Partial Revelation Automated Mechanism Design

Nathanaël Hyafil
Department of Computer Science
University of Toronto
Toronto, ON, M5S3H5, CANADA
{nhyafil}@cs.toronto.edu

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, ON, M5S3H5, CANADA
{cebly}@cs.toronto.edu

Abstract

In most mechanism design settings, optimal general-purpose mechanisms are not known. Thus the automated design of mechanisms tailored to specific instances of a decision scenario is an important problem. Existing techniques for automated mechanism design (AMD) require the revelation of full utility information from agents, which can be very difficult in practice. In this work, we study the automated design of mechanisms that only require partial revelation of utilities. 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. We provide a set of optimization routines that can be combined to address the trade-offs between the amount of communication, approximation of incentive properties, and objective value achieved by a mechanism. This allows for the automated design of partial revelation mechanisms with worst-case guarantees on incentive properties for any objective function (revenue, social welfare, etc.).

Introduction

Intelligent agents acting on behalf of specific, self-interested users are increasingly required to interact with each other in applications ranging from auctions to automated negotiation and bargaining. The design of interaction protocols that lead to desirable outcomes is known as *mechanism design* [6].

A *mechanism* is basically a game intended to implement some *social choice function (SCF)*—a function that selects an outcome as a function of the preferences of participating agents—thus usually requiring agents to reveal something of their preferences. There are, however, few SCFs that can be implemented in general settings without exploiting prior information about agent preferences. (Social welfare maximization is a rare positive example, implementable via the well known class of of VCG mechanisms.) In practice, however, a designer often has prior information over agent types and need only design a mechanism suitable for her particular context. *Automated mechanism design (AMD)* [3] techniques consist of algorithms that, given prior utility information, construct a mechanism tailored to the specific context and social choice objective of the designer.

A key result in mechanism design, the *revelation principle*, states that mechanisms can be restricted to those in which agents *fully* reveal their *true* preferences (e.g., as in VCG). Unfortunately, full preference revelation is often intractable, since utility functions can be extremely difficult for agents to compute effectively or communicate to the mechanism. Thus the design of mechanisms where preferences are only partially revealed has become an important problem in computational mechanism design. AMD has to date focused on full revelation (though see [10] for an exception).

In this work, we explore the automated design of *partial revelation mechanisms* for general objectives. Unfortunately, negative results are known regarding the implementation of partial revelation mechanisms with exact incentive properties [5; 9; 8]. We will therefore consider mechanisms with approximate incentives, where the potential gain an agent can achieve by revealing its type insincerely is bounded. Given the potentially considerable costs required to manipulate a mechanism, if this bound is low enough, manipulation will not be worthwhile; thus approximate incentive compatibility in the formal sense will suffice to induce truthful revelation in practice. This corresponds to the approach of [5], where we considered partial revelation mechanisms that attempt to minimize worst case loss in social welfare. Theoretical results in that work hold only for welfare maximization. Here, we extend the approach to account for general design objectives, and show how to maintain appropriate incentives without relying on those theoretical results.

After providing some background, we describe the general problem of *partial revelation mechanism design (PR-AMD)* in more detail. We then describe algorithmic techniques for both *Bayesian* PR-AMD and *regret-based* PR-AMD and present some preliminary empirical results.

Background

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 $SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$.

A *mechanism* $\mathbf{M}$ consists of a set of actions $A = \prod_i A_i$, an allocation function $m : A \to \mathbf{X}$ and n payment functions $p_i : A \to \mathbb{R}$. 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 $\pi_i : T_i \to A_i$ associating an action with its type.

The goal of mechanism design is to design $\mathbf{M}$ to implement some SCF $f : T \to \mathbf{X}$. For instance, f may be social welfare maximization (i.e., $f(t) = \arg\max SW(\mathbf{x}; t)$). Implementation then depends on the equilibrium concept used; specifically, if m induces strategies π_i for each agent in equilibrium, such that $m(\pi(t)) = f(t)$ for all $t \in T$, we say that $\mathbf{M}$ *implements* f.

