Mechanism Design with Partial Revelation

Nathanaël Hyafil and **Craig Boutilier**
{nhyafil,cebly}@cs.toronto.edu
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3H5, CANADA

Abstract

Classic direct mechanisms require full utility revelation from agents, which can be very difficult in practical multi-attribute settings. In this work, we study partial revelation within the framework of one-shot mechanisms. Each agent's type space is partitioned into a finite set of partial types and agents (should) report the partial type within which their full type lies. A classic result implies that implementation in dominant strategies is impossible in this model. We first show that a relaxation to Bayes-Nash implementation does not circumvent the problem. We then propose a class of partial revelation mechanisms that achieve *approximate dominant strategy implementation*, and describe a computationally tractable algorithm for myopically optimizing the partitioning of each agent's type space to reduce manipulability and social welfare loss. This allows for the automated design of one-shot partial revelation mechanisms with worst-case guarantees on both manipulability and efficiency.

1 Introduction

An important challenge facing AI is the design of protocols through which self-interested agents might interact to achieve some desirable outcome (such as negotiating an outcome that maximizes social welfare). As a consequence, *mechanism design* [17]—which studies precisely this problem from an economic and game-theoretic point of view—has become an important area of study within AI and computer science more broadly. Roughly speaking, a mechanism is a game intended to implement some social choice function (SCF), i.e., a function that selects some outcome as a function of the preferences of the participating agents.

A key result in mechanism design, the *revelation principle*, states that mechanisms can be restricted to *incentive compatible, direct mechanisms*, in which agents fully reveal their true *type* (i.e., utility function over outcomes). For instance, Vickrey-Clarke-Groves (VCG) is such a mechanism for social welfare maximization.

Unfortunately, direct type revelation is problematic in practice, since utility functions can be extremely difficult for agents to even compute effectively or communicate to the mechanism, especially in settings with large, multiattribute outcome spaces (a familiar example is combinatorial auctions [7]). Thus the design of mechanisms where utility functions are bot fully revealed (thus relieving agents of some of the computational and communicational burden) has become an important problem in computational mechanism design.

In this paper we consider the design of one-shot mechanisms that make decisions using partial type information. In Section 2, we present a model of partial revelation and survey those models and results that influence our approach, with emphasis on the tension between partial revelation and dominant strategy implementation. In Section 3, we show that relaxing implementation to Bayes-Nash or ex-post does not allow for the design of "useful" partial revelation mechanisms. We therefore consider *approximate dominant incentive compatibility*, in which the potential gain from misreporting one's partial type in bounded. We define a class of *regret-minimizing* mechanisms (Section 4) that chooses an outcome that minimizes the worst-case loss (w.r.t. social welfare) over all possible types in the declared partial types. We then define several payment schemes, describe the important properties of our mechanisms (specifically, approximate efficiency, rationality and incentive compatibility) and argue for their suitability. While these results hold for any partial types, the quality of the approximation depends critically on the choice of partial types. In Section 5 we define an algorithm to optimize the choice of partial types that allows one to tradeoff the amount of elicitation with the degree of efficiency and incentive compatibility loss. Taken together, regret-based mechanisms and the optimization algorithm provide a framework for the automated design of *partial revelation mechanisms* in which one can explicitly address such tradeoffs. Preliminary computational experiments confirm the efficacy of our approach. We defer all proofs to a longer version of the paper.

2 Background and Definitions

We begin with some essential background. To motivate our definitions, we will use a simple running example of a buyer wishing to purchase a car from a seller. We wish to facilitate the negotiation: they must agree on the car (from the seller's inventory) and the price to be paid; but buyer's valuation for different cars and the seller's cost is not known to us. Ideally, we would identify the car that maximizes surplus (the difference between the buyer's valuation and the seller's cost).

2.1 Mechanism Design

We adopt a standard quasi-linear environment with n agents in which the aim is to choose an *outcome* or *allocation* $\mathbf{x}$ from the set $\mathbf{X}$ of all possible allocations. Each agent $i \leq n$ has *type* t_i drawn from set T_i, and valuation function $v_i : \mathbf{X} \times T_i \rightarrow \mathbb{R}$, with $v_i(\mathbf{x}; t_i)$ denoting the value of allocation $\mathbf{x}$ if i has type t_i. In many cases, we can view t_i as encoding i's utility function over $\mathbf{X}$. Let $T = \prod_i T_i$ be the set of full type vectors. The *social welfare* of $\mathbf{x}$ given $t \in T$ is $SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$. Let t_{-i} denote a type vector over all agents but i, and $SW_{-i}(\mathbf{x}; t) = \sum_{j \neq i} v_j(\mathbf{x}; t_j)$. In our example, the buyer's type t_b would determine its valuation $v_b(\mathbf{x}; t_b)$ for any transacted vehicle $\mathbf{x}$ (and similarly for the seller, its cost). The space of possible types T_b could be defined in a number of ways. For instance, if $|\mathbf{X}| = n$, T_b could be the set of n-vectors $\mathbf{0} \leq \mathbf{v} \leq \mathbf{c}$ for some constant c (with v_i denoting the utility of the ith vehicle). However, valuations can often be represented more compactly. For instance, if a buyer's utility is known to be linear with respect to a small set of car features, then a T_b may be k-dimensional (for some $k < n$), and any type captured by a set of k *parameters*.

