The Power of Verification for Greedy Mechanism Design[*]

Dimitris Fotakis
National Technical Univ. of Athens, Greece
fotakis@cs.ntua.gr

Piotr Krysta
University of Liverpool, UK
pkrysta@liverpool.ac.uk

Carmine Ventre
Teesside University, UK
c.ventre@tees.ac.uk

ABSTRACT

Greedy algorithms are known to provide near optimal approximation guarantees for Combinatorial Auctions (CAs) with multidimensional bidders, ignoring incentive compatibility. Borodin and Lucier [5] however proved 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 subsets 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 for the greedy algorithm of [20], which is 2-approximate for submodular CAs, to become truthful with money in finite bidding domains. Our proof is based on an interesting structural analysis of the strongly connected components of the declaration graph.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Theory, Economics

Keywords

Algorithmic Mechanism Design; Mechanisms with Verification; Combinatorial Auctions

1. INTRODUCTION

Algorithmic mechanism design (AMD) is the study of optimization problems where part of the input data is private data of selfish agents. The goals of the protocol, called a *truthful mechanism*, 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. AMD problems have numerous applications in areas like, e.g., e-commerce, online auctioning, shopping systems.

Combinatorial Auctions (CAs) with the objective to maximize the social welfare 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 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 approximate in polynomial time. For vast literature on the design of approximate polynomial-time mechanisms for various versions of CAs, see, e.g., [25].

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 the weakest possible assumption used in literature to tackle the issue of exponentially (in n and m) large valuations [4]. 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 the best possible $\sqrt{m}$-approximation [21].

Despite achieving an optimal approximation ratio for CAs with single-minded bidders, greedy truthful mechanisms perform much worse for more general valuation functions. Borodin and Lucier [5] 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 algorithm (i.e., mechanism using a priority algorithm to compute its outcome) has an approximation ratio $\Omega(m)$. The results of [5] imply that any truthful priority algorithm does not achieve 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

[*] This work is partially supported by EPSRC through grants EP/K01000X/1 and EP/M018113/1 and by Greek NSRF grant Algorithmic Game Theory/THALES.

[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.

greedy approximation algorithm for submodular CAs gives an approximation guarantee of 2 [20]. 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 do not exist payments that make this algorithm truthful [20]. Moreover, Borodin and Lucier [5] 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 here in which general circumstances of multi-dimensional CAs, we can have truthful greedy mechanisms. Inspired by [18], 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., [7]). We want to understand what kind of relaxed notion 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 [9].

Our Contribution. Priority algorithms comprise a widely accepted general model of greedy-like algorithms [6]. 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. Extending and generalizing previous results in [18, 14], 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, 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. Observe that such general characterization-type results in AMD are scarce in literature.

To deal with the additional power of adaptive priority algorithms (cf. [5]), we introduce the concept of *strong verification*, where the bidders are not allowed to overbid on *any* subset of their winning set. We first characterize all 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. Thus, we study strong verification with money.

As our main technical result, we show that the 2-approximate (adaptive priority) greedy algorithm of [20] is truthful with money 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 [27, 30], which basically requires to study the cycles of the so-called *declaration graph*. 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 cycles (see, e.g., [27, 30]). Here, 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, 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 making the greedy algorithm truthful with money, even if it is allowed to execute all possible orders of goods.

We finally turn our attention to the computational complexity of computing the payments, 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 mechanism (once), is the technique of Babaioff et al. [3, 2]. 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 equivalently be 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), 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 indeed, we can always efficiently find (non-shortest path) payments in our declaration graph in the restricted case with $m = 2$ goods and any number of bidders, by carefully exploiting the structure of the declaration graph in this case. We in fact prove that the associated outcome graph, i.e., the declaration graphs with its 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 for payment computation.

Due to lack of space, some of the proofs are deferred to the full version of the paper.

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., [19, 18].

To put our results in a broader mechanism design context, it is in place to compare them with the celebrated VCG mechanism [29, 8, 16]. 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 the VCG, which requires an optimal solution to a computationally difficult problem (VCG does not need verification though). The crucial point is that the CAs setting relevant to applications is 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 allocation graph associated to the greedy mechanism has polynomial size and thus payments are computable in time polynomial time 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 on subsets of only fixed constant size subset of goods. Interestingly, for another such practically relevant setting of CAs with constant-size submodular valuations, Dobzinski and Vondrák [13] 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 is of size polynomial in n and m, we can compute the payments in polynomial time, see Theorem 8.