If each agent has a *dominant strategy* (i.e., a strategy that maximizes its utility no matter what others do) in the induced game, then we have dominant strategy equilibrium and implementation. In this work, we focus on ε-dominant strategies, namely, strategies which, no matter what others do, cannot be improved by more than ε. A mechanism is *direct* if $\forall i, A_i = T_i$, that is, if the agents' actions are possible reports of their type. A direct mechanism is *(dominant-strategy) incentive compatible (IC)* if reporting one's type truthfully is a dominant strategy. It is ε-IC if truth-telling is ε-dominant. A mechanism is *individually rational (IR)* (resp., ε-IR) if, in equilibrium, no agent can lose (resp., more than ε) by participating in the mechanism. When considering dominant strategy implementation, the *revelation principle* allows one to focus attention on direct, incentive compatible mechanisms in which $A_i = T_i$ and each agent will, in equilibrium, reveal its *full* type truthfully.

Partial Revelation Mechanisms As mentioned, full type revelation is often extremely costly. Following [4; 5], we consider mechanisms where agents only partially reveal their utility information. 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). We thus write $\mathbf{M} = (\Theta, m, p)$ where m and p map partial type vectors to allocations and payments respectively, and $\Theta = \prod_i \Theta_i$. Since agents only reveal partial types, the notion of truth telling must be relaxed somewhat:

Definition 1. *A PRM is* incentive compatible (IC) *if it induces a dominant strategy 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.

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. The incentive guarantees described below do not require non-overlapping types. We will, however, assume in what follows that partial types are exhaustive and write $\theta(t)$ for any partial type that contains t.

Implementation of PRMs with exact incentive properties is either impossible or "useless" in the general mechanism design problem [5; 9; 8]. For this reason, we focus on ε-dominant IC and ε-IR. Note that with this notion of approximate IC, an agent can gain at most ε by lying about its type, 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., [11]). Thus, if ε is small enough, it will not be worth the cost: this *formal, approximate* incentive compatibility is sufficient to ensure *practical, exact* incentive compatibility. 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). We will use the term *manipulability* to refer to the degree of both the incentive and rationality violations.

Automated Mechanism Design For many social choice functions, there is no known mechanism that implements it in general. For example, revenue-maximizing mechanisms are only known for very restricted settings such as one-item auctions [7]. In practice, however, a designer often has prior information over agents' types and only needs to design a mechanism suitable for her particular context. Automated mechanism design (AMD) [3] assumes the designer has a social choice *objective* quantifying social value of outcomes as a function of agent preferences. A mechanism can thus be designed to maximize the expected objective value given a distribution over agent types. Under full revelation and with a finite number of types, this optimization can be expressed as a linear program (LP) where IC and IR are imposed as linear constraints. If each agent has k full types, the number of variables of the LP is $k^n \cdot (|\mathbf{X}| + n)$.

Partial Revelation AMD

In this work, we extend the AMD framework to *partial revelation automated mechanism design (PR-AMD)*. Ideally, given a social choice objective, one would optimize the entire mechanism—allocation and payment functions as well as the type partitioning—in a single optimization. But even with a simple partial type space, this optimization typically requires an intractable polynomial program. To reduce it to a sequence of linear optimizations, we decompose the optimization into two steps: the first to optimize the mechanism functions given a fixed partition; and the second to refine the partition. Given a fixed partition, in order to extend the AMD framework to partial revelation, one must define

a suitable objective function, and adapt the IC and IR constraints to their (approximate) partial revelation equivalent. There are, therefore, three relevant criteria to "evaluate" a mechanism: its objective value; its level of manipulability ε; and its revelation "cost" (e.g., communication complexity or cost of computing partial types).

The bound on manipulability could be fixed a priori, but in many settings it will be beneficial to optimize this as well. One could also incorporate ε into the designer's objective function, for example, by formalizing the various costs involved in manipulating the mechanism. This is, however, beyond the scope of this work, and we will here consider optimizing the mechanism's objective and its incentive properties separately. We therefore have three basic tasks when automatically designing a PRM: optimize the objective value for a fixed ε and partition; optimize ε for a fixed objective level and partition; and optimize the partition for a fixed mechanism. In this section we describe different objective functions one might consider, and the basic constraints that will be involved in the optimization. The following two sections describe the objective and manipulability optimization steps in the Bayesian and regret case respectively. We then describe the partition optimization step.