A *mechanism* consists of a set of actions $A = \prod_i A_i$, an allocation function $O : A \rightarrow \mathbf{X}$ and n payment functions $p_i : A \rightarrow \mathbb{R}$. Intuitively, the mechanism offers the action set A_i to i, and chooses an allocation based on the actions taken by each agent. We assume *quasi-linear utility*; that is, an agent i's utility for an allocation $\mathbf{x}$ and payment ρ_i is $u_i(\mathbf{x}, \rho_i, t_i) = v_i(\mathbf{x}; t_i) - \rho_i$. Mechanism m induces a (Bayesian) game assuming probabilistic beliefs over types: each agent i adopts a strategy $\pi_i : T_i \rightarrow A_i$ associating an action with its type.[1]

The goal of mechanism design is to design m to implement some SCF $f : T \rightarrow \mathbf{X}$. For instance, f may be social welfare maximization (i.e., $f(t) = \arg\max SW(\mathbf{x}; t)$). In this work, we focus on social welfare maximization or *efficient allocation*. Implementation then depends on the equilibrium concept used; specifically, if m induces strategies π_i for each agent in equilibrium such that $O(\pi(t)) = f(t)$ for all $t \in T$, we say that m *implements* f. Standard equilibrium concepts lead to *dominant strategy*, *ex post*, and *Bayes-Nash* implementation.[2] The *revelation principle* allows one to focus attention on direct, incentive compatible mechanisms in which $A_i = T_i$ and each agent will reveal his type truthfully in equilibrium. In our example, this would allow restriction to mechanisms in which we ask the car buyer and seller to report their *complete* valuation and cost functions, respectively.

The Groves scheme is a famous class of mechanisms for quasi-linear environments (of which VCG is an instance) in which social welfare maximization is implemented in dominant strategies: it is characterized by any efficient allocation function, and the Groves payments [11]:

$$p_i(t) = p_i(t_{-i}, \mathbf{x}^*) = h_i(t_{-i}) - SW_{-i}(\mathbf{x}^*; t_{-i}) \quad (1)$$

for any functions $h_i : T_{-i} \rightarrow \mathbb{R}$. In non-trivial settings, the Groves scheme is the *only* class of mechanisms that can implement *any* SCF in dominant strategies. This follows from two famous results: Roberts [22] showed that if $\mathbf{X}$ contains at least 3 outcomes and all valuations are possible, then an SCF is implementable in dominant strategies iff it is an affine welfare maximizer (i.e., affine transformation of social welfare); while Green and Laffont [10] proved that to implement an affine welfare maximizer one must use Groves payments. Thus, to implement social welfare maximization in dominant strategies, one must not only elicit enough information to determine the efficient allocation but generally also enough *further* information to determine the Groves payments.[3]

2.2 Partial Revelation Mechanisms

We define a *partial type* $\theta_i \subseteq T_i$ for agent i to be any subset of i's types. A partial type vector θ includes a partial type for each agent. A *(direct) partial revelation mechanism* (PRM) is any mechanism in which the action set A_i is a set of partial types Θ_i (i.e., the agent is asked to declare the partial type in which its true type lies). Since agents only reveal partial types, the notion of truth telling must be relaxed somewhat:

Definition 1. *A PRM is* incentive compatible (IC)*—under the relevant equilibrium concept—if it induces equilibrium strategies π_i for each agent i such that $t_i \in \pi_i(t_i)$.*

In other words, an IC PRM will induce each agent to report a partial type that contains its true type. In our example, we might define the partial type space of the buyer by defining a set of rough bounds on the valuation for each car, with some cars having a more precise range than others. For instance, one partial type might assert that $v_b(car_1) \in [17,000, 20,000]$, $v_b(car_2) \in [23,000, 24,000]$, and so on. Limiting revelation to partial types of this form allows the buyer to negotiate without having to precisely determine its valuation for each car, but simply estimate it roughly.

Partial types may or may not be overlapping or exhaustive. If they are not exhaustive, incentive compatibility is not generally possible. If they are overlapping, more than one truthful report may be possible, and an agent can reason strategically to choose among them while maintaining truthfulness, something that is not possible if types do not overlap. Our key results, specifically incentive and efficiency guarantees, do not require non-overlapping types. We will, however, assume in what follows that partial types are exhaustive.

In general, given a partial type vector θ, a mechanism will not be able to determine an efficient allocation $\mathbf{x}$—the allocation $\mathbf{x}^*(t)$ that maximizes social welfare for type t may be different for various $t \in \theta$. A corollary of Roberts' result is that a one-shot PRM cannot be used for dominant strategy implementation: unless the partitioning of each agent's type space is such that each joint partial type θ determines a unique maximizing $\mathbf{x}^*$ for all $t \in \theta$, dominant strategy implementation will not be realized (in general).

[1] Without priors, the game is in *pre-Bayesian form* [12] or has *strict type uncertainty* [13].

[2] If each agent has a *dominant strategy* (i.e., a strategy that is optimal no matter what others do) in the induced game, then we have dominant strategy equilibrium and implementation. An *ex post equilibrium* is a vector of strategies π such that π_i is optimal for i even when the types of others are known to i (assuming strategies in π). A *Bayes-Nash equilibrium* is such that π_i maximizes i's utility in expectation over others' types.

[3] In auction-like settings, this result holds for any SCF satisfying the *independence of irrelevant alternatives* [15].

2.3 Related Work

Recent research has examined methods involving limited or incremental elicitation of types to circumvent the difficulties of full type revelation (especially in single-good and combinatorial auctions). Most work on incremental elicitation involves techniques that elicit "just enough" information to determine the VCG outcome fully, both allocation *and* payments fully, thus maintaining incentive properties (e.g., in CAs [5; 20]). Such mechanisms, being incremental, do not fit precisely into our framework, and generally allow only ex post implementation; issues pertaining to approximate efficiency are avoided altogether. Furthermore, the sequential approach is critical to avoiding full revelation. Unfortunately, the amount of information required to fully determine the VCG outcome may be considerable [19] (and enforcing IC over and above just implementing an SCF itself induces significant cost [8]).

