Combinatorial Auctions with Externalities: Basic Properties and Bidding Languages

Piotr Krysta* Tomasz Michalak+ Tuomas Sandholm† Michael Wooldridge*

*Computer Science, University of Liverpool, Liverpool L69 3BX, UK

{pkrysta,mjw}@liverpool.ac.uk

+Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

tpm@ecs.soton.ac.uk

†Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

sandholm+@cs.cmu.edu

August 6, 2010

Abstract

Although combinatorial auctions have received a great deal of attention from the computer science community over the past decade, research in this domain has focused on settings in which a bidder only has preferences over the bundles of goods they themselves receive, and is indifferent about how other goods are allocated to other bidders. In general, however, bidders in combinatorial auctions will be subject to *externalities*: they care about how the goods they are not themselves allocated are allocated to others. Our aim in the present paper is to study such *combinatorial auctions with externalities* from a computational perspective. We first present our formal model, and then develop a classification scheme for the types of externalities that may be exhibited in a bidder's valuation function. We then develop a *bidding language* for combinatorial auctions with externalities. The language uses *weighted logical formulae* to represent bidder valuation functions. We then investigate the properties of this representation: we study the complexity of the winner determination problem, and characterise the complexity of classifying the properties of valuation functions. We then present two approaches to winner determination for our bidding language: an exact approach based on integer linear programming, and approximation methods.

1 Introduction

Combinatorial auctions have been closely studied over the past decade [5]. In a combinatorial auction, a number of goods are simultaneously put to auction, and agents can submit bids for bundles of goods. Within the computer science/AI literature, four main aspects of combinatorial auctions have been considered: *bidding languages*, where the goal is to design compact, expressive, natural, and computationally tractable languages for defining bidder valuation functions [19]; *mechanism design*, where the goal is typically to design bidding, allocation, and payment schemes so that bidders are incentivised to truthfully report their valuation function [20]; *winner determination*, where the goal is typically to compute efficiently a social welfare-maximising allocation of goods to bidders, given a representation of bids/preferences [24, 18]; and *preference elicitation*, where the goal is to elicit efficiently a valuation function from a potential bidder.

Although details differ, a common model for such combinatorial auctions is the following. We have a set $\mathcal{Z}$ of goods to be auctioned to agents $\mathcal{N} = \{a_1, \ldots, a_n\}$, and each agent $a_i \in \mathcal{N}$ has preferences represented by a *valuation function*, $v_i : \mathbf{2}^{\mathcal{Z}} \to \mathbb{R}$, assigning a numeric value to every possible bundle of goods. Implicit within this framework is a rather significant (and arguably rather unrealistic) assumption: that *bidders only have preferences over the allocation of goods that they receive, and are indifferent about how other goods are allocated to other agents*. This point is very well-known in the economics literature, where the term *externality* is used to describe the effect that a transaction has on an individual that is not directly involved in the transaction. If the individual is adversely affected by the transaction, then the externality is said to be *negative*, while if the individual benefits from the transaction, then the externality is *positive*. In a combinatorial auction with externalities, bidders have preferences not just over the bundles of goods *they* receive, but also over the way in which other goods are allocated to *others*. This holds even in the extreme case where a bidder is allocated *no* goods: they may still have preferences over the way in which goods are allocated to others, and if the externalities are sufficiently severe, such a bidder may even be motivated to pay the auctioneer to prevent another agent being allocated some good, even though they themselves are allocated nothing: see [15] for a case study of this, where the goods in question are nuclear weapons!

We emphasise that this is not an artificial consideration: externalities are extremely common in auctions. For example, consider spectrum auctions. In such auctions, telecommunications companies bid for licenses to exploit electromagnetic spectrum. From the point of view of a company, it is beneficial to have licenses in geographically contiguous locations – this makes it easier/cheaper to provide services. Conventional combinatorial auction models can directly capture such preferences. However, from the point of view of a company, it is also *bad* if *another* company is allocated licenses in geographically contiguous locations, since this makes *them* more competitive. This is a *negative externality*, which cannot be captured using conventional combinatorial auction models [5].

In this paper we consider the computational aspects of combinatorial auctions with externalities. We begin by presenting our formal model. In Section 3, we develop a classification of some types of externalities that may be exhibited in a bidder's valuation function. In Section 4, we develop a *bidding language* for combinatorial auctions with externalities. The language uses *weighted logical formulae* to represent bidder valuation functions (cf. [13, 17, 26, 6, 26]). Given this representation, we investigate the complexity of the winner determination problem, and the complexity of classifying the properties of valuation functions we identified in Section 3. We then present two approaches to winner determination for our bidding language: an exact approach based on integer linear programming, and an approximation method.

2 Basic Definitions

We start by assuming a finite, non-empty set $\mathcal{Z} = \{z_1, \ldots, z_m\}$ of *atomic goods*. We assume these goods are indivisible and that each good is unique. We use $Z, Z', Z_1, \ldots$ as variables ranging over subsets of $\mathcal{Z}$. Next, we assume a finite, non-empty set $\mathcal{N} = \{a_0, a_1, \ldots, a_n\}$ of *agents* (a.k.a. bidders). We use $G, G', G_1, \ldots$ as variables for subsets of $\mathcal{N}$, and we refer to such subsets as *groups*.

Allocations: An *allocation* is a function $\alpha : \mathcal{N} \to \mathbf{2}^{\mathcal{Z}}$ such that $\alpha(a_1), \ldots, \alpha(a_n)$ partitions $\mathcal{Z}$. The intended interpretation is that $\alpha(a_i)$ is the set of goods allocated to agent a_i under allocation α. Let $\mathcal{A}(\mathcal{Z}, \mathcal{N})$ denote the set of all possible allocations over $\mathcal{N}$, $\mathcal{Z}$. Where $\mathcal{N}, \mathcal{Z}$ are clear from context, we omit reference to them and write $\mathcal{A}$. Observe that the inverse of an allocation α defines a mapping from $\mathcal{Z}$ to $\mathcal{N}$; we denote this inverse by $\hat{\alpha}$. Thus if $\hat{\alpha}(z) = a_i$, then good $z \in \mathcal{Z}$ is allocated to agent $a_i \in \mathcal{N}$ under allocation α, i.e., $z \in \alpha(a_i)$.

When we need to write allocations explicitly, we use the following notation:

$$\{(Z_0; a_0), (Z_1; a_1), \ldots, (Z_n; a_n)\}$$

with the intended meaning that Z_i is the bundle of goods allocated to agent a_i.

Example 1 *Suppose $\mathcal{N} = \{a_0, a_1, a_2\}$ and $\mathcal{Z} = \{z_1, z_2\}$. Then the feasible allocations are:*

$$\begin{array}{lclllll}
\alpha_0 & = & \{ & (\{z_1, z_2\}; a_0), & (\emptyset; a_1), & (\emptyset; a_2) & \} \\
\alpha_1 & = & \{ & (\emptyset; a_0), & (\{z_1, z_2\}; a_1), & (\emptyset; a_2) & \} \\
\alpha_2 & = & \{ & (\emptyset; a_0), & (\{z_1\}; a_1), & (\{z_2\}; a_2) & \} \\
\alpha_3 & = & \{ & (\{z_2\}; a_0), & (\{z_1\}; a_1), & (\emptyset; a_2) & \} \\
\alpha_4 & = & \{ & (\{z_1\}; a_0), & (\{z_2\}; a_1), & (\emptyset; a_2) & \} \\
\alpha_5 & = & \{ & (\emptyset; a_0), & (\emptyset; a_1), & (\{z_1, z_2\}; a_2) & \} \\
\alpha_6 & = & \{ & (\emptyset; a_0), & (\{z_2\}; a_1), & (\{z_1\}; a_2) & \} \\
\alpha_7 & = & \{ & (\{z_2\}; a_0), & (\emptyset; a_1), & (\{z_1\}; a_2) & \} \\
\alpha_8 & = & \{ & (\{z_1\}; a_0), & (\emptyset; a_1), & (\{z_2\}; a_2) & \}
\end{array}$$