Objectives

An objective function $f(t, \mathbf{M})$ reflects the designer's value for mechanism $\mathbf{M} = (\Theta, m, p)$ when the agents' types are t. It often makes sense to decompose f as

$$f(t, \mathbf{M}) = f_m(t, m, \Theta) + f_p(t, p, \Theta) + c(\mathbf{M}).$$

Here f_m represents standard allocation-level objectives; for instance, if social welfare is being considered, we define:

$$f_m(t, m, \Theta) = \sum_{\mathbf{x}} m^{\mathbf{x}}_{\theta(t)} \cdot SW(\mathbf{x}; t)$$

where $m^{\mathbf{x}}_{\theta} = Pr(m(\theta) = \mathbf{x}|\theta)$ is the probability that allocation $\mathbf{x}$ is chosen when agents report θ. f_p corresponds to payment-level objectives such as revenue: $f_p(t, p, \Theta) = \sum_i p_i(\theta(t))$. Finally, $c(\mathbf{M})$ represents costs associated with the execution of the mechanism, for instance, a cost of α per bit required for agents to report their partial types: $c(\mathbf{M}) = -\alpha \log_2(|\Theta|)$. In what follows we assume f_m, f_p and c are multi-linear maps of (t, m, Θ), (t, p, Θ), and (m, p, Θ) respectively.

Given an objective function $f(t, \mathbf{M})$, we consider two approaches to PR-AMD: maximize the *expected objective value* of the mechanism; and minimize the *regret* of the mechanism with respect to the objective function.

Expected Objective Given a probabilistic prior over agent types $Pr(t)$, the expected objective value of a PRM (assuming truthful revelation) is well-defined:

$$\int_{t \in T} Pr(t) f(t, \mathbf{M}) dt = \sum_{\theta \in \Theta} Pr(\theta) \int_{t \in \theta} Pr(t|t \in \theta) f(t, \mathbf{M}) dt$$

For a fixed partition and mechanism, we can compute $Pr(\theta)$ and $\int_{t \in \theta} Pr(t|t \in \theta) f(t, \mathbf{M}) dt$, for all $\theta \in \Theta$.

Minimax regret When the prior information over agent types is not quantified probabilistically (e.g., we only have bounds on utility parameters), minimax regret is an appropriate decision criterion. Even when probabilistic information is available, minimax regret has been shown to be an excellent guide for the elicitation of preferences relevant to a particular decision problem [2; 4; 5]. Here the decision that is made is the choice of a mechanism. Let $\mathbb{M}(\Theta)$ be the set of partial revelation mechanisms with partition Θ. The pairwise regret of choosing mechanism $\mathbf{M}$ versus $\mathbf{M}'$, given that the type vector of the agents is t is:

$$R(\mathbf{M}, \mathbf{M}') = \max_{\theta \in \Theta} \max_{t \in T} f(t, \mathbf{M}') - f(t, \mathbf{M})$$

The max regret of $\mathbf{M}$ and the minimax regret optimal mechanism $\mathbf{M}^*$ are, respectively:

$$MR(\mathbf{M}) = \max_{\mathbf{M}' \in \mathbb{M}(\Theta)} R(\mathbf{M}, \mathbf{M}')$$

$$\mathbf{M}^* = arg \min_{\mathbf{M} \in \mathbb{M}(\Theta)} MR(\mathbf{M})$$

$R(\mathbf{M}, \mathbf{M}')$ represents the worst-case loss in objective value, over all possible realizations t of types. It can be thought of as the regret of choosing $\mathbf{M}$ and not $\mathbf{M}'$ when an adversary gets to pick the types of the agents. $MR(\mathbf{M})$ is the level of regret when the adversary also chooses $\mathbf{M}'$. For example, if the objective is to maximize social welfare, we have:

$$\begin{aligned} R(\mathbf{M}, \mathbf{M}') &= \max_{t \in T} SW(m'(t); t) - SW(m(t); t) \\ &= \max_{\theta \in \Theta} \max_{t \in \theta} \sum_{\mathbf{x}} (m'^{\mathbf{x}}_{\theta(t)} - m^{\mathbf{x}}_{\theta(t)}) \cdot SW(\mathbf{x}; t) \end{aligned}$$

Constraints

Given a mechanism $\mathbf{M} = (\Theta, m, p)$, the utility of agent i of type t_i for reporting θ_i, when others report θ_{-i} is

$$u_i(\theta_i|\theta_{-i}; t_i) = \sum_{\mathbf{x}} m^{\mathbf{x}}_{\theta} \cdot v_i(\mathbf{x}; t_i) - p_i(\theta).$$

ε-IR can be obtained by ensuring that $u_i(\theta_i(t_i)|\theta_{-i}(t_{-i}); t_i) \geq -\varepsilon, \ \forall i, \forall t_i, \forall t_{-i}$. For a fixed partition, when optimizing the mechanism variables, this is equivalent to the following linear constraints:

$$\sum_{\mathbf{x}} m^{\mathbf{x}}_{\theta} \cdot \min_{t_i \in \theta_i} v_i(\mathbf{x}; t_i) - p_i(\theta) \geq -\varepsilon, \ \forall \theta, \forall i$$