Alternatively one can elicit enough information to determine an *approximately* optimal outcome, a common approach in single-agent elicitation [4], sacrificing decision quality to reduce elicitation effort. We adopt this perspective here. Recently, we used this approach in the design of *sequential* PRMs [14], leading to approximate ex-post implementation. The one-shot mechanisms presented in this paper have stronger incentive properties (approximate dominant implementation), and allow for more interesting payment schemes. Another class of approaches to PRMs is exemplified by priority games [3; 2]. In these models, partial types are elicited and exact (not approximate) dominant strategy implementation is realized. However, these PRMs are designed only to deal with agents having one-dimensional (or "single-parameter") types, for example, agents in a single-item auction where valuation for an outcome can be specified by one parameter. Indeed, Roberts' result is escaped only by restricting the space of preferences in this severe fashion. Combinatorial auctions with single-minded agents are similarly restricted [16]. Unlike our model, these mechanisms do not generalize to more realistic valuation structures [15].

Approximate IC has been considered before from several perspectives. Nisan and Ronen [18] show that *computational* approximation of VCG can destroy truth-telling, but manipulation of approximate VCG can be made computationally difficult [23], thus inducing "practical" incentive compatibility. IC in expectation or with high probability can also be demanded [1]. Finally, one can attempt to bound the gain an agent can achieve by lying (e.g., as proposed for exchanges in [21]). It is this latter view of approximate IC that we adopt here. Our class of partial revelation mechanisms, along with our partitioning algorithm provide the first approach to automated *partial revelation* mechanism design [6].

3 Bayes-Nash and Ex-Post Implementation

In the Bayes-Nash context, each agent has a probabilistic prior over the types of the others, $Pr(t_{-i}|t_i)$. If truth-telling is a Bayes-Nash equilibrium in a PRM, this defines a distribution over the reports (partial types) of other agents; hence for each of its reports θ_i, agent i has a distribution, $x_i^{\theta_i}$, over allocations selected by the mechanism. For each $\mathbf{x}$ define:

$$x_i^{\theta_i}(\mathbf{x}) = Pr(\mathbf{x}|\theta_i) = \sum_{\theta_{-i}} Pr(\theta_{-i}|\theta_i) \cdot Pr[O(\theta_i\theta_{-i}) = \mathbf{x}]$$

In this section, we restrict ourselves to partitions of the type space that are "grid-based": each parameter's space of values is split into a finite number of intervals of the form $[lb, ub)$, with $lb < ub$. We can derive the following results:

Theorem 1. *If all valuation functions are possible, in a Bayes-Nash IC grid-based PRM, we have:*

$$\forall i, \forall \theta_i, \theta_i' \quad : \quad x_i^{\theta_i} = x_i^{\theta_i'} =_{\text{def}} x_i \qquad (2)$$

$$\forall i, j \quad : \quad x_i = x_j \qquad (3)$$

Property 2 states that, regardless of its report, agent i has the same posterior x_i over outcomes. Property 3 states that this distribution is also the same for all agents. If the allocation function is deterministic, then it selects the same allocation for each report vector. We call such a mechanism *trivial*. Triviality may be avoided if allocations are probabilistic, but even then, these properties are very restrictive:

Proposition 1. *No Bayes-Nash IC grid-based PRM has higher expected social welfare than the trivial mechanism that always picks the allocation with highest* ex ante *social welfare.*

Proposition 2. *If ex-interim (or,* a fortiori *ex-post) individual rationality (IR) is required, the expected sum of payments of any Bayes-Nash IC grid-based PRM is zero.*

In a sense, though partial revelation Bayes-Nash implementation is not strictly trivial, it is useless since it achieves the same result as a mechanism with no revelation. Given that an ex-post equilibrium is a vector of strategies that are in Bayes-Nash equilibrium for all possible probabilistic priors, the above results imply that any ex-post IC PRM is trivial.

Note that the grid-based restriction is a sufficient condition for the above results to hold. We are currently looking into identifying necessary conditions. For example, we strongly suspect that any partitioning of type space into convex partial types will lead to the same negative results. We do not have such results at present however.

4 Regret-based PRMs

A partial revelation mechanism must choose an allocation $\mathbf{x}(\theta)$ for each reported partial type $\theta \in \Theta$, but cannot generally do so in a way that ensures efficiency. We propose the use of *minimax regret* [4; 14] to choose the allocations associated with each partial type vector.

Definition 2. *The* pairwise regret *of decision $\mathbf{x}$ with respect to decision $\hat{\mathbf{x}}$ over feasible type set θ is*

$$R(\mathbf{x}, \hat{\mathbf{x}}, \theta) \quad = \quad \max_{t \in \theta} SW(\hat{\mathbf{x}}; t) - SW(\mathbf{x}; t), \qquad (4)$$

This is the most the mechanism could regret choosing $\mathbf{x}$ instead of $\hat{\mathbf{x}}$ (e.g., if an adversary could impose any type vector in θ). The maximum regret *of decision $\mathbf{x}$ and the* minimax regret *of feasible type set θ are respectively:*

$$MR(\mathbf{x}, \theta) \quad = \quad \max_{\hat{\mathbf{x}}} R(\mathbf{x}, \hat{\mathbf{x}}, \theta) \qquad (5)$$

$$MMR(\theta) \quad = \quad \min_{\mathbf{x}} MR(\mathbf{x}, \theta) \qquad (6)$$

A *minimax-optimal* decision for θ, denoted $\mathbf{x}^*(\theta)$, is any allocation that minimizes Eq. 6. Without distributional information over the set of possible utility functions, choosing (or recommending) a minimax-optimal decision $\mathbf{x}^*$ minimizes the worst case loss in efficiency with respect to possible realizations of the types $t \in \theta$. We refer to the regret maximizing $\hat{\mathbf{x}}$ in Eq. (6) as the *witness* for $\mathbf{x}$.