Valuation Functions: In the literature on combinatorial auctions, a *valuation function* for an agent $a_i \in \mathcal{N}$ is usually understood as a function $v_i : \mathbf{2}^{\mathcal{Z}} \to \mathbb{R}$, i.e., a function that gives the value $v_i(Z)$ to agent $a_i \in \mathcal{N}$ of the bundle of goods $Z \subseteq \mathcal{Z}$. Implicit in such a definition of valuation functions is the idea that a valuation depends *only* on the goods that are allocated to a_i, and not on the way that goods are allocated to other agents. In the present paper, we will be concerned with valuation functions for agents that take into account not just the goods allocated to that agent, but also the way that goods are allocated to others. Thus, for our purposes, a valuation for agent $a_i \in \mathcal{N}$ is a function $v_i : \mathcal{A} \to \mathbb{R}$. Let $\mathcal{V}(\mathcal{Z}, \mathcal{N})$ denote the set of valuation functions over $\mathcal{Z}, \mathcal{N}$; again, where context makes $\mathcal{Z}, \mathcal{N}$ clear, we simply write $\mathcal{V}$.

Example 2 *For the allocations as defined in Example 1 let valuation functions for every agent involved in the auction be:*

$$\begin{array}{llll}
\forall_{\alpha \in \mathcal{A}} v_0(\alpha) = 0 & & & \\
v_1(\alpha_0) = 0 & v_1(\alpha_5) = 7 & v_2(\alpha_0) = 1 & v_2(\alpha_5) = 4 \\
v_1(\alpha_1) = 12 & v_1(\alpha_6) = 4 & v_2(\alpha_1) = 6 & v_2(\alpha_6) = 5 \\
v_1(\alpha_2) = 4 & v_1(\alpha_7) = 0 & v_2(\alpha_2) = 3 & v_2(\alpha_7) = 3 \\
v_1(\alpha_3) = 6 & v_1(\alpha_8) = 2 & v_2(\alpha_3) = 5 & v_2(\alpha_8) = 3 \\
v_1(\alpha_4) = 10 & & v_2(\alpha_4) = 9 &
\end{array}$$

Combinatorial Auctions with Externalities: Bringing the above components together, we say a combinatorial auction with externalities is a tuple

$$\langle \mathcal{Z}, \mathcal{N}, v_1, \ldots, v_n \rangle$$

where $\mathcal{Z}$ is the set of goods, $\mathcal{N}$ is the set of agents, and $v_i \in \mathcal{V}$ is the valuation function for agent $a_i \in \mathcal{N}$.

Winner Determination: The WINNER DETERMINATION problem in this setting is analogous to conventional combinatorial auctions: given $\langle \mathcal{Z}, \mathcal{N}, v_1, \ldots, v_n \rangle$, the aim is to find an allocation α^* that maximizes *social welfare*:

$$\alpha^* = \arg\max_{\alpha \in \mathcal{A}} \sum_{a_i \in \mathcal{N}} v_i(\alpha).$$

Before we can say much about this problem, of course, we need to fix on a representation for the valuation functions v_i; we consider this below.

It is worth making some remarks on the relationship of this problem to the standard winner determination problem, as the existence of externalities raises a number of additional issues. Notice that bidders in our setting have valuations over allocations in which *no item is assigned to them*. If such externalities are negative and severe enough, then bidders will have an incentive to *pay the auctioneer for not selling anything*. Perverse as it may seem, such a solution might be efficient from the point of view of utilitarian social welfare. In [15], the following example of this situation is given. In the early 1990s, some nations from the former Soviet Union found themselves in possession of nuclear weapons, even though they themselves had no aspirations to be a nuclear state. Both Russia and the USA were concerned about the nuclear weapons falling into the wrong hands. They therefore made payments to these countries, in effect to ensure that the nuclear weapons were not made available to third parties. Such a possibility is not taken into account in most standard auction mechanisms: the auctioneer is not paid by those who do not win the item not to sell it.

3 Allocations and Valuations

Implicit in our definition of valuation functions $v_i : \mathcal{A} \to \mathbb{R}$ is the idea that the value an agent obtains is not dependent only on the goods it is allocated, but also on the way that other goods are allocated to other agents. Our aim in this section is to dig deeper into this idea: we investigate the *structure* of allocations and valuation functions, with the ultimate aim of classifying the different types of externalities that may be exhibited by valuation functions. Before we can classify valuation functions in this way, we first need to investigate the structure of *allocations*.

3.1 Allocation Structure

We say that two allocations α, α' are *individually equivalent* with respect to a group of agents $G \subseteq \mathcal{N}$ if the allocation of each agent $a_i \in G$ under α is the same as its allocation under α'. We denote the fact that α and α' are individually equivalent with respect to G by $\alpha \sim_G \alpha'$. Formally:

Definition 1 (Individual equivalence w.r.t. G)

$$\alpha_1 \sim_G \alpha_2 \quad \textit{iff} \quad \forall a_i \in G : \alpha_1(a_i) = \alpha_2(a_i).$$

Example 3 *Recall Example 1. For $G = \{a_0\}$, there are the following individual internal equivalence relationships $\alpha_1 \sim_G \alpha_2 \sim_G \alpha_5 \sim_G \alpha_6$, $\alpha_3 \sim_G \alpha_7$, and $\alpha_4 \sim_G \alpha_8$. For $G = \{a_1\}$, we have $\alpha_0 \sim_G \alpha_5 \sim_G \alpha_7 \sim_G \alpha_8$, $\alpha_2 \sim_G \alpha_3$, and $\alpha_4 \sim_G \alpha_6$. For $G = \{a_2\}$, $\alpha_0 \sim_G \alpha_1 \sim_G \alpha_3 \sim_G \alpha_4$, $\alpha_2 \sim_G \alpha_8$, and $\alpha_6 \sim_G \alpha_7$. Finally, for any combination of two agents from $\mathcal{N}$,* i.e. *$\{a_0, a_1\}$, $\{a_0, a_2\}$ and $\{a_1, a_2\}$ as well as for $\mathcal{N}$, no equivalence of this type exists.*

We say that two allocations α_1, α_2 are *collectively equivalent* with respect to a group of agents $G \subseteq \mathcal{N}$ if the allocation to the group G under α_1 is the same as its allocation under α_2. Note that with this notion, we are not concerned with how goods are allocated *within* the group G, only with the set of goods that are allocated to G. We denote the fact that α_1 and α_2 are collectively equivalent with respect to $G \subseteq \mathcal{N}$ by $\alpha_1 \approx_G \alpha_2$. Formally:

Definition 2 (Collective equivalence w.r.t. G)

$$\alpha_1 \approx_G \alpha_2 \quad \textit{iff} \quad \bigcup_{a_i \in G} \alpha_1(a_i) = \bigcup_{a_i \in G} \alpha_2(a_i).$$

Example 4 *Example 1 again. For $G = \{a_0, a_1\}$, we have $\alpha_0 \approx_G \alpha_1 \approx_G \alpha_3 \approx_G \alpha_4$, $\alpha_2 \approx_G \alpha_8$, and $\alpha_6 \approx_G \alpha_7$.*

For singleton groups $G = \{a_i\}$, individual and collective equivalence coincide:

$$\forall a_i \in \mathcal{N}, \forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_{\{a_i\}} \alpha_2 \quad \text{iff} \quad \alpha_1 \approx_{\{a_i\}} \alpha_2.$$

Furthermore, all allocations are collectively equivalent w.r.t. $G = \mathcal{N}$:

$$\forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \approx_{\mathcal{N}} \alpha_2.$$