To obtain ε-dominant strategy IC we must have, $\forall i, \forall \theta_i \in \Theta_i, \forall t_i \in \theta_i, \forall \theta'_i \in \Theta_i, \forall \theta_{-i} \in \Theta_{-i}$:

$$u_i(\theta_i(t_i)|\theta_{-i}; t_i) \geq u_i(\theta'_i|\theta_{-i}; t_i) - \varepsilon.$$

If type space is continuous, this corresponds to an infinite number of constraints (one for each t_i). But these constraints are equivalent to $\forall \theta_i, \theta'_i, \theta_{-i}$:

$$\min_{t_i \in \theta_i} \left[\sum_{\mathbf{x}} (m^{\mathbf{x}}_{\theta} - m^{\mathbf{x}}_{\theta'}) \cdot v_i(\mathbf{x}; t_i) \right] + p_i(\theta') - p_i(\theta) \geq -\varepsilon$$

If the partition consists of upper and lower bounds on valuation parameters, since valuation functions are linear in t_i, it is sufficient to post one constraint for each vertex of θ_i.

For both IC and IR,, the number of constraints is thus proportional to the size of the partition Θ. If Θ is the Cartesian product of independently constructed agent partitions Θ_i, its size is exponential in the number of agents. Our partition optimization routine (described below), however, partitions type *vector* space directly, so that the size of Θ is equal to the number of steps in the optimization. In addition, to avoid posting one constraint for each partial type vector, one can use constraint generation to iteratively post the most violated constraint until none are violated. In practice, only a fraction of the constraints will be posted.

Bayesian PR-AMD

In this section, we consider PR-AMD with Bayesian objectives. We can define two sub-routines: one focused on increasing the expected objective of our mechanism, subject to satisfying ε-IR and ε-IC; the other to minimize the manipulability, ε, subject to achieving a specific expected objective level. The two can be combined with our partition refining algorithm in a number of ways, depending on the designer's goal.

For example, one could first initialize ε at some value and refine the type partition to increase expected objective value, subject to maintaining ε, until some stopping criterion is met. One could then hold this objective value to at least its terminal value and refine the partition to reduce the value of ε. Stopping criteria could include a target objective or ε level, or a specific number of bits (partial type limit) per phase. One could also alternate the sub-routines: fix an initial ε, optimize the objective subject to this ε, then reduce ε subject to this objective level, etc.

Our aim is not to argue for one particular approach, but instead to provide the tools for a designer to achieve her goal, whatever it may be.

Objective Maximization Given a fixed partition and a fixed ε, finding the ε-IR and ε-IC mechanism with highest expected objective can be formulated as the following LP:

$$\max_{\mathbf{M}} \sum_{\theta \in \Theta} Pr(\theta) \cdot g_{ob}(\theta, \mathbf{M}), \qquad \text{s.t. } \mathbf{M} \text{ is } \varepsilon\text{-IC and } \varepsilon\text{-IR}$$

where $g_{ob}(\theta, \mathbf{M}) = \int_{t \in \theta} Pr(t | t \in \theta) f(t, \mathbf{M}) dt$. When using our partition refining algorithm (discussed below), the probabilities of the partial type vectors , $Pr(\theta)$, can be computed iteratively, two at each step.

Manipulability Minimization Given a fixed partition and a fixed objective level F, we can construct the mechanism with least manipulability:

$$\begin{aligned} \min_{\mathbf{M}, \varepsilon} \ \ \varepsilon \quad & \text{s.t. } \mathbf{M} \text{ is } \varepsilon\text{-IC}, \varepsilon\text{-IR}, \\ & \text{and} \sum_{\theta \in \Theta} Pr(\theta) \cdot g_{ob}(\theta, \mathbf{M}) \geq F \end{aligned}$$

As previously mentioned, ε-IR and ε-IC constraints are linear in the mechanism variables, and since the partition is fixed, the two optimizations above are LPs. They can be solved directly, or as a sequence of smaller LPs using constraint generation.