Recent approaches to minimax regret optimization have shown it to be practical when utility models are *factored* into a convenient functional form such as *generalized additive independence (GAI)* [9], and utility uncertainty is expressed in the form of linear constraints on such factored models [4]. In this setting, minimax regret optimization can be formulated as a linear, mixed-integer program (MIP) with exponentially many constraints, but can be solved using an iterative constraint generation procedure that, in practice, enumerates only a small number of (active) constraints [4].

Definition 3. *A* regret-based partial revelation mechanism *is any mechanism in which the allocation function chooses an outcome that minimizes max regret given the revealed partial type vector; that is, $O(\theta) = \mathbf{x}^*(\theta)$ for all $\theta \in \Theta$.*

Assume we have a PRM m in which each agent declares a partial type $\theta_i \in \Theta_i$, and that m is regret-based, i.e., O chooses $\mathbf{x}^*(\theta)$ with minimax regret w.r.t. social welfare for any declared type vector θ:

Observation 1. *Let m be a regret-based partial revelation mechanism with partial type space Θ. If $MR(\mathbf{x}^*(\theta), \theta) \leq \varepsilon$ for each $\theta \in \Theta$, then m is ε-efficient for truth-telling agents.*

This simply formalizes the obvious fact that since max regret for any mechanism choice is bounded by ε, then if all agents reveal their partial types truthfully we are assured to be within ε of maximizing social welfare.

With PRMs we cannot generally guarantee efficiency: different type profiles within a partial type vector θ may require a different allocation choice to maximize social welfare. As a consequence, the result of Roberts means we will be unable to implement our "approximate" choice function in dominant strategies. Instead, we relax the implementation concept in a natural fashion and derive a payment scheme that ensures approximate IR and IC in dominant strategies.

Consider the following generalization of Groves payments. Given joint report $\theta = (\theta_i, \theta_{-i})$ of all agents, and the corresponding choice $\mathbf{x}^*$, agent i's payment is:

$$p_i(\theta) = p_i(\theta_{-i}, \mathbf{x}^*) = h_i(\theta_{-i}) - SW_{-i}(\mathbf{x}^*; f_i(\theta_{-i}))$$

where $h_i : \Theta_{-i} \rightarrow \mathbb{R}$ is an arbitrary function and $f_i : \Theta_{-i} \rightarrow T_i$ is any function that, given partial type vector θ_{-i}, selects a type vector t_{-i} from that set (i.e., $f_i(\theta_{-i}) \in \theta_{-i}$).

Recall that under full revelation, $f_i(\theta_{-i})$ is the *complete* type vector t_{-i} reported by the other agents and h_i must take that particular t_{-i} as an argument. Our *partial Groves payment scheme*, however, can select an *arbitrary* type for each agent consistent with their declared partial types and apply standard Groves payments (Eq. 1) to these. The selected types can differ for each payment function p_i, and the arbitrary h_i functions also depend only on the partial types revealed. Partial Groves payments can thus require significantly less revelation. Together with regret-based allocation, they give:

Theorem 2. *Let m be a regret-based partial revelation mechanism with partial type space Θ and partial Groves payment functions p_i. If $MR(\mathbf{x}^*(\theta), \theta) \leq \varepsilon$ for each $\theta \in \Theta$, then m is ε-efficient and ε-dominant IC.*

In other words, truth telling is an ε-dominant strategy equilibrium (i.e., truth telling for any agent has utility within ε of optimal regardless of the reports of others).

We can specialize partial Groves payments to *partial Clarke payments*:

$$p_i(\theta_{-i}, \mathbf{x}^*) = SW(\mathbf{x}^*_{-i}(\theta_{-i}); f_i(\theta_{-i})) - SW_{-i}(\mathbf{x}^*; f_i(\theta_{-i}))$$

where $\mathbf{x}^*_{-i} : \Theta_{-i} \rightarrow \mathbf{X}$ is an arbitrary function that chooses an allocation based only the reports of agents *other than* i. This restriction allows the following IR guarantee:

Theorem 3. *Let m be a regret-based partial revelation mechanism with partial type space Θ and partial Clarke payments p_i. If $MR(\mathbf{x}^*(\theta), \theta) \leq \varepsilon$ for each $\theta \in \Theta$, then m is ε-efficient, ε-dominant IC and ex-post ε-IR.*

In other words, no agent has incentive greater than ε not to participate in the mechanism.

We have provided some intuitive justification above for the use of minimax regret to determine the allocation associated with any revealed partial type profile. We can also provide formal justification for the use of minimax regret with respect to incentive properties of PRMs. Specifically, under the partial Groves and Clarke payment schemes, worst-case manipulability (over possible type profiles) is exactly equal to the greatest minimax regret-level (again, over possible type profiles). Thus one can show:

Proposition 3. *Let Θ be a fixed partial type space, and M a regret-based PRM (w.r.t. Θ) using the partial Clarke payment scheme. Any other PRM M' (w.r.t. Θ) using the partial Clarke scheme, will have worst-case manipulability, efficiency loss and rationality violation at least as great as that of M. There exist non-regret-based PRMs where this inequality is strict.*

In other words, no non-regret based scheme can perform better than a regret-based scheme with respect to efficiency loss (obviously), gain from misreporting one's partial type, and incentive for non-participation. The analog of this proposition holds regarding manipulability and efficiency when using the partial Groves payment scheme.