We can also define individual and collective equivalence with respect to sets of *goods*, rather than with respect to sets of *agents*. We say two allocations α_1 and α_2 are individually equivalent with respect to a set of goods $Z \subseteq \mathcal{Z}$ if these goods are allocated to the same agents in both α_1 and α_2. With a slight abuse of notation, we write $\alpha_1 \sim_Z \alpha_2$ to mean that goods $Z \subseteq \mathcal{Z}$ are allocated to the same agents in α_1 as in α_2. Formally:

Definition 3 (Individual equivalence w.r.t. Z)

$$\alpha_1 \sim_Z \alpha_2 \quad \textit{iff} \quad \forall z \in Z : \hat{\alpha_1}(z) = \hat{\alpha_2}(z).$$

Example 5 *With respect to Example 1, if $Z = \{z_1\}$ then we have: $\alpha_0 \sim_Z \alpha_4 \sim_Z \alpha_8$, $\alpha_1 \sim_Z \alpha_2 \sim_Z \alpha_3$, and $\alpha_5 \sim_Z \alpha_6 \sim_Z \alpha_7$,*

With another abuse of notation, we write $\alpha_1 \approx_Z \alpha_2$ to mean that α_1 and α_2 agree on the group of agents that receives goods $Z \subseteq \mathcal{Z}$:

Definition 4 (Collective equivalence w.r.t. Z)

$$\alpha_1 \approx_Z \alpha_2 \;\; \textit{iff} \;\; \{\hat{\alpha_1}(z) \mid z \in Z\} = \{\hat{\alpha_2}(z) \mid z \in Z\}.$$

Again, for singleton sets of goods $Z = \{z\}$, individual and collective equivalence w.r.t. Z coincide:

$$\forall z \in \mathcal{Z}, \forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_{\{z\}} \alpha_2 \;\; \text{iff} \;\; \alpha_1 \approx_{\{z\}} \alpha_2.$$

Example 6 *With respect to Example 1, if $Z = \{z_1, z_2\}$ then we have for example $\alpha_2 \approx_Z \alpha_6$ and $\alpha_7 \approx_Z \alpha_8$.*

Valuation Function Structure: It makes sense to classify the various different types of externality that may exist in our setting. We do this with reference to the properties of allocations that were discussed above. Later in this paper, we will consider the computational problem of classifying valuation functions with respect to these properties.

We start with the simplest case: *no* externalities! We say that a valuation $v_i : \mathcal{A} \to \mathbb{R}$ for agent $a_i \in \mathcal{N}$ is *externality free* if it only depends on the goods allocated to agent a_i. Formally:

Definition 5 (Externality Freeness) *Valuation function $v_i : \mathcal{A} \to \mathbb{R}$ is said to be* externality free *iff:*

$$\forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_{\{a_i\}} \alpha_2 \;\; \textit{implies} \;\; v_i(\alpha_1) = v_i(\alpha_2).$$

Example 7 *Consider the auction defined in Examples 1 and 2. Valuation function v_0 is trivially free of externalities, since it gives 0 for all allocations. However, v_1 is subject to externalities. For example, although $\alpha_4 \sim_{a_1} \alpha_6$, we have $v_1(\alpha_4) \neq v_1(\alpha_6)$: in this case, if a_1 is allocated good z_2, then it would prefer a_2 not to be allocated z_1.*

Of course, the point of the paper is to consider valuation functions that do not have this property. The next property we consider is that externalities take a very simple structural form, where a valuation function can be additively decomposed into a collection of simpler valuation functions.

Definition 6 (Additively decomposable valuations) *Formally, $v_i : \mathcal{A} \to \mathbb{R}$ is said to be* additively decomposable *if there exists a collection of n functions $\{v_i^n : \mathbf{2}^{\mathcal{Z}} \to \mathbb{R} : i \in \mathcal{N}\}$*

$$\begin{array}{l} v_i^1 : \mathbf{2}^{\mathcal{Z}} \to \mathbb{R} \\ v_i^2 : \mathbf{2}^{\mathcal{Z}} \to \mathbb{R} \\ \cdots \\ v_i^n : \mathbf{2}^{\mathcal{Z}} \to \mathbb{R} \end{array}$$

such that $\forall \alpha \in \mathcal{A}$ we have:

$$v_i(\alpha) = \sum_{a_j \in \mathcal{N}} v_i^j(\alpha(a_j)).$$

An even simpler kind of valuation function is as follows.

Definition 7 (Primitively decomposable valuations) *A valuation function $v_i : \mathcal{A} \to \mathbb{R}$ is said to be* primitively decomposable *if there exists an $m \times n$ matrix M_i of real numbers such that $\forall \alpha \in \mathcal{A}$:*

$$v_i(\alpha) = \sum_{a_j \in \mathcal{N}} \sum_{z_k \in \alpha(a_j)} M_i[k,j].$$

Thus, with primitively decomposable valuation functions, $M_i[k,j]$ represents the value that agent a_i would obtain if good z_k was allocated to agent a_j. Notice that primitively decomposable valuation functions have a very succinct representation: we only need to represent an $m \times n$ matrix of reals.

Both of the above possibilities, of course, represent somewhat extremal cases. We therefore consider other types of externality, as follows. First, suppose that an agent only cares about the goods received by a *particular group of agents*; the particular allocation of goods to agents *outside* this group or *within it* are not a concern. (Externality freeness is a special case where the group of agents in question is a singleton consisting of the agent itself.) There are two further obvious possibilities here. The first is that the valuation function is sensitive to the allocation of goods to players within the group, while the second is that the valuation function is only sensitive to the goods that are allocated to the group as a whole (i.e., is not sensitive to which agents within the group receive which particular goods, only to the goods allocated to the group overall).

We then say that a valuation function v_i is *individually sensitive to $G \subseteq \mathcal{N}$* if the values it assigns only depend on the allocations of goods to individual member of G. Formally, individual sensitivity of v_i with respect to G is given by the following condition:

Definition 8 (Individual sensitivity w.r.t. G) *A valuation function v_i is said to be* individually sensitive *w.r.t. $G \subseteq \mathcal{N}$ iff:*

$$\forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_G \alpha_2 \text{ implies } v_i(\alpha_1) = v_i(\alpha_2).$$

A valuation function v_i is *collectively sensitive to $G \subseteq \mathcal{N}$* if the values it assigns only depends on the collective allocation of goods to G.

Definition 9 (Collective sensitivity w.r.t. G) *A valuation function v_i is said to be* collectively sensitive *w.r.t. G iff:*

$$\forall \alpha, \alpha' \in \mathcal{A} : \alpha \approx_G \alpha' \text{ implies } v_i(\alpha) = v_i(\alpha').$$

In a similar vein to the above, we might consider agents that are sensitive only to how a particular set of goods are allocated. We will say a valuation function v_i is *individually sensitive to goods $Z \subseteq \mathcal{Z}$* if the value of v_i only depends upon who is allocated goods in Z.

Definition 10 (Individual sensitivity w.r.t. Z) *A valuation function v_i is said to be* individually sensitive *w.r.t. $Z \subseteq \mathcal{Z}$ iff:*

$$\forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_Z \alpha_2 \text{ implies } v_i(\alpha_1) = v_i(\alpha_2).$$

And finally, we can think of agents who are concerned not with which individuals receive which goods, but which groups receive them.