CAs with submodular bidders were widely studied both from algorithmic and AMD perspective, see, e.g., [22, 17, 31, 20, 11, 10, 12, 13]. CAs with submodular bidders given by value oracles as an optimization problem (without strategic consideration) is **NP**-hard to solve optimally and **NP**-hard to approximate with factors better than $e/(e-1)$ [17]; such factors cannot be obtained with using only polynomially many value oracles [22]. On the other hand, there are the following polynomial time algorithms: a deterministic greedy 2-approximation [20] and a randomized $e/(e-1)$-approximation algorithm [31]. Now, considering truthful mechanisms for this problem with payments, very strong non-constant lower bounds on the approximation ratios are known. Dobzinski [10] 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 [11], which is best possible by this hardness result. Moreover, Dobzinski and Vondrák [12] 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 $\mathbf{NP} = \mathbf{RP}$. Similar strong non-constant lower bound holds even for randomized polynomial time truthful in expectation mechanisms for CAs with succinctly represented submodular valuations [12].[2] In light of such strong hardness results, our 2-approximate truthful mechanism assumes particular relevance (also considering its applicability to practical auction domains).

The concept of a posteriori verification (i.e., verification that depends on the output computed) in AMD dates back to the seminal work of Nisan and Ronen [24]; subsequently, it has been extensively studied in machine scheduling settings (see, [26] and references therein) and introduced to CAs in [18]. The feasibility of trading verification with money in mechanism design for CAs has been studied in [14].

2. MODEL AND PRELIMINARIES

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 *true 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. 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 $\mathbf{b}_{-i}$:

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

In the *(weak) verification model* [18, 14] 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 whenever $A_i(b_i, \mathbf{b}_{-i}) \neq \emptyset$:

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

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

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

When (2) or (3) is not satisfied, the bidder is caught lying by the relevant verification step and the mechanism punishes her so 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) ((3), respectively) does not hold for the weak (strong, respectively) 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) ((3), respectively) is fulfilled. All the concepts can be adapted to the setting without money when $P_i(\mathbf{b}) = 0$ for any i and $\mathbf{b}$.

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 $S, T \subseteq \mathsf{U}$, $b_i(S) + b_i(T) \geq b_i(S \cup T) - b_i(S \cap T)$.

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.

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

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 $a \to_V b$ (e.g., $a \to_V b$ if and only if (2) is true whenever V is weak verification). The *declaration graph* associated to algorithm A has a vertex for each possible declaration in the domain D_i. For the verification V, we add an arc between a and b in D_i whenever $a \to_V b$. 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. The following known result relates the edges/cycles of the declaration graph to the existence of payments leading to truthful auctions.

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 s.t. (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 [27, 30] to the verification setting V as in [28]. 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 [28].

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 ([28]). *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 condition which is necessary and sufficient for the existence of payments-per-outcome implementing algorithm A.

Greedy Priority Algorithms. We briefly introduce priority algorithms, following [6]. 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 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 grating S to bidder i is feasible for the problem at hand (e.g., S is disjoint to the previously allocated subsets of U).

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. Below, we call $A_i(\mathbf{b})$ the winning component of bidder i. Our first result concerns truthfulness without money.

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 iff $a(Y) \geq b(Y)$, $Y = A_i(b, \mathbf{b}_{-i})$. The edge weights $a(A_i(a, \mathbf{b}_{-i})) - a(Y)$. 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 $b(Y) \geq b_j^*(Z) \geq a(Y)$, 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. □

Let $f_i : \mathbb{R}^+ \to \mathbb{R}^+$ be a strictly increasing function and assume that the greedy fixed priority algorithm uses func-

[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 $a(\emptyset) = b(\emptyset) = 0$.

tion f_i to rank the bids $b_i(S)$ of bidder i for component S. This generalization can add negative-weight edges (cf. [18, Example 2]) but no negative-weight cycle, as shown next.

Theorem 2. *A greedy fixed priority algorithm that orders elementary bids using strictly monotone functions f_i of their value (and breaks ties in a fixed manner independent from 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 $C := a^0 \to a^1 \to \ldots \to a^k \to a^{k+1} = a^0$.

Let us first consider the case in which 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 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 $Y_X = A_{j_X}(b_{j_X})$ 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 $f_i(b(Z)) \geq f_{j_Z}(b_{j_Z}(A_{j_Z}(b_{j_Z}))) \geq f_i(a(Z))$. 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.