Even with the "Clarke-style" restriction above, our payment scheme is quite general: $\mathbf{x}^*_{-i}$ and f_i are arbitrary functions. The choice of these will not affect the worst-case properties above, but it can be used to: (a) reduce the likelihood (if any) of a rationality violation; and/or (b) maximize revenue of the mechanism. If reducing or removing a rationality violation implies revenue loss, then a trade-off can be made between the two criteria. An attractive feature of our PRMs is the considerable scope for optimization of the payment scheme due to the nature of the partial type revelation.

When dealing with *approximate* incentive properties, one must be aware that a small deviation from the truth by one agent can cause major changes in the mechanism's allocation (leading, say, to large losses in efficiency). But with partial Groves payments, an agent can gain at most ε compared to revealing its partial type truthfully. In most settings, the computational cost of finding a good lie, especially given the considerable uncertainty in the value of any lie (due to uncertainty about others' types), will be substantial (see, e.g., [23]). Thus, if ε is small enough, it will not be worth the cost: our *formal, approximate* incentive compatibility is sufficient to ensure *practical, exact* incentive compatibility.

To develop a sense of the difficulty associated with manipulating such a mechanism, consider that an agent must be able to compute an untruthful strategy (or lie) with greater utility than truth-telling in order to exploit our approximate incentive guarantee. To do this one must first determine the true value of a lie (incurring the valuation or cognitive costs similar to revealing truthfully). However evaluating a lie also requires considerable (and accurate) information about the types and strategies of the others; even with decent priors, the costliness of such computations (e.g., in time, cognitive, or computational resources) implies that manipulation is not worthwhile unless the bound ε is quite loose, and incentive compatibility will thus, in practice, be exact.

A similar argument can be made regarding approximate IR: determining whether you gain from not participating will be very difficult. Thus a *potential* small loss will be worthwhile for an agent given the savings our mechanism provides in revelation and computational costs (relative to the full revelation alternative). Finally, given the complexity of many mechanism design settings, when cognitive, computational and communication costs are accounted for, the potential loss in efficiency will be an acceptable trade-off, given the high level of revelation required by exactly efficient mechanisms.

5 Construction of Partial Types

So far we have focused on the design of regret-based PRMs with a fixed collection of partial types. However, the selection of partial types is critical to the performance guarantees above, since it is these types that determine the degree of regret incurred by the mechanism. The key design issue is the construction of a suitable set of partial types that minimizes both revelation and the maximum minimax regret (i.e., ε) over that set. We describe a heuristic, but reasonably tractable, approach to the automated optimization of the type space partitioning. Although the class of mechanisms is fixed and it is the partition that is being optimized by our algorithm, together these constitutes a tractable approach to automated *partial revelation* mechanism design.

In what follows, we assume an agent type is simply a bounded n-vector with valuations for each allocation $\mathbf{x} \in \mathbf{X}$. When dealing with a multi-attribute outcome space, we allow for an agent's type/utility function to be *factored* using a generalized additive independence representation [9], in which utility parameterized with local sub-utility functions over small sets of attributes. In a flat (un-factored) model, the parameters are simply the valuations for allocations $\mathbf{x}$. We assume in what follows that the type space T_i is given by upper and lower bounds over the parameters of agent i's utility model and focus on partial types specified similarly.

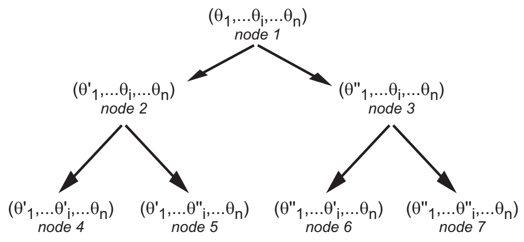

Figure 1: Example of a mechanism tree.

5.1 Partial Type Optimization Algorithm

We describe an *offline*, iterative, myopic approach to the optimization of agent type space partitions. It is myopic in the following two senses: (a) at each step, it focuses on reducing the minimax regret of the joint partial type with greatest regret by refining (or splitting) it, without considering the impact on other partial types; and (b) it only considers the immediate effects of this refinement, with no look-ahead to future splits.

To simplify the presentation, we first describe a naive, computationally intensive method, formulated in terms of decision tree construction, and then show how it can be modified to be made much more tractable. The algorithm uses a heuristic function which, given a partial type vector, selects an agent whose partial type will be split. It is important to realize that these splits are not "queries" to the agent—the mechanism is not sequential. Rather, splitting a partial type further will increase the number of partial types from which an agent must choose when the mechanism is actually executed. Once all splits to all agents are determined, the mechanism will ask agents to select from the partial types induced by this refinement process. In other words, offline we construct the partial types used by the *one-shot* mechanism. We discuss the heuristic function further below.

Figure 5.1 illustrates the creation of partial types for a PRM in terms of decision tree construction. At the outset, the only information available to the mechanism is the set of possible types for each agent given by our prior, defining the initial partial type vector $(\theta_1, \ldots, \theta_n)$ with $\theta_i = T_i$. This labels the initial (root) node of our tree (node 1). We call the heuristic function on this vector, which selects an agent, say agent 1, and a split of that agent's partial type θ_1 into two more refined partial types θ'_1 and θ''_1. The reasons for choosing a particular split are elaborated below, but intuitively, such a split should have a positive impact with respect to max regret reduction of the mechanism. This creates two child nodes corresponding to partial type vectors $(\theta'_1, \ldots, \theta_i, \ldots, \theta_n)$ and $(\theta''_1, \ldots, \theta_i, \ldots, \theta_n)$ (see nodes 2 and 3 in Figure 5.1). These two new leaves in the tree correspond to the partial type vectors to be used by the mechanism should the splitting process terminate at this point. Thus, we update the partial type space Θ_i for agent 1 by removing θ_1 and adding θ'_1 and θ''_1. We then compute the minimax regret level, optimal allocation and witness (in a single optimization) for these two new leaf nodes given their partial type vectors.