Regret-based PR-AMD

In this section, we focus on PR-AMD where the designer wishes to minimize her regret over the choice of mechanism. The approach is similar to that of the previous section, with one notable difference. Imposing IC constraints on regret minimization optimization is inappropriate, since the adversary gets to choose a mechanism, as well as the actual realization of agent types. Taken together, this means the adversary is simply choosing an allocation and a payment vector to maximize the regret of M (since the mechanism and the type vector directly dictate the outcome). Thus any IC constraints imposed on the adversary's mechanism will be vacuous, and the meaning of the regret level achieved will be compromised. We therefore separate the mechanism optimization into two phases: one to reduce regret without any consideration of incentives up to some desired point, the other to reduce manipulability subject to maintaining the desired regret level.

Regret Minimization When the partition is fixed, the only variables in $\mathbf{M}$ (m and p) are functions that map partial type vectors to allocations and payments. We therefore have:

$$\begin{aligned} \min_{\mathbf{M}} \quad & \max_{\mathbf{M}'} \max_{t \in T} f(t, \mathbf{M}') - f(t, \mathbf{M}) \\ = \quad & \min_{\mathbf{M}} \max_{\mathbf{M}'} \max_{\theta \in \Theta} \max_{t \in \theta} f(t, \mathbf{M}') - f(t, \mathbf{M}) \\ = \quad & \max_{\theta \in \Theta} \min_{m(\theta), p(\theta)} \max_{\mathbf{M}'} \max_{t \in \theta} f(t, \mathbf{M}') - f(t, \mathbf{M}) \end{aligned}$$

The regret minimizing mechanism is thus a mechanism which, for each partial type vector picks the regret minimizing allocation and payments. This significantly reduces the size of the LPs involved. Although $|\Theta|$ is exponential in the number of agents, enumeration can be avoided, in practice, using constraint generation. Note that when using our partition refining algorithm (see below), the minimax regret of the mechanism can be computed iteratively, with only the regret values of two partial type vectors being computed at each step. Since individual rationality constraints are "local", i.e., each constraint applies to a unique partial type vector, adding them does not alter the formula above. Since incentive compatibility involves comparing decisions made for different reports, this would not be true if we added IC constraints as well. If one wanted to impose IC constraints, despite the fact that it makes the meaning of regret ambiguous, the optimization above can be computed as a sequence of (much larger) LPs using constraint generation. The resulting mechanism would be the ε-IC, ε-IR PRM with lowest possible regret.

Manipulability Minimization This phase is very similar to its Bayesian equivalent. Given a fixed partition and a fixed regret level δ, finding the mechanism with lowest ε can be formulated as the following LP:

$$\min_{\mathbf{M}, \varepsilon} \varepsilon \ \text{ s.t. } \mathbf{M} \text{ is } \varepsilon\text{-IC}, \varepsilon\text{-IR, and } \delta \geq MR(\mathbf{M})$$

This can be solved as a sequence of LPs using constraint generation.

Partition Optimization

To refine the partition, we use the partition optimization algorithm of [5]. The algorithm is based on decision tree construction techniques adapted to this setting for computational tractability. It is an iterative, myopic algorithm which, given a current partition, uses a heuristic function to select a partial type vector to split, and the dimension (agent and utility parameter) along which it will be split. In this fashion, communication complexity increases by only one partial type vector per step. We have so far only considered very simple heuristic functions for the case of revenue optimization. In the regret approach, the heuristic selects the vector with highest regret (with respect to revenue), and the

agent and utility parameter with highest degree of uncertainty, among all parameters involved in either our mechanism's choice of allocation or the adversary's choice. This is a simple application of the *current solution* heuristic (see [2] for further elaboration of heuristics for preference elicitation). In the Bayesian approach, the heuristic picks the partial type vector with highest probability, and the agent and utility parameter with highest uncertainty, among all parameters involved in the outcome with highest minimum social welfare.

When using our partition optimization algorithm, the size of Θ is equal to the number of partition-refining steps, and is thus not exponential in the number of agents. The LPs described in previous sections have, for each partial type vector, $|\mathbf{X}|$ allocation variables and n payment variables. After s steps, the total number of variables is therefore $s\cdot(|\mathbf{X}|+n)$. Although the number of steps necessary to achieve satisfactory objective and manipulability levels can be high, our algorithm avoids exponential complexity. Note that the complexity of PR-AMD with this algorithm is no greater than that of full revelation AMD, which only handles small finite type spaces. In fact this algorithm could easily be adapted to allow classical AMD to handle full revelation with much larger type spaces.

In the next section, we compare these heuristics to a uniform partitioning of type space that simply splits each partial type evenly across all parameters, one at a time.