Let us now focus on the case in which $A_i(a^j) \neq \emptyset$ for all $j \in \{0, \ldots, k\}$. By definition of the priority algorithm, for $0 \leq j \leq k$, $f_i(a^j(A_i(a^j))) \geq f_i(a^j(A_i(a^{j+1})))$; while the existence of edge (a^j, a^{j+1}) yields $a^j(A_i(a^{j+1})) \geq a^{j+1}(A_i(a^{j+1}))$ and then $f_i(a^j(A_i(a^{j+1}))) \geq f_i(a^{j+1}(A_i(a^{j+1})))$. 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 $\sum_{j=1}^{k} a^j(A_i(a^j)) - a^j(A_i(a^{j+1})) = 0$. This concludes this 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 it allows some bidder i to overbid on some non-empty winning component X. Consider the instance where X is the most preferred component of i and assume that X is the only component maximizing i's utility. 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 $f_j(b_j(Y)) > f_i(a_i(X))$, $Y = A_j(b_j)$ 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 3-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 $b(X) > a_i(X)$ in such a way that $f_i(b(X)) > f_j(b_j(Y))$ and X is allocated to i. Since the verification allows this lie, then the declaration graph has the edge (a_i, b) of weight $\delta = a_i(A_i(a_i, \mathbf{b}_{-i})) - a_i(X) < 0$. Consider now b' defined as b with the exception that $a_i(A_i(a_i, \mathbf{b}_{-i})) \geq b'(A_i(a_i)) = b(X) - \epsilon$, with $\epsilon < -\delta$. (Since $f_i(b(X)) > f_i(a_i(A_i(a_i, \mathbf{b}_{-i})))$ and f_i is increasing such an ϵ exists.) Observe that $f_i(b'(X)) > f_i(b'(A_i(a_i, \mathbf{b}_{-i})))$ and then $A_i(b') = X$. Therefore, edges (b, b') and (b', a_i) belong to the graph, since there is no overbidding on winning component; (b, b') weighs 0, while (b', a_i) weighs ϵ. The cycle $a_i \to b \to b'$ has then negative weight. □

Remark 1. *Theorems 1 and 2 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 [6].*

For adaptive priority algorithm 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 in [20]. The concept of verification used to implement this algorithm turns out to be the strong verification introduced above.

4. STRONG VERIFICATION

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 3. *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.*

This characterization relates to those in [14] for moneyless mechanisms with weak verification. However, while in the latter case mechanisms are monotone, e.g., as in [23, 21] (i.e., to higher valuations there must correspond "better" sets), in our case the resulting mechanisms can be highly non-monotone. For example, it can be that for a known double-minded bidder[4] with valuation (v_1, v_2), the set $S_1 \supseteq S_2$ is won but for $b_i = (v_1 + \epsilon, v_2 - \epsilon)$ is not, i.e., there is a discontinuity in bids winning S_1. This makes the class of 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 tractable.

Strong verification with money. We focus on the greedy algorithm introduced in [20] (cf. Algorithm 1). We assume that the selection of the bidder with maximum marginal valuation in line 3 uses a fixed bid-independent tie-breaking rule. Algorithm 1 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

[4] This terminology indicates a bidder 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 their two valuations rather than a value oracle.