With multiple leaves, the heuristic function must first select a leaf node for splitting before selecting a split. It does this by selecting the partial type vector (leaf node) with greatest minimax regret. The algorithm iterates in this fashion, repeatedly selecting the leaf node with greatest minimax regret and using the heuristic to decide which agent's partial type (within that node) to split (and how to split it), until some termination criterion is met (e.g., the worst-case max regret is reduced to some threshold, or some maximum number of partial types—per agent or overall—is reached).

Unfortunately, unlike standard decision tree construction, a split at one leaf has implications for all other leaves as well. For example, after the initial split above, a split may be recommended at node 2 in the tree, corresponding to $(\theta'_1, \ldots, \theta_i, \ldots, \theta_n)$. Suppose a split of agent i's partial type θ_i into θ'_i and θ''_i is suggested for some $i \neq 1$. Since θ_i is included in the partial type vectors of *both* nodes 2 and 3, this split affects both child nodes, since agent i will have to distinguish (at the very least) θ'_i from θ''_i. We there must replace nodes 2 and 3 with four new leaf nodes (nodes 4–7), corresponding to the combinations of θ'_1, θ''_1 and θ'_i, θ''_i.

This naive approach has two obvious problems. First, there is an exponential blow-up in the number of leaves of the "mechanism tree" since any reasonable heuristic (including the one described below) will often (roughly) alternate splits between the agents whose type uncertainty is still relevant. The algorithm is therefore computationally demanding. Second, consider the example above. The split of θ_i at the second iteration of the algorithm was recommended by the heuristic based on the partial type vector at node 2 (which includes θ'_1), because of its ability to reduce the minimax regret level of that specific partial type vector. However, this split may have little or no effect on minimax regret when applied to node 3 (in which agent 1's partial type θ''_1 is different). This split may indeed be "useless" when considered at node 3. These problems can be avoided by modifying the algorithm as follows.

When a split is made at some node k, even though the partition has been updated in a way that might affect a partial type at another node k', we can choose to ignore the effect on k'. Thus node k' corresponds to a partial type vector that is no longer in our partition but includes a collection of partial type vectors that are. We call k' a *joint* node, and for each such node, we record the splits that have been ignored. In our example, the split of node 2 on θ_i may be ignored at node 3, leaving node 3 to be a joint node. In the tree, node 2 would generate two descendants (nodes 4 and 5), while node 3 would remain a leaf node (nodes 6 and 7 would not be generated). While the split of i's type into θ'_i and θ''_i will *eventually* need to be considered at node 3 (or its descendants), we defer this decision (as discussed below), and can consider making additional splits of node 3 first should this split of θ_i be of little value. This saves having to consider the splits of multiple descendants of node 3 independently.

Note that multiple ignored splits may be "nested". For example, perhaps the first split cuts the partial type into one where some utility parameter p_1 is greater than $.5$ (call this $\theta_i^{>0.5}$) and one where it is lower ($\theta_i^{<0.5}$); the second splits $\theta_i^{>0.5}$ on p_1 at .7; the third splits $\theta_i^{<0.5}$ along parameter p_2 at $.4$, and so on. When such joint node is the leaf with greatest regret level (i.e., the one to be considered for expansion), we first check if there is a useful ignored split before considering the split recommended by the heuristic. If so, we "un-ignore" it, thus creating another leaf but without increasing the complexity of the partition. The precise way in which we select splits is described in Tables 1 and 2. A split is called "good" if both resulting nodes have lower regret than the node that was split, and "OK" if only one of them does. With this improved algorithm, new mechanism nodes are only added if they are helpful. Since the naive approach results in useless splits, computational requirements can be greatly reduced by adopting this more sophisticated approach.

```
Find leaf node $N$ with highest regret
Call heuristic on $N$'s partial type vector. Output: $hSplit$
**If** there exists ignored split $iSplit$ on same parameter as $hSplit$:
  $splitNode(N, iSplit)$
**Else** Search for "good" ignored split $gSplit$
  **If** there exists one:
    $splitNode(N, gSplit)$
  **Else** Test $hSplit$
    **If** $hSplit$ is "good':
      $splitNode(N, hSplit)$
    **Else-if** $hSplit$ is "OK':
      Search for "OK" ignored split $okSplit$
      **If** there exists one:
        $splitNode(N, okSplit)$
      **Else**:  $splitNode(N, hSplit)$
```

Table 1: Split selection algorithm

```
$splitNode(N, split)$
  $\theta$ = partial type vector of $N$; $i$ = agent involved in $split$
  split $\theta_i$ into $\theta'_i$ and $\theta''_i$ according to $split$
  update $i$'s partition
  compute new MMR for both $(\theta'_i, \theta_{-i})$ and $(\theta''_i, \theta_{-i})$
  create corresponding new nodes $N'$ and $N''$
  separate $N$'s ignored split list into those for $N'$ and $N''$
  replace leaf $N$ in tree with leaf nodes $N'$ and $N''$
  **For all** leaf nodes with $\theta_i$ in their partial type vector
    add $split$ to list of ignored splits
```

Table 2: splitNode function

5.2 Heuristic Function

The role of the heuristic function is to split a partial type *vector* in two, creating one additional node in the mechanism. This must be done by splitting the partial type of one of the agents. However, in the case of a joint node, the partial type of an agent passed to the heuristic function may correspond to several "actual" partial types in the partition. For example, the input partial type may ignore the fact that it has been split along parameter μ_1 at the value .5. If the heuristic recommends splitting that partial type along parameter μ_2, this corresponds to splitting *two* actual partial types (with values greater or less than .5 for μ_1). But the partial type vector is only split in two, each successor node inheriting the ignored splits of its parent.

