., & Schapira, M. (2005). Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC 2005*, pp. 610–618.

Dobzinski, S., & Vondrák, J. (2012). The computational complexity of truthfulness in combinatorial auctions. In *Proceedings of ACM Conference on Electronic Commerce, EC '12*, pp. 405–422.

Dobzinski, S., & Vondrák, J. (2013). Communication complexity of combinatorial auctions with submodular valuations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pp. 1205–1215.

Fotakis, D., Krysta, P., & Ventre, C. (2014). Combinatorial auctions without money. In *Proceedings of International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14*, pp. 1029–1036.

Fotakis, D., Krysta, P., & Ventre, C. (2015). The power of verification for greedy mechanism design. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015*, pp. 307–315.

Galperin, H., & Wigderson, A. (1983). Succinct representations of graphs. *Information and Control*, *56*(3), 183–198.

Groves, T. (1973). Incentive in Teams. *Econometrica*, *41*, 617–631.

Khot, S., Lipton, R., Markakis, E., & Mehta, A. (2005). Inapproximability results for combinatorial auctions with submodular utility functions. In *Proceedings of Internet and Network Economics, WINE 2005*, pp. 92–101.

Kovács, A., Meyer, U., & Ventre, C. (2015). Mechanisms with monitoring for truthful ram allocation. In *Proceedings of the International Conference on Web and Internet Economics, WINE 2015*, pp. 398–412.

Krysta, P., & Ventre, C. (2015). Combinatorial auctions with verification are tractable. *Theoretical Computer Science*, *571*, 21–35.

Lavi, R., & Swamy, C. (2009). Truthful mechanism design for multidimensional scheduling via cycle monotonicity. *Games and Economic Behavior*, *67*(1), 99–124.

Lehmann, B., Lehmann, D. J., & Nisan, N. (2006). Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, *55*(2), 270–296.

Lehmann, D. J., O'Callaghan, L., & Shoham, Y. (2002). Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, *49*(5), 577–602.

Mirrokni, V., Schapira, M., & Vondrák, J. (2008). Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC-2008), 2008*, pp. 70–77.

Mu'Alem, A., & Nisan, N. (2008). Truthful approximation mechanisms for restricted combinatorial auctions. *Games and Economic Behavior*, *64*(2), 612–631.

Nisan, N., & Ronen, A. (2001). Algorithmic Mechanism Design. *Games and Economic Behavior*, *35*, 166–196.

Nisan, N., Roughgarden, T., Tardos, E., & Vazirani, V. (Eds.). (2007). *Algorithmic Game Theory*. Cambridge University Press.

Penna, P., & Ventre, C. (2014). Optimal collusion-resistant mechanisms with verification. *Games and Economic Behavior*, *86*, 491–509.

Rochet, J. (1987). A Condition for Rationalizability in a Quasi-Linear Context. *Journal of Mathematical Economics*, *16*, 191–200.

Serafino, P., Vidali, A., & Ventre, C. (2017). Truthfulness on a budget: Trading money for approximation through monitoring. Submitted.

Vazirani, V. (2001). *Approximation Algorithms*. Springer.

Ventre, C. (2014). Truthful optimization using mechanisms with verification. *Theor. Comput. Sci.*, *518*, 64–79.

Vickrey, W. (1961). Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, 8–37.

Vohra, R. (2011). *Mechanism Design: A Linear Programming Approach*. Cambridge University Press.

Vondrák, J. (2008). Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, pp. 67–74.

Wilkens, C. A., Cavallo, R., & Niazadeh, R. (2017). GSP: the cinderella of mechanism design. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*, pp. 25–32.