Algorithm 1: 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 $S = (S_1, S_2, \ldots, S_n)$
```

marginal valuations $v_j(S_j \cup \{e\}) - v_j(S_j)$ and next all other bids in an arbitrary order (note that [5, Section 4] uses a slightly different way of describing Algorithm 1 as a priority algorithm using goods as input atomic items).

THEOREM 4. *There exists a payment function that paired with Algorithm 1 gives rise to a truthful mechanism with strong verification for CAs with submodular bidders given by value oracles bidding from finite domains.*

PROOF. By Proposition 1, Algorithm 1 (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:

$$a^j(T) \geq a^{j+1}(T) \; \forall T \subseteq A_i(a^{j+1}, \mathbf{b}_{-i}). \tag{4}$$

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

$$a^j(U) = a^{j+1}(U) \; \forall U \subseteq \cap_{l=1}^{k} A_i(a^l, \mathbf{b}_{-i}). \tag{5}$$

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 to 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 (4) and (5), 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 from Rochet's theorem.

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 4 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.

Necessity of Strong Verification for the Truthfulness of Algorithm 1. We show that strong (no-overbidding) verification is necessary for the payments of Theorem 4 to exist. Namely, we give a domain for bidder 2 and a declaration for bidder 1 such that no matter what order Algorithm 1 uses in line 2, the declaration graph of bidder 2 has a negative-weight cycle, unless strong verification is used. We have $\mathsf{U} = \{\alpha, \beta\}$ and 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 $v(\{\alpha, \beta\}) = z$. Let $v_1 = (9.4, 6, 11)$ and $D_2 = \{b_2, b_2', b_2'', b_2'''\}$ where $b_2 = (11, 10, 18)$, $b_2' = (9, 10, 19)$, $b_2'' = (11, 7, 16.4)$ and $b_2''' = (11, 5.5, 16.5)$. Note that if Algorithm 1 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 the order of goods is reversed then we observe that $A_2(v_1, b_2'') = \{\alpha, \beta\}$ and $A_2(v_1, b_2''') = \{\alpha\}$. If no strong verification is used, e.g., weak verification would be adopted, then edges (b_2, b_2'), (b_2', b_2), for order α, β, and (b_2'', b_2''') and (b_2''', b_2''), for order β, α, are present in the declaration graph of bidder 2. Moreover, $b_2(\{\alpha, \beta\}) - b_2(\{\beta\}) = 18 - 10 = 8$ and $b_2'(\{\beta\}) - b_2'(\{\alpha, \beta\}) = 10 - 19 = -9$, and then $b_2 \to b_2' \to b_2$ is a negative cycle, if the order considered is α, β. Similarly, in the case in which the order is β, α, we have $b_2''(\{\alpha, \beta\}) - b_2''(\{\alpha\}) = 16.4 - 11 = 5.4$ and $b_2'''(\{\alpha\}) - b_2'''(\{\alpha, \beta\}) = 11 - 16.5 = -5.5$, and then $b_2'' \to b_2''' \to b_2''$ is a negative cycle. Note that strong verification would eliminate edges (b_2', b_2) and (b_2''', b_2'') in the respective cases. Therefore, Algorithm 1 does not admit truthful payments for any order in which the goods are processed, unless strong verification is used.

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 [19, 18]. However, when this fails, no other source of hardness for payment computation is sought [19, Section 4] [28]. We initiate here a study of the complexity of payment computation by decoupling the latter from the computational complexity of the combinatorial algorithm, using Algorithm 1 as a case study.

We begin by observing, via a connection with the implicit payment computation of [2, 3], that shortest-path payments cannot seem to 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 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. 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.

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 1. As noted above, by Theorem 4, 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 1 (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 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 in [18], 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 [1, 2, 3], one can set the payment of bidder i as follows:

$$P_i(b_i, \mathbf{b}_{-i}) = b_i(A_i(b_i, \mathbf{b}_{-i}) - \int_0^1 b_i(A_i(\lambda b_i, \mathbf{b}_{-i}))d\lambda. \quad (6)$$

It would be particularly interesting if the payments for Algorithm 1 can be computed in this way because [1, 2, 3] prove that random sampling w.r.t. λ yields an unbiased estimator of the integral in (6). Therefore, using the techniques of [1, 2, 3], we could estimate the payments in randomized polynomial time, thus turning Algorithm 1 into a truthful-in-expectation mechanism with strong verification. Our next example shows that (6) 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 1 considers first α and next β. Then, $A_2(b_2, v_1) = A_2(b'_2, v_1) = \{\alpha\}$, while $A_2(\frac{10}{11}b'_2, v_1) = \{\beta\}$. Also, $A_2(\lambda b_2, v_1) = \{\alpha\}$ for all $\lambda \in (\frac{11-\varepsilon}{23}, 1]$, $A_2(\lambda b_2, v_1) = \{\beta\}$ for all $\lambda \in (\frac{1+\varepsilon}{12}, \frac{11-\varepsilon}{23}]$, and $A_2(\lambda b_2, v_1) = \emptyset$ for all $\lambda \in [0, \frac{1+\varepsilon}{12}]$. Thus, (6) yields that bidder 2 should give a positive payment of about $12.8 - 0.145\varepsilon + 0.052\varepsilon^2$ to the mechanism, while since the weight of edge (b_2, b'_2) is 0 and the weight of edge $(b'_2, \frac{10}{11}b'_2)$ is -1, we should pay bidder 2 at least 1 for declaration (b_2, v_1).

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 **P** = **NP**, we cannot efficiently compute minimum payments, i.e., payments defined as shortest paths on the declaration graph.