In a flat utility model, each utility parameter is simply the valuation for an allocation $\mathbf{x}$. In this case, our heuristic can be described as follows. At each node, given its partial type vector, we have the corresponding minimax regret solution $\mathbf{x}^*$ and witness $\hat{\mathbf{x}}$. Intuitively, regret can be reduced by raising the lower bound of $\mathbf{x}^*$ or lowering the upper bound on $\hat{\mathbf{x}}$. However, since the optimization is offline, the split "Is utility for $\mathbf{x}$ greater than .5?" must account for both possibilities, yes and no. When splitting, say, $\mathbf{x}^*$, the partial type corresponding to the no answer (i.e., lowering the upper bound) is unlikely to have lower regret unless $\mathbf{x}^*$ also turns out to be the witness of the second best regret minimizing allocation. In that case, lowering its upper bound will help reduce the second lowest regret and raising the lower bound will help with the lowest. We therefore also compute the second lowest max regret solution $\mathbf{x}^{*2}$ and its witness $\hat{\mathbf{x}}^2$. If both $\hat{\mathbf{x}} = \mathbf{x}^{*2}$ and $\mathbf{x}^* = \hat{\mathbf{x}}^2$ are true, or neither is true, we split the one with the largest gap (the difference between its upper and lower bounds). If only one is true, we split that parameter, unless its gap is below some threshold, and the other gap is not.

This seemingly trivial strategy has interesting properties when utility models are factored using GAI. In this setting, our heuristic is an adaptation of the *current solution* elicitation strategy of [4] to an offline one-shot elicitation scenario. We defer details to a longer version of this paper.

6 Empirical Results

We report on experiments in a multi-attribute negotiation domain. A buyer and a seller are bargaining over a multi-attribute good to trade. The set of 16 possible goods is specified by four boolean variables, X_1, X_2, X_3, X_4, denoting the presence or absence of four specific item features. The buyer's valuation and the seller's cost are represented using GAI structure. Each agent's utility function is decomposed into two factors, each factor with two different variables: $v_b(\mathbf{x}) = v_b^1(x_1, x_2) + v_b^2(x_3, x_4)$ and $v_s(\mathbf{x}) = v_s^1(x_1, x_3) + v_s^2(x_2, x_4)$. Each subutility function is specified using four parameters, indicating the local value of the four possible combinations of features. Thus eight utility parameters fully define each agent's utility function.

Initial prior bounds on utility parameters (i.e., the initial type space) are drawn uniformly between 0 and 100 (seller cost) or 100 and 200 (buyer value), ensuring a positive transaction exists. Though 16 goods may seem small, it is much larger than problems solved by existing automated approaches to (one-shot) mechanism design (all of which are restricted to a small, finite type space). The social welfare maximizing allocation is that good that maximizes surplus (difference between buyer value and seller cost).

We assess the performance of our mechanisms by showing how the *worst case* minimax regret level (over all possible agent types) reduces with the number of partial types created by our approach (expressed in number of bits). Since PRMs are designed to work for agents with any true types, there is no *single* true type or optimal allocation to compare to. We could simulate *specific* agents, but our worst-case regret bounds any such "true loss" results, and these bounds are tight in the sense that at least one set of agent type profiles will incur this worst case regret. Of course, regret any specific collection of agents (or type profile) will generally be lower, so we also show *expected* regret in our results, assuming a uniform distribution over type parameters. We compare our myopic approach to type construction (i.e., where regret reduction is used to determine splits of partial types) to a simple approach for partial type construction that simply splits each partial type evenly across all parameters.[4]

Figure 6 shows the worst case minimax regret level of our regret-based PRMs (averaged over thirty runs using different priors) when partial types are constructed using our myopic algorithm and when uniform splitting is used. We also show expected minimax regret level (assuming a uniform distribution over types). Results are reported as a function of the number of bits of communication necessary for an agent to report some partial type in the proposed partition. Bounds on manipulability, efficiency loss, and rationality violation are all dictated by this worst-case regret (depending on whether partial Groves or Clarke payments are used).[5]

It is clear that using regret-reduction to decide how to "refine" partial type space offers a significant improvement over a uniform partitioning. Our regret-based approach provides good anytime behavior while uniform partitioning reduces ε as a step function (averaging smooths the results in the graph). Naturally, average manipulability is lower than worst-case manipulability, thus further justifying the use of approximate incentives. We note that the initial regret level (assuming only one partial type, i.e., T_i, for each agent i) corresponds to an error between 50% and 146% of the optimal social welfare, depending on the actual agent types, and is reduced using 11 bits of communication (to communicate one's partial type) to 20–56% error by our regret-based approach versus only 30–86% by the uniform approach. With only 11 bits, the regret-based approach reduces efficiency loss and manipulability by about 60% while a uniform partition reduces it by about 38%. To reach an average manipulability level of around 70, a uniform approach requires 11 bits compared to 6.5 for regret-based splitting, which constitutes a 40% savings in communication. To reach manipulability level of roughly 90, the regret-based approach provides a 50% savings in communication (5.5 bits vs. 11 bits). Note that this savings will be realized repeatedly if the mechanism is used to support, say, multiple bargaining instances. Finally, it is worth remarking that 11 bits corresponds to about 1.4 bits per parameter and 0.7 bits per allocation, which is quite small in an example of this size.

Preliminary tests using a (computationally demanding) myopically optimal heuristic (that considers each parameter and chooses the highest regret drop) show only modest improvement over regret-based splitting, thus motivating the investigation of non-myopic splitting techniques.