Definition 11 (Collective sensitivity w.r.t. Z**)** *A valuation function* v_i *is said to be* collectively sensitive *w.r.t.* $Z \subseteq \mathcal{Z}$ *iff:*

$$\forall \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \approx_Z \alpha_2 \ \textit{implies} \ v_i(\alpha_1) = v_i(\alpha_2).$$

4 A Bidding Language

A common problem now arises: how to succinctly represent valuation functions $v_i : \mathcal{A} \to \mathbb{R}$. In the literature on bidding languages for combinatorial auctions [3, 19], a common approach is to allow a bidder to submit a number of atomic bids of the form (Z, p), where $Z \subseteq \mathcal{Z}$ and $p \in \mathbb{R}_+$. The semantics of a bid (Z, p) is that "I would be prepared to pay p to be allocated goods Z". An agent's atomic bids are aggregated into a valuation function using, for example, "OR" or "XOR" constructions [19]. This approach does not work for our scenarios, since an agent is not just concerned with goods allocated to himself, but also about how goods are allocated to others. We instead propose a *weighted rule* bidding language for combinatorial auctions with externalities, which derives inspiration from the *weighted formula* representations that have been used to represent preferences and valuation functions in other areas of AI (cf. [13, 17, 26, 6, 26]). With this approach, we specify agent a_i's valuation function v_i as a set of rules $\mathcal{R}_i$, with each rule taking the form $(\textit{condition}, \textit{value})$, where *condition* is a logical predicate over allocations $\mathcal{A}$, and $\textit{value} \in \mathbb{R}_+$. To obtain the value of an allocation α given a set of rules $\mathcal{R}$, we sum the values x of all the rules (φ, x) in $\mathcal{R}$ whose condition φ is satisfied by α.

A Language for Conditions: First, we define a language for the conditions of our rules. The condition language is essentially that of conventional propositional logic, except that primitive propositions are replaced with expressions for referring to allocations. These expressions are of the form $a_i : z$, where $a_i \in \mathcal{N}$ is an agent and $z \in \mathcal{Z}$ is a good. The intended interpretation of the expression $a_i : z$ is, naturally enough, that agent a_i is allocated good z. We refer to an expression of the form $a_i : z$ as an *atomic allocation*. To obtain the condition language, we allow atomic allocations to be combined with the conventional connectives of classical Boolean logic $(\neg, \vee)$. Formally, the grammar for conditions is as follows:

$$\varphi ::= a_i : z \mid \neg\varphi \mid \varphi \vee \varphi$$

where $a_i \in \mathcal{N}$ and $z \in \mathcal{Z}$. The other connectives of classical Boolean logic ($\wedge$ – "and", $\to$ – "implies", $\leftrightarrow$ – "if, and only if", etc.), may be defined as abbreviations in terms of $\neg, \vee$ in the conventional manner (e.g., $\varphi \to \psi \equiv (\neg\varphi \vee \psi)$).

Where $\alpha \in \mathcal{A}$ is an allocation and φ is a condition, we write $\alpha \models \varphi$ to mean that the allocation α satisfies condition φ. Formally, the relation $\models$ is inductively defined as follows:

$\alpha \models a_i : z$ iff $z \in \alpha(a_i)$;

$\alpha \models \neg\varphi$ iff it is not the case that $\alpha \models \varphi$;

$\alpha \models \varphi \vee \psi$ iff $\alpha \models \varphi$ or $\alpha \models \psi$.

Where φ is a condition in this language, let $\mathcal{N}(\varphi)$ denote the set of agents named in φ, and let $\mathcal{Z}(\varphi)$ denote the set of goods named in φ. For example, if $\varphi = (a_1 : z_3) \wedge (a_4 : z_7)$ then $\mathcal{N}(\varphi) = \{a_1, a_4\}$ and $\mathcal{Z}(\varphi) = \{z_3, z_7\}$.

Rules: A *rule* is a pair (φ, x) where φ is a condition and $x \in \mathbb{R}_+$ is a real. A set of rules $\mathcal{R}$ defines a valuation function $v_\mathcal{R}$ as follows:

$$v_\mathcal{R}(\alpha) = \sum_{(\varphi_i, x_i) \in \mathcal{R} : \alpha \models \varphi_i} x_i$$

If all rules in $\mathcal{R}$ have equal weight, then we say that $\mathcal{R}$ is *homogeneous*.

We refer to our representation as the *weighted rule* representation.

4.1 Basic Properties of Weighted Rules

We first establish basic properties of weighted rule representation:

Theorem 1

1. *The weighted rule representation can capture all valuation functions. More precisely, $\forall v \in \mathcal{V}$, $\exists \mathcal{R}$ such that $v = v_\mathcal{R}$.*

2. *For some classes of valuation functions, the weighted rule representation is exponentially more succinct than the explicit representation.*

3. *For some classes of valuation function, the smallest weighted rule representation requires a number of rules that is exponential in $|\mathcal{N} \cup \mathcal{Z}|$.*

Proof 1 *For item (1), it suffices to note that given an allocation α we can define a "canonical" condition φ_α, such that $\alpha' \models \varphi_\alpha$ iff $\alpha' = \alpha$:*

$$\varphi_\alpha = \bigwedge_{a_i \in \mathcal{N}} \left(\bigwedge_{z \in \alpha(a_i)} a_i : z \right)$$

Items (2) and (3) follow easily from well-known results in Boolean function theory: essentially, Boolean formulae provide a representation that in many cases is exponentially more succinct than the extensive representation, but in the worst case we need formulae of size exponential in the number of Boolean variables [27].

Now, although our condition language borrows much from Boolean logic, it is important to understand that the logic of conditions is rather different. To see this, consider the following. Let us say a condition is *positive* if it contains only the Boolean operators $\vee, \wedge$. Negation cannot be defined in classical Boolean logic using only these operators, and as a consequence, the satisfiability problem for positive Boolean formulae is trivially solvable in polynomial time. However:

Theorem 2 *The satisfiability problem for positive conditions is* NP*-complete.*

Proof 2 *Membership is obvious; for hardness we reduce* SAT*. Let φ be an instance of* SAT *over Boolean variables $x_1, \ldots, x_k$, which we assume w.l.o.g. is in* CNF*. We obtain a condition φ^* from φ by systematically substituting for each positive literal x that occurs in φ the atomic allocation $a_\top : x$, and for each negative literal $\neg x$ that occurs in φ the atomic allocation $a_\perp : x$. Notice that φ^* is a positive condition. Then define $\mathcal{N}_\varphi = \{a_\top, a_\perp\}$ and $\mathcal{Z}_\varphi = \{x_1, \ldots, x_k\}$. Now, there exists an allocation $\alpha \in \mathcal{A}(\mathcal{Z}_\varphi, \mathcal{N}_\varphi)$ such that $\alpha \models \varphi^*$ iff φ is satisfiable.*

Note that this result does *not* imply that negations in our condition language can be defined using positive conditions! Theorem 2 raises an interesting question, namely, for *what classes of condition is the satisfiability problem easy*. We now identify one such class. Let us say a formula φ is *read once* if no good is named more than once in φ [1]. For example, the formula $(a_0 : z_0) \wedge \neg(a_0 : z_1)$ is read-once, while the formula $(a_0 : z_0) \vee (a_1 : z_0)$ is not (since z_0 is named twice). Then the following is easy:

Theorem 3 *All positive read-once conditions are satisfiable, and moreover, it is possible to compute a satisfying allocation for a positive read-once formula in polynomial time.*

The proof is simple, but is used later, and so we state it in full.

Proof 3 *Construct the parse tree of condition φ: interior nodes in the tree will be either $\wedge$ or $\vee$, while leaves will be atomic allocations. Let s_0 be the root of the tree. We construct a function L which maps nodes of the tree to sets of atomic allocations. We start with leaves. If a leaf node s is $a_i : z$ then define $L(s) = \{a_i : z\}$. We then iteratively repeat the following, until L is defined for all nodes in the parse tree: For each interior node s such that L is defined for all of s's children, then define $L(s)$ as follows: if s is an $\vee$ node, then define $L(s)$ to be the smallest size set $L(s')$ such that s' is a child of s (if there are multiple such children, pick the one that is leftmost in the parse tree); if s is an $\wedge$ node, then define $L(s) = L(s_1) \cup \cdots \cup L(s_k)$ where $s_1, \ldots, s_k$ are the children of s. When L labels all states, define allocation α_φ by $\alpha_\varphi(a_i) = \{z \mid a_i : z \in L(s_0)\}$. Note that since φ is read once, every good is allocated to at most one agent, so α_φ is well defined. Then $\alpha_\varphi \models \varphi$.*

4.2 Classifying Valuation Functions

In Section 3, we gave a preliminary classification scheme for valuation functions. Now that we have a representation for valuations, it is interesting to ask how hard it is to classify valuation functions represented using this scheme. We start with the decision problem EXTERNALITY FREENESS, where we are given a rule set $\mathcal{R}$ (over $\mathcal{Z}, \mathcal{N}$), and we are asked whether $v_\mathcal{R}$ is free of externalities.