THEOREM 5. *The problem of computing the shortest-path payments for Algorithm 1 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 arcs 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 arc (b_1, b'_1). This follows from the fact that $b_1(\{\alpha\}) < x$ and $b'_1(\{\alpha\}) > x$, therefore if arc (b_1, b'_1) were present this would contradict the no-overbidding verification assumption. Similarly, there is no arc 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 arc (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 1 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 by small circuits by Galperin and Wigderson [15, 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 arc (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 type "does b_1 satisfy ϕ?". □

We complement this result by showing that for $m = 2$, it is indeed possible to compute efficiently different payments that are not defined as shortest paths on the declaration graph. Let α, β denote the two items and assume that Algorithm 1 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 6. *The payment P_i^ω leads to a truthful CA with strong verification when coupled with Algorithm 1 (even for infinite declaration domains).*

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 6, 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 7. *There exists an instance of CAs with submodular bidder with $n = 2$ and $m = 3$, for which the outcome graph associated to Algorithm 1 has a negative weight cycle.*

PROOF. Let A denote Algorithm 1 with item order α, β, γ. 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 $\epsilon > 0$:

	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 and $A_1(a, b_2) = \{\alpha, \beta\}$, $A_1(b, b_2) = \{\alpha, \gamma\}$, $A_1(c, b_2) = \{\alpha, \gamma\}$ and $A_1(d, b_2) = \{\alpha, \beta\}$. It is also not hard to check that the edges (a, b) and (c, d) belong to the declaration graph. Their weight is $-1+\epsilon$ and $4/3\epsilon$, respectively. Therefore the outcome graph has the negative-weight cycle $\{\alpha, \beta\} \rightarrow \{\alpha, \gamma\} \rightarrow \{\alpha, \beta\}$. □

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.

THEOREM 8. *There is a setting of CAs with submodular bidders, for which the problem of computing payments for Algorithm 1 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 [13] 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 are the following instances of the Max n-Cover problem (see [13], page 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 (page 5 in [13]), 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 1 has polynomial size and payments are polynomial time computable. □

6. CONCLUSIONS

We considered the question of how truthful greedy mechanisms are, 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 in [20] 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 algorithm in [20]. 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); more specifically, tools adopted to prove the hardness of computing Nash equilibria seem to be required.

7. REFERENCES

[1] A. Archer, C. H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. of SODA*, pages 205–214, 2003.

[2] M. Babaioff, R.D. Kleinberg, and A. Slivkins. Truthful mechanisms with implicit payment computation. In *Proc. of EC*, pages 43–52, 2010.

[3] M. Babaioff, R.D. Kleinberg, and A. Slivkins. Multi-parameter mechanisms with implicit payment computation. In *Proc. of EC*, pages 35–52, 2013.

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

[5] A. Borodin and B. Lucier. On the limitations of greedy mechanism design for truthful combinatorial auctions. In *ICALP*, pages 90–101, 2010.

[6] A. Borodin, M.N. Nielsen, and C. Rackoff. (Incremental) priority algorithms. *Algorithmica*, 37(4):295–326, 2003.

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

[8] E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.

[9] A. Daniely, M. Schapira, and S. Gal. Inapproximability of truthful mechanisms via generalizations of the VC dimension. In *STOC*, 2015.

[10] S. Dobzinski. An impossibility result for truthful combinatorial auctions with submodular valuations. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 139–148, 2011.

[11] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 610–618, 2005.

[12] S. Dobzinski and J. Vondrák. The computational complexity of truthfulness in combinatorial auctions. In *ACM Conference on Electronic Commerce, EC '12, Valencia, Spain, June 4-8, 2012*, pages 405–422, 2012.

[13] S. Dobzinski and J. Vondrák. Communication complexity of combinatorial auctions with submodular valuations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1205–1215, 2013.

[14] D. Fotakis, P. Krysta, and C. Ventre. Combinatorial auctions without money. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 1029–1036, 2014.

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

[16] T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.

[17] S. Khot, R.J. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. In *Internet and Network Economics, First International Workshop, WINE 2005, Hong Kong, China, December 15-17, 2005, Proceedings*, pages 92–101, 2005.

[18] P. Krysta and C. Ventre. Combinatorial auctions with verification are tractable. *Theoretical Computer Science*, 571:21–35, 2015.

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

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

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

[22] V.S. Mirrokni, M. Schapira, and J. Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *Proceedings 9th ACM Conference on Electronic Commerce (EC-2008), Chicago, IL, USA, June 8-12, 2008*, pages 70–77, 2008.

[23] A. Mu'Alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. of 18th AAAI*, pages 379–384, 2002.

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

[25] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory.* Cambridge University Press, 2007.

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

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

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

[29] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.

[30] R.V. Vohra. *Mechanism Design: A Linear Programming Approach.* Cambridge University Press, 2011.

[31] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 67–74, 2008.