7 Conclusion

We have proposed a general model for partial revelation mechanisms in which social welfare maximization is the desired objective and explored their incentive properties. As a result of our negative results on Bayes-Nash and ex-post implementation (along with classical results on dominant strategies), we have relaxed the requirement of exact incentive

[4]This is simulated so that at each step only one partial type is split in order to allow a more accurate comparison to our approach.

[5]For agents with a *specific* type, we can usually derive much tighter bounds on gain from manipulation than the worst case ε; hence expected manipulability is less.

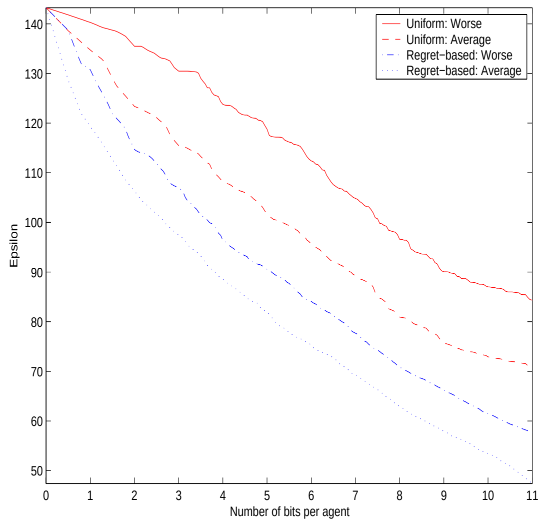

Figure 2: Worst case and expected ε, as a function of the number of bits used per agent. Averaged over 30 runs.

compatibility and focused on approximating both the efficiency and incentive properties of PRMs, allowing one to exploit the trade-off between the computational, communication and cognitive costs of type revelation with the degree of approximation. Regret-based PRMs, in particular, allow one to bound efficiency loss, gain from manipulation and non-participation, and admit promising optimization methods for the automated design of PRMs. Critically, when the gain from non-truthful revelation is sufficiently small, our results on formal, approximate incentive compatibility ensures "practical" true incentive compatibility.

Apart from more extensive empirical evaluation, there are several directions in which this work should be extended. The first is the integration of techniques for approximating regret computations into our algorithm to tackle realistic problems [4]. The second is the investigation of more efficient splitting heuristics that improve both the communication requirements of our mechanisms and the computational costs of designing them. The design of non-myopic incremental partial mechanisms is of special interest. Precisely determining the complexity of manipulation by formally modeling the costs involved is also an important task in further justifying our emphasis on approximate IC and IR. We are exploring further benefits offered by partial revelation and (bounded) approximate IC w.r.t. to circumventing some other classic drawbacks and "impossibility" results for direct mechanisms. Finally, we are very interested in the exploiting prior distributional information about types in the construction of partial type space to reduce *expected* efficiency loss and manipulability, while retaining our worst-case guarantees, similar to the approach taken in (full revelation) automated mechanism design [6].

Acknowledgements: Thanks to Vincent Conitzer, Kevin-Leyton Brown, David Parkes and the referees for helpful discussions and comments. This work was funded by NSERC.

References

[1] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *SODA-03*, pp.205–214, Baltimore, 2003.

[2] L. Blumrosen, M. Feldman. Implementation with a bounded action space. *ACM EC-06*, pp.62–71, Ann Arbor, 2006.

[3] L. Blumrosen and N. Nisan. Auctions with severely bounded communication. *FOCS-02*, pp.406–416, Vancouver, 2002.

[4] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Art. Intell.*, 170:686–713, 2006.

[5] W. Conen and T. Sandholm. Partial-revelation VCG mechanisms for combinatorial auctions. *AAAI-02*, pp.367–372, Edmonton, 2002.

[6] V. Conitzer and T. Sandholm. Complexity of mechanism design. *UAI-02*, pp.103–110, Edmonton, 2002.

[7] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, Cambridge, 2005.

[8] R. Fadel, I. Segal. The communication cost of selfishness: ex post implementation. *TARK-05*, pp.165–176, Singapore, 2005.

[9] P. C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *Intl. Econ. Rev.*, 8:335–342, 1967.

[10] J. Green and J.-J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[11] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[12] R. Holzman, N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games and Econ. Behavior*, pp.104–123, 2004.

[13] N. Hyafil and C. Boutilier. Regret minimizing equilibria and mechanisms for games with strict type uncertainty. *UAI-04*, pp.268–277, Banff, AB, 2004.

[14] N. Hyafil and C. Boutilier. Regret-based incremental partial revelation mechanisms. *AAAI-06*, pp.672–678, Boston, 2006.

[15] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. *FOCS-03*, pp.574–583, Cambridge, MA, 2003.

[16] D. Lehman, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49:577–602, 2002.

[17] A. Mas-Colell, M.D̃. Whinston, and J.R̃. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.

[18] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. *ACM EC-00*, pp.242–252, Minneapolis, 2000.

[19] N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices. *J. Economic Theory*, 2006. to appear.

[20] D.C̃. Parkes. Auction design with costly preference elicitation. *Annals of Math. and AI*, 44:269–302, 2005.

[21] D.C̃. Parkes and J. Kalagnanam. Iterative multiattribute Vickrey auctions. *Management Science*, 51:435–451, 2005.

[22] K. Roberts. The characterization of implementable choice rules. In J.-J. Laffont, ed., *Aggregation and Revelation of Preferences*, pp.321–349. North-Holland, Amsterdam, 1979.

[23] S. Sanghvi and D.C̃. Parkes. Hard-to-manipulate combinatorial auctions. Tech. report, Harvard Univ. Boston, 2004.