[1] Read-once formulas have been studied for combinatorial auctions with no externalities both from a preference elicitation and winner determination perspective in [28]. Note that our usage differs slightly from the regular use of the term "read once" in logic, where it is usually taken to mean that no *Boolean variable* occurs more than once in a formula.

Theorem 4 EXTERNALITY FREENESS *is co-*NP*-complete.*

Proof 4 *Membership is obvious. For hardness, we show that the complement is* NP*-hard by reduction from* SAT. *The complement problem involves checking whether:* $\exists \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_{a_i} \alpha_2$ *and* $v_i(\alpha_1) \neq v_i(\alpha_2)$. *Let* φ *be a* SAT *instance over variables* $x_1, \ldots, x_k$ *which we assume w.l.o.g. is in* CNF. *Define* $\mathcal{N} = \{a_\top, a_\bot, a_d\}$, $\mathcal{Z} = \{x_1, \ldots, x_k, d\}$, *and define condition* φ' *by replacing positive literals* x_i *in* φ *by* $(a_\top : x_i)$ *and negative literals* $\neg x_i$ *by* $\neg(a_\top : x_i)$. *Then define* $\varphi^* = \varphi' \wedge (a_d : d)$, *and define* $\mathcal{R} = \{(\varphi^*, 1)\}$. *It is easy to see that then* φ *is satisfiable iff* $v_\mathcal{R}$ *is not externality free.*

We can also consider the decision problems: INDIVIDUAL SENSITIVITY W.R.T. G, COLLECTIVE SENSITIVITY W.R.T. G, INDIVIDUAL SENSITIVITY W.R.T. Z, and COLLECTIVE SENSITIVITY W.R.T. Z. For example, in the problem INDIVIDUAL SENSITIVITY W.R.T. G, we are given a rule set $\mathcal{R}$ (over $\mathcal{Z}, \mathcal{N}$), and a set of agents $G \subseteq \mathcal{N}$, and we are asked whether $v_\mathcal{R}$ is individually sensitive w.r.t. G; the other problems are formulated in the obvious way. Using Theorem 4, we can directly prove:

Theorem 5 *The decision problems* INDIVIDUAL SENSITIVITY W.R.T. G, COLLECTIVE SENSITIVITY W.R.T. G, INDIVIDUAL SENSITIVITY W.R.T. Z, *and* COLLECTIVE SENSITIVITY W.R.T. Z *are all co-*NP*-complete.*

Proof 5 *The first two results follow from Theorem 4; the others can be obtained by similar reductions, which we omit in the interests of space restrictions. We show that the complement problem is* NP*-hard by reduction from* SAT. *In the complement problem, we are asked whether:*

$$\exists \alpha_1, \alpha_2 \in \mathcal{A} : \alpha_1 \sim_G \alpha_2 \text{ and } v_i(\alpha_1) \neq v_i(\alpha_2).$$

Let φ *be an instance of* SAT *over Boolean variables* $x_1, \ldots, x_k$, *which we assume w.l.o.g. is in* CNF. *We create an instance of* INDIVIDUAL SENSITIVITY W.R.T. G *as follows. Define* $\mathcal{N} = \{a_\top, a_\bot, a_d, a_e\}$, *and* $\mathcal{Z} = \{x_1, \ldots, x_k, d\}$. *Define* φ^* *using the same transformation as used in the proof of Theorem 2, and define* $\psi = \varphi^* \wedge (a_d : d)$. *Define* $\mathcal{R} = \{(\psi, 1)\}$ *and* $G = \{a_\top, a_\bot\}$. *We claim that* φ *is satisfiable iff the instance created is not individually sensitive to* G. *(*$\rightarrow$*) Assume* φ *is satisfiable; then we can construct an allocation* α_1 *yielding* $v_\mathcal{R}(\alpha) = 1$ *in which* a_d *is allocated good* d, *but there is another allocation differing only in the allocation of* d *(to* a_e*) in which* $v_\mathcal{R}(\alpha_2) = 0$. *Hence the instance is a positive instance of the problem. (*$\leftarrow$*) Assume the instance is individually sensitive to* G. *Then there exist valuations* α_1, α_2 *such that* $v_\mathcal{R}(\alpha_1) = 1$ *and* $v_\mathcal{R}(\alpha_2) = 0$. *Since* $v_\mathcal{R}(\alpha_1) = 1$, *then* $\alpha_1 \models \psi$, *which implies* $\alpha_1 \models \varphi^*$, *and so from* α_1 *we can construct a satisfying assignment for* φ.

5 Winner Determination

Let us now turn to the WINNER DETERMINATION problem for the weighted rule representation. An instance of the problem for this representation will be a tuple

$\langle \mathcal{Z}, \mathcal{N}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$, where for each $a_i \in \mathcal{N}$, $\mathcal{R}_i$ is a rule set defining a_i's valuation function.

Complexity: We start by establishing some results on the complexity of the WINNER DETERMINATION problem assuming the weighted rule representation of valuation functions:

Theorem 6

1. *The* WINNER DETERMINATION *problem for the weighted rule representation is* FP^{NP}*-complete.*
2. *The* WINNER DETERMINATION *problem for homogeneous rule sets is* $\text{FP}^{\text{NP}[\log|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|]}$*-complete.*

Proof 6 *For (1), first notice that the associated decision problem (does there exist an allocation α^* with social welfare at least k) is easily seen to be* NP*-complete from Theorem 2. This implies membership in* FP^{NP}*, since all optimization problems whose decision problem is in* NP *are in* FP^{NP} *[21, p.416]. For hardness, we reduce the optimization problem* MAX WEIGHT SAT *[21, p.416]. An instance of* MAX WEIGHT SAT *is given by a set of propositional clauses $\psi_1, \ldots, \psi_a$, over Boolean variables $x_1, \ldots, x_b$, together with integer weights $w_1, \ldots, w_a$ for each clause. The aim is to find the valuation that maximises the sum of weights of clauses satisfied by the valuation. As in the* SAT *reduction of Theorem 2, we create two agents $\mathcal{N} = \{a_\top, a_\bot\}$. For each clause ψ_i with weight w_i, we create a rule (ψ_i^*, w_i) in $\mathcal{R}_\top$, where ψ^* is obtained by the same transformation on Boolean formulae that we used in the reduction of Theorem 2: replace positive literals x with $a_\top : x$ and negative literals $\neg x$ with $a_\bot : x$. Set $\mathcal{R}_\bot = \emptyset$. Any social welfare maximising allocation α^* for this problem will yield a solution to the given* MAX WEIGHT SAT *problem: set a variable x to true if α^* allocates it to $a_\top$, and set it false if α^* allocates it to $a_\bot$. Notice that the form of the instance constructed matches the statement of the theorem. For (2), note that if all rules have equal weights, then an allocation that maximises social welfare will be one that maximises the number of rules with conditions satisfied. Next, notice that the following problem is trivially seen to be* NP*-complete: "given $\langle \mathcal{N}, \mathcal{Z}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ and $k \in \mathbb{N}$, does there exist an allocation satisfying at least k rules?" Now, there are $|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$ rules in total, so we can find the allocation that maximises the number of satisfied rules by using binary search, requiring $O(\log|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|)$ queries to an* NP*-oracle for the problem we just described. Our first query will set $k = \lceil \frac{|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|+1}{2} \rceil$; if the answer is "no" we query with $k = \lceil \frac{|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|+1}{4} \rceil$, while if the answer is "yes" then we query with $k = \lceil \frac{3|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|+1}{4} \rceil$, and so on; we converge on the correct value in $O(\log|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|)$ queries. Hence the problem is in* $\text{FP}^{\text{NP}[\log|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|]}$*. For hardness, we can reduce* MAX SAT *problem [21, p.186]; the construction is essentially identical to that used in item (1), with all rule weights set to* 1.