Preliminary Empirical Results

As mentioned, the aim of this paper is to provide a mechanism designer with the tools necessary to satisfy her goals. Since we cannot explore the range of all possible goals a designer may have, we focus on just a few ways of using these tools to empirically test the efficacy of our various optimization and constraint generation routines. We focus on revenue maximization here as the underlying social choice objective, with exact IR, first for generic mechanism design problems, and then for the special case of one-item auctions.

General Mechanism Design

We first consider both PR-AMD approaches on generic mechanism design problems with two agents and three outcomes with randomly generated initial bounds as priors. In both the Bayesian and the regret approaches, we compare the performance of PR-AMD with our splitting heuristics with that of a uniform partitioning.

For Bayesian PR-AMD, we adopted the following framework: initially fix ε at 20% of the smallest (across agents and outcomes) average valuation given the prior bounds; first refine the partition to maximize expected revenue subject to this initial ε until we have increased the expected revenue achievable with no revelation by 150%; then minimize ε subject to this target expected revenue.

Figure 1 plots expected revenue and manipulability as a function of the number of bits used, for both heuristic and uniform partitioning. Clearly, even our simple heuristic significantly outperforms a uniform partitioning of type space. Note that with heuristic partitioning, with 12 bits of communication, revenue increases by slightly more than 150% because, during the manipulability reducing phase, the elicited information can be used to somewhat increase expected revenue even given the new best ε. Although manipulability appears to be decreasing very early on, this a result of averaging over several runs, with different starting points for the second phase. Manipulability is reduced by almost 45% with 12 bits using heuristic partitioning, even though only a few of those bits correspond to the ε reducing phase.

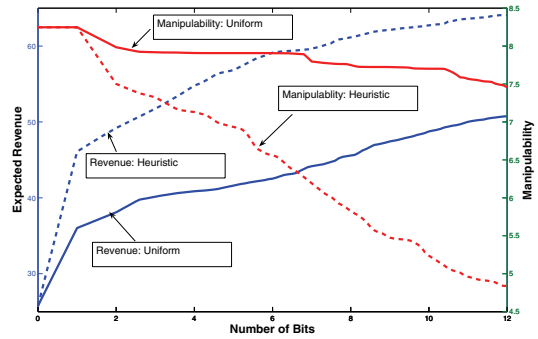

Figure 1: Expected revenue and worst case manipulability as a function of the number of bits used. Variable phase switch. Averaged over 20 runs.

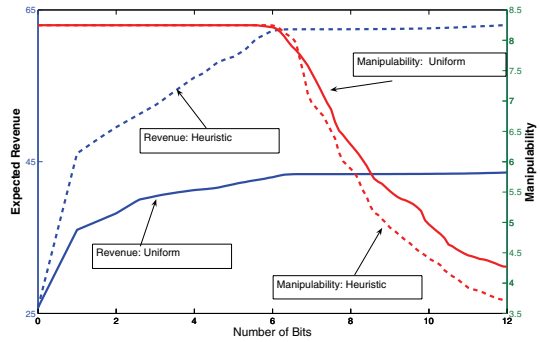

Figure 2: Expected revenue and worst case manipulability as a function of the number of bits used. Fixed phase switch. Averaged over 20 runs.

In order to more accurately compare each phase, we can decide to switch phases at a fixed number of bits. Figure 2 plots expected revenue and manipulability as a function of the number of bits used, for both heuristic and uniform partitioning, when the revenue-maximizing phase is run for six bits, and manipulability reduction is run for an additional six bits. Again, the first phase increases revenue by about 150% in 6 bits using our heuristic, with a large improvement over uniform partitioning (only about 75% increase). The second phase reduces manipulability by almost 55% (heuristic) versus about 45% (uniform) with six bits of communication. Although the reduction is impressive, the advantage of the heuristic approach is smaller in this phase. Thjis is because IC, unlike revenue or IR, is not a local property, defined for each partial type vector independently, but rather a global one that links type vectors together. There are therefore more "relevant" nodes (i.e. partial type vectors), and the uniform approach is at less of a disadvantage by not focusing

on a smaller number of nodes.

For regret PR-AMD, we perform the regret minimization phase until initial regret has been reduced by 50%, and then reduce ε subject to maintaining that regret level. We also computed, for interest, the lowest achievable ε given the current regret level during the first phase, and the lowest regret level given the current ε in the second. Figure 3 plots revenue-regret and manipulability as a function of the number of bits used, for both heuristic and uniform partitioning. Again, our simple heuristic significantly outperforms uniform partitioning. Interestingly, unlike in the Bayesian case, the information elicited in the second phase to reduce ε does not allow further improvement in regret. Hence the beginning of the second phase can be identified by the point where the regret curve flattens out. Since ε is unconstrained in the first phase, its value can both increase and decrease as regret is reduced. In the second phase, our heuristic approach reduces ε by about 70% with only an additional 6 bits of communication.

Depending on the setting and the costs involved in manipulating a mechanism, the final value of ε after 12 bits could be considered too high. This happens because, although our manipulability reducing phase is very efficient, the initial value of ε at the beginning of the phase is very high. If needed, this can be resolved by constraining the regret reduction phase with ε-IC constraints, for an appropriate value of ε. One possible approach could be to leave the initial value of ε unconstrained, but ensure, through IC constraints, that this value never increases during the regret reduction phase. In settings where the designer has a known specific tolerance for manipulability, another approach is simply to constrain ε to be less than that value, from the beginning.

For the non-increasing approach (NIε), we perform the regret minimization phase, with the additional IC constraints, until initial regret has been reduced by 50%, and then further reduce ε subject to maintaining that regret. In the second approach (Initε), we initially fix ε at 20% of the smallest (across agents and outcomes) average valuation given the prior bounds. Since this constrains ε at a much lower value than the first approach, reducing regret by 50% would require more bits. In order to more accurately compare each phase of both approaches, we would like the second phase to start roughly at the same number of bits. In the second approach, we therefore perform the regret-reducing phase while the number of bits used is less than 6, then switch to the ε-reducing phase for another 6 bits. This ensures that the second phase starts roughly at the same number of bits in both approaches.

Figure 4 plots revenue-regret and manipulability as a function of the number of bits used with a heuristic partitioning, for both of these approaches. Naturally, the more IC is constrained, the less quickly regret is reduced as there is less freedom to choose allocations and payments. As the figure shows, even with a highly constrained IC bound, regret can be reduced significantly (30% in 6 bits for Initε). And in settings where the required bound on manipulability is less precise, the non-increasing approach provides an excellent trade-off between regret and manipulability reduction (45% and 75% respectively in 12 bits). Also note that

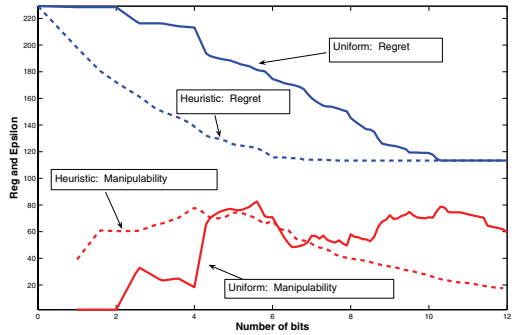

Figure 3: Regret wrt revenue and worst case manipulability as a function of the number of bits used. Heruistic vs. Uniform. Averaged over 20 runs.

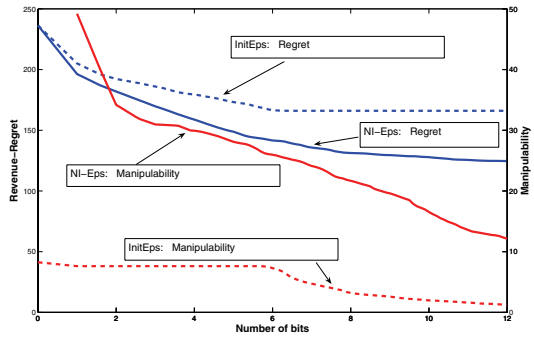

Figure 4: Regret wrt revenue and worst case manipulability as a function of the number of bits used. Non-increasing vs. initially constrained IC. Averaged over 20 runs.

when IC constraints are more flexible, it is possible to reduce both regret and manipulability simultaneously.

One-item Auctions

We also considered the special case of one-item auctions with two agents. In such single-parameter settings, implementation of PRMs with exact dominant incentive compatibility is possible. In fact, Blumrosen and Nisan [1] have proposed a class of PRMs for such auctions that is not only dominant-IC, but, with the appropriate partitioning of type space, provides the optimal expected revenue among all Bayes-Nash IC, ex-interim IR mechanisms with a fixed amount of communication per agent.