Exact Winner Determination: *Integer Linear Programming* is one of the most successful and widely-used practical approaches to solving computationally complex optimization problems [2, pp.65–67]. We now show how the winner determination problem for our weighted rule representation can be solved by ILPs. First, some definitions are needed. Where φ is a condition, then we denote the set of *sub-formulae* of φ by $\mathit{sf}(\varphi)$:

$$\mathit{sf}(\varphi) = \begin{cases} \{\psi, \chi\} \cup \mathit{sf}(\psi) \cup \mathit{sf}(\chi) & \text{if } \varphi = \psi \vee \chi \text{ or } \varphi = \psi \wedge \chi \\ \{\psi\} \cup \mathit{sf}(\psi) & \text{if } \varphi = \neg\psi \\ \{a_i : z\} & \text{if } \varphi = a_i : z,\ a_i \in \mathcal{N},\ z \in \mathcal{Z}. \end{cases}$$

Where $\mathcal{R} = \{(\varphi_1, x_1), \ldots, (\varphi_k, x_k)\}$ is a set of rules, we define the set $\mathit{cl}(\mathcal{R})$ by:

$$\mathit{cl}(\mathcal{R}) = \bigcup_{(\varphi_i, x_i) \in \mathcal{R}} \mathit{sf}(\varphi_i).$$

Let $\langle \mathcal{Z}, \mathcal{N}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ be an instance of the WINNER DETERMINATION problem using the weighted rule representation: we produce an ILP as shown in Figure 1. The construction makes use of two sets of variables: for each $a_i \in \mathcal{N}$ and $z \in \mathcal{Z}$, a variable $\mathit{alloc}(a_i, z) \in \{0, 1\}$, used to indicate whether good z is allocated to agent a_i in the optimal allocation computed by the ILP ($\mathit{alloc}(a_i : z) = 1$) or not ($\mathit{alloc}(a_i : z) = 0$); and for each $\psi \in \mathit{cl}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n)$, a variable $\tau(\psi) \in \{0, 1\}$, used to indicate whether the condition ψ is satisfied by the optimal allocation computed by the ILP ($\tau(\psi) = 1$), or not ($\tau(\psi) = 0$).

Notice that the construction yields a polynomial number of constraints.

Theorem 7 *Any allocation defined by a solution to the* ILP *in Figure 1 for input instance $\langle \mathcal{Z}, \mathcal{N}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$ is a solution to the* WINNER DETERMINATION *problem for $\langle \mathcal{Z}, \mathcal{N}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$.*

Proof 7 *First notice that any solution does indeed define an allocation, in that it allocates every good to exactly one agent by constraint (4). Next, let α^* be an allocation defined by a solution σ to the* ILP*: we claim that $\forall \varphi \in \mathit{cl}(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n)$, $\alpha^* \models \varphi$ iff $\tau(\varphi) = 1$ in the solution σ. The proof is by induction on the structure of φ. The inductive base is for atomic allocations $a_i : z$, and is given by (5). Now assume the result is proved for strict sub-formulae of φ. For the inductive step, we reason by cases: the case where φ is of the form $\varphi = \neg\psi$ is given by constraint (6); $\varphi = \psi \vee \chi$ is given by constraints (7)–(9); and $\varphi = \psi \wedge \chi$ is given by constraints (10)–(12). Finally, the objective function (1) ensures the allocation is optimal. Note that the only unknowns are variables $\mathit{alloc}(a_i, z)$; all variables $\tau(\psi)$ are dependent. Finally, note that the values x_i in the objective function (1) are constants.*

Approximate Winner Determination: The ILP framework above of course has worst case running time exponential in the size of the auction. This raises the question of whether it is possible to find a polynomial time *approximation algorithm* for winner determination, i.e., an polynomial time algorithm that is guaranteed to compute an allocation with some guarantee of performance [2]. We start with a negative result.

Theorem 8 *The* WINNER DETERMINATION *problem with the weighted rule representation cannot be approximated in polynomial time within any approximation ratio* maximize:

$$\sum_{(\varphi_i, x_i) \in (\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n)} \tau(\varphi_i) \cdot x_i \tag{1}$$

subject to constraints:

$$\tau(\psi, s) \in \{0, 1\} \quad \text{for all } \psi \in cl(\mathcal{R}_1 \cup \cdots \mathcal{R}_n) \tag{2}$$

$$alloc(a_i, z) \in \{0, 1\} \quad \text{for all } a_i \in \mathcal{N}, z \in \mathcal{Z} \tag{3}$$

$$\sum_{a_i \in \mathcal{N}} alloc(a_i, z) = 1 \quad \text{for all } z \in \mathcal{Z} \tag{4}$$

$$\tau(a_i : z) = alloc(a_i, z) \quad \text{for all } a_i : z \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{5}$$

$$\tau(\neg\psi) = 1 - \tau(\psi) \quad \text{for all } \neg\psi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{6}$$

$$\tau(\psi \vee \chi) \leq \tau(\psi) + \tau(\chi) \quad \text{for all } \psi \vee \chi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{7}$$

$$\tau(\psi \vee \chi) \geq \tau(\psi) \quad \text{for all } \psi \vee \chi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{8}$$

$$\tau(\psi \vee \chi) \geq \tau(\chi) \quad \text{for all } \psi \vee \chi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{9}$$

$$\tau(\psi \wedge \chi) \leq \tau(\psi) \quad \text{for all } \psi \wedge \chi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{10}$$

$$\tau(\psi \wedge \chi) \leq \tau(\chi) \quad \text{for all } \psi \wedge \chi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{11}$$

$$\tau(\psi \wedge \chi) \geq 1 - ((1 - \tau(\psi)) + (1 - \tau(\chi))) \quad \text{for all } \psi \wedge \chi \in cl(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n) \tag{12}$$

Figure 1: ILP for winner determination.

$r(m)$, where $r(\cdot)$ is any polynomial time computable function, unless P = NP. *This claim holds even for problem instances with only 2 agents, and with conditions which are only positive formulae, and with the weights in the weighted rule representation assumed to all have value 1.*

Proof 8 *We use the reduction from the proof of Theorem 2. We encode any instance of* SAT *by a polynomial size instance of the* WINNER DETERMINATION *problem such that deciding an existence of a feasible solution to the* WINNER DETERMINATION *problem with positive social welfare is equivalent to deciding if the given* SAT *instance is a "Yes" instance.*

Given an instance φ of SAT *over Boolean variables $x_1, \ldots, x_m$, let φ^* be the positive condition obtained as in the proof of Theorem 2, and $\mathcal{N}_\varphi = \{a_\top, a_\bot\}$.*

We define the following instance of the WINNER DETERMINATION *problem. The set of goods is $\mathcal{Z} = \{x_1, \ldots, x_m\}$, and the set of agents is $\mathcal{N} = \{a_\top, a_\bot\}$, and both agents have the same set of rules, containing just one rule, $\mathcal{R}_{a_\top} = \mathcal{R}_{a_\bot} = \{(\varphi^*, 1)\}$.*

Suppose now that we are given a polynomial time $r(m)$-approximation algorithm A for the WINNER DETERMINATION *problem. We consider two cases. Suppose first that the given* SAT *instance is a "Yes" instance, i.e., there exists an allocation satisfying condition φ^*. This allocation has social welfare 2, and we can assume that any other allocation (which does not satisfy φ^*) has social welfare 0. Thus, because algorithm A has a finite approximation ratio, in this case, A has to output an allocation which satisfies φ^*.*

Suppose now that the given SAT *instance is a "No" instance, i.e., there does not exist any satisfying allocation. Therefore, any allocation has in this case a social welfare of zero.*

We have thus argued that algorithm A can distinguish in polynomial time "Yes" and "No" instances of the SAT *problem, which would imply that* P = NP.

Theorem 8 essentially establishes that there is no hope for a polynomial time approximation algorithm with any performance guarantees (even as bad as $2^{poly(m)}$, where $poly(m)$ is a fixed polynomial of m) which works on all cases (unless P = NP). However, this does not imply that we cannot identify certain classes of problem instances that can be approximated. To illustrate this, we give an approximation method for instances with positive read-once rules.

We first prove that the WINNER DETERMINATION problem with only positive read-once rules still remains hard to approximate, but, on the other hand we will show that it is much more tractable now.

Two allocations $\alpha_1 : \mathcal{N} \to \mathbf{2}^{\mathcal{Z}_1}$ and $\alpha_2 : \mathcal{N} \to \mathbf{2}^{\mathcal{Z}_2}$ are *compatible* if for all $z \in \mathcal{Z}_1 \cap \mathcal{Z}_2$, we have $\hat{\alpha_1}(z) = \hat{\alpha_2}(z)$. Given a set of positive read-once conditions $C = \{\varphi_1, \ldots, \varphi_k\}$, we say that a given allocation α, such that $\alpha \models \varphi$ for some $\varphi \in C$, *collides* with a condition $\varphi' \in C \setminus \{\varphi\}$ if there exists an allocation α' such that $\alpha' \models \varphi'$ and allocations α and α' are not compatible.

Theorem 9 *Consider* WINNER DETERMINATION *with n agents and m goods, where each agent has only one positive read-once rule with weight 1, and assume that for any condition, any allocation that satisfies this condition collides with at most $\Delta \in \mathbb{N}$ other* *conditions in the instance. Assume also that for any given condition, any satisfying allocation for this condition allocates at most $d \in \mathbb{N}$ goods to the agents.*

This problem cannot be approximated in polynomial time within an approximation ratio of:

1. *$n^{1-\delta}$, for any constant $\delta > 0$,*
2. *Δ^{ε}, for some absolute constant $\varepsilon > 0$,*
3. *$O(d/\log(d))$,*
4. *$m^{1/2-\delta}$, for any constant $\delta > 0$,*

unless P = NP.

In the proof of this theorem, we slightly relax the definition of the WINNER DETERMINATION problem, in that we do not require anymore that a feasible solution must allocate all the goods, which corresponds now to the fact that we show lower bounds and we consider this problem as an optimization, and not necessarily decision, problem.

Proof 9 *We present a polynomial approximability-preserving reduction from a well known d-*SET PACKING *problem [7]. The problem is given an universe U of $m \in \mathbb{N}$ elements and a family $\mathcal{S} = \{S_1, \dots, S_n\} \subseteq 2^U$ of n subsets of U, each of size at most d (i.e., $|S_i| \leq d$, for any $S_i \in \mathcal{S}$), compute a subfamily $\mathcal{S}' \subseteq \mathcal{S}$ of pairwise disjoint subsets (called a* packing*) of maximum size, i.e., such that $|\mathcal{S}'|$ is maximized.*

*Given an instance of d-*SET PACKING *we define the following instance of the* WINNER DETERMINATION *problem. The set of goods is $\mathcal{Z} = U$ and the set of agents is $\mathcal{N} = \{1, \dots, n\}$. We define the following positive read-once formula: $\varphi_i = \left(\bigwedge_{z \in S_i} (i : z)\right)$, which is true if all goods from set S_i are assigned to agent i and, possibly, some other goods outside his set as well. Each agent $i \in \{1, \dots, n\}$ has the following rule $\mathcal{R}_i = \{(\varphi_i, 1)\}$. It is straightforward to argue now that if we are given a set packing of size k, then we can easily obtain an allocation with social welfare of k. And, conversely, if we are given an allocation of social welfare k, then we can easily obtain a set packing of size k. Observe, however, that in this case a feasible allocation may also allocate to an agent goods outside his set, but then we can always repair the solution by simply declaring these goods as not being allocated to this agent. This does not change the size of the packing. d-*SET PACKING *is known to be* NP*-hard to approximate within $O(d/\log(d))$ [12], which implies claim 3. The remaining claims follow from a standard reduction of the d-*SET PACKING *problem to the* MAXIMUM INDEPENDENT SET *problem. We will outline this reduction here for completeness. Recall, that the* MAXIMUM INDEPENDENT SET *problem, given an undirected graph $G = (V, E)$, asks for a maximum size independent set $V' \subseteq V$ of G, i.e., a set V' such that no two vertices from V' are adjacent; formally, for any $u, v \in V'$, $(u, v) \notin E$. For a given instance of the d-*SET PACKING *problem, we define an instance $G = (V, E)$ of the* MAXIMUM INDEPENDENT SET *problem as follows: $V = \mathcal{S} = \{S_1, \dots, S_n\}$ and $E = \{(S_i, S_j) : i \neq j \wedge S_i, S_j \in V \wedge S_i \cap S_j \neq \emptyset\}$. (We note here, that we can assume very special instances of d-*SET PACKING *problem, in which each $e \in U$ appears in* *precisely two input sets $\{S_1, \ldots, S_n\}$.) Observe now, that the defined parameters of the* WINNER DETERMINATION *problem translate into the* MAXIMUM INDEPENDENT SET *instance as follows: $|V| = n$, $|E| = m$, and Δ is an upper bound on the maximum degree of graph G. The remaining claims follow now by the hardness results for the* MAXIMUM INDEPENDENT SET *problem in [29, 11] (claims 1 and 4), and in [1] (claim 2).*

We will now provide almost tight upper bounds (i.e., approximation algorithms) for the instances with positive read-once conditions, which are close to the lower bounds proven in the first two claims of Theorem 9.

Theorem 10 *Consider* WINNER DETERMINATION *with n agents and m goods, where each agent i's set of rules $\mathcal{R}_i$ has only positive read-once rules (possibly, $|\mathcal{R}_i| \geq 1$) with arbitrary weights, and assume that for any condition, any allocation that satisfies this condition collides with at most $\Delta \in \mathbb{N}$ other conditions in the instance. (Note, that we allow here for arbitrary externalities.) Then, there is a polynomial time Δ-approximation algorithm for this problem.*