While we do not advocate using PR-AMD (with an $\varepsilon = 0$) in this restricted setting where priority games are known to be exact and optimal, we use this comparison to test the quality of our myopic heuristic partitioning algorithm. Figure 5 shows how expected revenue (given a uniform prior over $[0, 100]$) varies with communication for both revenue-optimal priority games and PR-AMD with $\varepsilon = 0$. Interestingly, the myopic behavior of our partitioning algorithm gives rise to a very small loss in revenue compared to the optimal partitioning of priority games: with anything over 1.5 bits of communication per agent, PR-AMD is within about 2.5% of the optimal revenue achievable with that many bits.

Although implementation with exact dominant IC is possible in one-item auctions, it is interesting to explore how

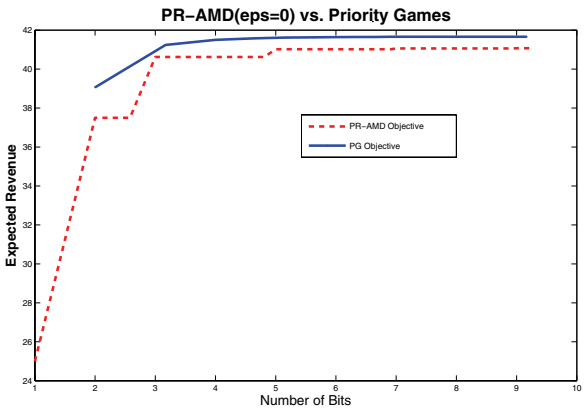

Figure 5: Expected revenue as a function of the number of bits, for the optimal priority game, and for Bayesian PR-AMD with $\varepsilon = 0$

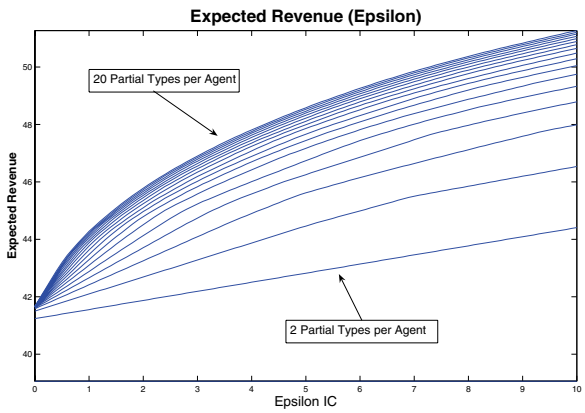

Figure 6: Expected revenue as a function of ε, for varying numbers of partial types per agent

much additional expected revenue could be obtained by allowing for varying amounts of manipulability. In order to remove the influence of our myopic partitioning, we can use the optimal priority game partition. Figure 6 plots revenue as a function of ε, where each curve represents a different number of partial types per agent (from 2 to 20). With five partial types per agent, allowing a manipulability level of only 2 increases expected revenue by about 10%.

Conclusion and Future Work

In this work we have provided a general framework for automatically designing partial revelation mechanisms with appropriate incentive properties, for any social choice objective. The PR-AMD problem consists of optimizing three relevant criteria: objective value, manipulability bound and communication requirement. Naturally optimal optimization of all three is intractable, but we show how it can be approximated by sequentializing it into three (linear) sub-optimizations. The framework is highly flexible so that the various routines developed here can be combined in different ways to allow a designer to make whichever trade-offs corresponds to her own preferences over mechanisms. Preliminary empirical results confirm the efficacy of our approach.

Apart from more extensive empirical evaluation of the techniques presented here, we would like to further investigate the use of more efficient splitting (i.e., partial type discovery) heuristics. The design of non-myopic incremental partial mechanisms is a natural future direction. Precisely determining the complexity of manipulation by formally modeling the costs involved to further justify our emphasis on approximate IC and IR is of special interest.

References

[1] L. Blumrosen, N. Nisan, and I. Segal. Auctions with severely bounded communication. *J. Artificial Intelligence Research*, 28:233–266, 2007.

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

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

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

[5] N. Hyafil and C. Boutilier. One-shot mechanism design with partial revelation. In *IJCAI-07*, pp. 1333–1340, Hyderabad, India, 2007.

[6] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, NY, 1995.

[7] R. Myerson. Optimal auction design. *Math. of Operations Research*, 6:58–73, 1981.

[8] N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices. *J. Economic Theory*, 129:192–224, 2006.

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

[10] T. Sandholm, V. Conitzer, and C. Boutilier. Automated design of multistage mechanisms. In *IJCAI-07*, pp. 1500–1506, Hyderabad, India, 2007.

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