Proof 10 *Let us first observe that a straightforward $|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$-approximation algorithm simply outputs any feasible allocation (obtained using the proof of Theorem 3) to the condition which has the largest weight, say w. Obviously, the optimum solution cannot have social welfare better than $w \cdot |\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$. Now, observe that $|\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$-approximation garantee becomes n for the instances from the proof of Theorem 9 which prove $n^{1-\delta}$-hardness of approximation. Also, note that $\Delta \leq |\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n|$, and thus we focus now on Δ-approximation algorithms. We now define a Δ-approximation algorithm. Let initially $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_n$ and allocation $\beta = \emptyset$. As a first step, choose a rule $(\varphi, w) \in \mathcal{R}$ with the largest weight w and let α be the allocation obtained for φ using Theorem 3. Update $\mathcal{R} := \mathcal{R} \setminus \{(\varphi, w)\}$ and $\beta :=$ (allocation β combined with α). We now "eliminate" all allocations from all rules $\mathcal{R}$ which collide with α as follows: for any atomic allocation $i : z$, $i \in \mathcal{N}$, $z \in \mathcal{Z}$, which appears in α, replace any occurrence of the atomic formulae $j : z$ with $j \in \mathcal{N} \setminus \{i\}$ in all conditions in $\mathcal{R}$, with special symbol $\mathbf{f}$ (for* forbidden*). (Note, that this can be done in linear time.) If after this operation there are any rules $(\varphi', w') \in \mathcal{R}$ where φ' has only $\mathbf{f}$'s as atomic formulae, then update $\mathcal{R} := \mathcal{R} \setminus \{(\varphi', w')\}$ for all such rules. Now, iteratively choose a rule $(\varphi, w) \in \mathcal{R}$ with the largest weight w in the current set $\mathcal{R}$. Now let α be the allocation obtained for φ using Theorem 3 with an additional constraint that partial allocations containing $\mathbf{f}$ are avoided. Update $\mathcal{R} := \mathcal{R} \setminus \{(\varphi, w)\}$, $\beta :=$ (allocation β combined with α). "Eliminate" all colliding allocations: for any atomic allocation $i : z$, $i \in \mathcal{N}$, $z \in \mathcal{Z}$, which appears in α, replace any occurrence of the atomic formulae $j : z$ with $j \in \mathcal{N} \setminus \{i\}$ in all conditions in $\mathcal{R}$, with symbol $\mathbf{f}$. If after this operation there are any rules $(\varphi', w') \in \mathcal{R}$ where φ' has only $\mathbf{f}$'s as atomic formulae, then update $\mathcal{R} := \mathcal{R} \setminus \{(\varphi', w')\}$ for all such rules. Repeat iteratively the steps described above until possible. Finally, output the current allocation β. Observe first, that despite the fact that this algorithm iterates potentially over sets of exponentially many allocations (each single condition may have exponentially many feasible allocations), its running time is strongly polynomial. This follows immediately from the fact that each iteration allocates at least one more good to some agent; thus the* *number of iterations is at most m. To see that its approximation guarantee is* Δ*, notice that each time it chooses the largest weight rule from the current set* $\mathcal{R}$ *(this rule can be satisfied as it must contain at least one non-*$\mathbf{f}$ *atomic subformula), and it eliminates at most* Δ *other* allocations *with smaller weights. Thus, in each such iteration, we gain at least* $1/\Delta$*-fraction of social welfare that any optimum allocation could gain.*

6 Conclusions & Related Work

While there is a large literature on auctions and combinatorial auctions in economics and computer science, relatively little work has considered externalities in auctions. In economics, Jehiel and Moldovanu is probably the most notable example of such work [14]; the authors distinguish between *allocative* externalities (where agents care about the allocation of items to others), and *informational* externalities (where the utility of an agent is affected by the information held by others). The main focus of their analysis is on mechanism design (e.g., some impossibility results for mechanism design with informational externalities). In computer science, Salek and Kempe consider a special case of a combinatorial auction with externalities in which every buyer is interested in only one specific bundle of goods [23]. For these single-minded auctions they derive sufficient conditions for a truthful allocation and propose an $\sqrt{m}$-approximation algorithm for maximizing social welfare. The algorithm is essentially tight unless P=NP. Conitzer and Sandholm [4] present a general formalization of domains with externalities, and investigate the complexity of various decision problems in this setting, for several kinds of externality. However, they derive most of the results under the assumption that the effect of one variable on an agents utility is independent of the effect of another variable on that agents utility. This disallows, in particular, the combinatorial auctions with externalities, unless there are no complementarities or substitutabilities among the items. More recently, a number of authors have focused on externalities in online advertising (non-combinatorial) auctions (see, e.g. [8, 16, 10, 9, 22]).

Mechanism design is an obvious issue for future work. There is much more work to be done on both exact and approximate winner determination for our bidding language. Finally, it would be interesting to look at alternative bidding languages, allowing more expressive forms of bids to be submitted — see, e.g., the discussion in [25].

7 Acknowledgments

Tomasz Michalak and Michael Wooldridge acknowledge support from the EPSRC (Engineering and Physical Sciences Research Council) under the project Market Based Control of Complex Computational Systems (GR/T10657/01). Piotr Krysta acknowledges support from the EPSRC (EP/F069502/1).

References

[1] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation*. Springer-Verlag: Berlin, Germany, 1999.

[3] C. Boutilier and H. H. Hoos. Bidding languages for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, Seattle, WA, 2001.

[4] V. Conitzer and T. Sandholm. Expressive negotiation in settings with externalities. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005)*, Pittsburgh, PA, 2005. Extended version to appear in *Journal of Computer and System Sciences*.

[5] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. The MIT Press: Cambridge, MA, 2006.

[6] E. Elkind, L. A. Goldberg, P. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. *Mathematical Logic Quarterly*, 55(4):362–376, 2009.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*. W. H. Freeman: New York, 1979.

[8] A. Ghosh and M. Mahdian. Externalities in online advertising. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 161–168, New York, NY, USA, 2008. ACM.

[9] A. Ghosh and A. Sayedi. Expressive auctions for externalities in online advertising. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 371–380, New York, NY, USA, 2010. ACM.

[10] I. Giotis and A.R. Karlin. On the equilibria and efficiency of the gsp mechanism in keyword auctions with externalities. In *WINE*, pages 629–638, 2008.

[11] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, pages 627–636, 1996.

[12] E. Hazan, M. Safra, and O. Schwartz. On the Complexity of Approximating k-Set-Packing. *Computational Complexity*, 15(1):20–39, 2006.

[13] S. Ieong and Y. Shoham. Marginal contribution nets: A compact representation scheme for coalitional games. In *Proceedings of the Sixth ACM Conference on Electronic Commerce (EC'05)*, Vancouver, Canada, 2005.

[14] P. Jehiel and B. Moldovanu. Allocative and informational externalities in auctions and related mechanisms, 2006.

[15] P. Jehiel, B. Moldovanu, and E. Stacchetti. How (not) to sell nuclear weapons. *American Economic Review*, 86(4):814–29, 1996.

[16] D. Kempe and M. Mahdian. A cascade model for externalities in sponsored search. In *WINE '08: Proceedings of the 4th International Workshop on Internet and Network Economics*, pages 585–596, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] J. Lang, U. Endriss, and Y. Chevaleyre. Expressive power of weighted propositional formulas for cardinal preference modelling. In *Proceedings of Knowledge Representation and Reasoning (KR 2006)*, Lake District, England, 2006.

[18] R. Müller. Tractable cases of the winner determination problem. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, pages 319–336. The MIT Press: Cambridge, MA, 2006.

[19] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, pages 215–213. The MIT Press: Cambridge, MA, 2006.

[20] N. Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 209–242. Cambridge University Press: Cambridge, England, 2007.

[21] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley: Reading, MA, 1994.

[22] D.H. Reiley, S.M. Li, and R.A. Lewis. Northern exposure: a field experiment measuring externalities between search advertisements. In *EC '10: Proceedings of the 11th ACM conference on Electronic commerce*, pages 297–304, New York, NY, USA, 2010. ACM.

[23] M. Salek and D. Kempe. Auctions for share-averse bidders. In *Proceedings of WINE 2008 (LNCS Volume 5385)*, pages 609–620. Springer-Verlag: Berlin, Germany, 2008.

[24] T. Sandholm. Optimal winner determination algorithms. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, pages 337–368. The MIT Press: Cambridge, MA, 2006.

[25] T. Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

[26] J. Uckelman, Y. Chevaleyre, U. Endriss, and J. Lang. Representing utility functions via weighted goals. *Mathematical Logic Quarterly*, 55(4):341–361, 2009.

[27] I. Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons, 1987.

[28] M. A. Zinkevich, A. Blum, and T. Sandholm. On polynomial-time preference elicitation with value queries. In *In Proceedings of the ACM Conference on Electronic Commerce (ACM-EC*, pages 176–185. ACM-EC, 2003.

[29] David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3:103–128, 2007.