Automated Design of Multistage Mechanisms[*]

Tuomas Sandholm
Carnegie Mellon University
Computer Science Department
sandholm@cs.cmu.edu

Vincent Conitzer
Duke University
Dept. of Computer Science
conitzer@cs.duke.edu

Craig Boutilier
University of Toronto
Dept. of Computer Science
cebly@cs.toronto.edu

Abstract

Mechanism design is the study of preference aggregation protocols that work well in the face of self-interested agents. We present the first general-purpose techniques for automatically designing *multistage* mechanisms. These can reduce elicitation burden by only querying agents for information that is relevant given their answers to previous queries. We first show how to turn a given (e.g., automatically designed using constrained optimization techniques) single-stage mechanism into the most efficient corresponding multistage mechanism given a specified elicitation tree. We then present greedy and dynamic programming (DP) algorithms that determine the elicitation tree (optimal in the DP case). Next, we show how the query savings inherent in the multistage model can be used to design the underlying single-stage mechanism to maximally take advantage of this approach. Finally, we present negative results on the design of multistage mechanisms that do not correspond to *dominant-strategy* single-stage mechanisms: an optimal multistage mechanism in general has to randomize over queries to hide information from the agents.

1 Introduction

In multiagent settings, often an *outcome* (e.g., presidents, joint plans, allocations of resources) must be chosen based on the preferences of a set of agents. Since the preference aggregator generally does not know these preferences *a priori*, the agents must report their preferences to the aggregator. Unfortunately, an agent may have an incentive to misreport its preferences in order to mislead the aggregator into selecting an outcome that is more desirable to the agent than the outcome that would be selected if the agent revealed its preferences truthfully.

Mechanism design is concerned with the creation of preference aggregation rules that lead to good outcomes in spite of such strategic behavior by agents. Classical mechanism design provides some general mechanisms, which, under certain assumptions, satisfy some notion of nonmanipulability and maximize some objective. Typically, such mechanisms do not rely on (even probabilistic) information about the agents' preferences (e.g., the Vickrey-Clarke-Groves (VCG) mechanism [29; 7; 12]), or can be easily applied to any distribution over preferences [11; 1; 21; 19]. However, these general algorithms only work in restricted settings (e.g., they may require the possibility of side payments, or work only for specific objectives) and may not reflect the designer's objectives.

Recently, *automated mechanism design (AMD)* has been proposed as a means to design mechanisms automatically for the setting at hand [8; 3]. This approach, based on constrained optimization, produces optimal special-purpose mechanisms even in settings for which no good general mechanisms are known, or for which impossibility results preclude the existence of good general mechanisms for the class of instances. While AMD is a relatively new approach, it has already started to be adopted in applications, for example, in strategic sourcing, reserve price setting in asset recovery, and recommender systems [14]. However, all prior work on general-purpose AMD has focused on single-stage mechanisms, in which all agents reveal their preferences completely and simultaneously. This is problematic for several reasons. First and foremost, agents may need to invest significant computational, cognitive or other resources to determine their preferences over outcomes [24; 25; 16]. For instance, when bidding on trucking tasks in a combinatorial reverse auction, an agent needs to solve, for each subset of tasks, a complex vehicle-routing problem). Second, the agents lose all privacy about their preferences. Third, it can require a large amount of communication. While the third reason applies only when the space of possible preferences is large, the first two reasons are significant even in settings with few outcomes or possible preference functions.

Much of this computation, communication, and privacy loss is unnecessary when certain aspects of an agent's preferences have no influence on the final outcome. For instance, if a second agent can perform a task at much lower cost than a first, we need not determine precisely how suboptimal assigning the task to the first agent is. Unfortunately, single-stage mechanisms cannot take advantage of this: we cannot *a priori* rule out the need to know the first agent's precise preferences

[*]This material is based upon work supported by the National Science Foundation under ITR grants IIS-0121678 and IIS-0427858, a Sloan Fellowship, and an IBM Ph.D. Fellowship.

for the task—this only becomes apparent after receiving information from the second.

Our solution is to use *multistage* mechanisms, where the aggregator queries the agents about certain aspects of their preferences, and chooses the next query to ask (and who to ask it of) based on answers to earlier queries. In a non-game-theoretic setting, a move to multistage protocols can yield an exponential savings in bits communicated [15]. In mechanism design settings, such a move can yield an exponential savings in communication and the aggregator's computation [10].

Multistage mechanisms have been *manually* designed for several applications, such as voting [9], single-item auctions (e.g., [5]), and combinatorial auctions (see reviews [26; 22]). Automated design of multistage mechanisms has been addressed *for specific settings* in parallel with our work [17; 27; 13]. Prior work has studied the design of multistage mechanisms in strategic multi-party computation [28; 2], but the issues in that setting (e.g., that an agent may be tempted not to invest the effort necessary to determine its private information) are very different from those we address. In this paper, we introduce the first *general* techniques for *automated* design of multistage mechanisms. We adopt a very specific methodology: first design a single-stage mechanism using existing techniques for AMD; this is then converted into a multistage mechanism using one of several techniques we propose. We also show how the multistage model can be used to to influence the design the underlying single-stage mechanism to maximally take advantage of the savings inherent in the multistage model. As such, relying on an intermediate single-stage mechanism incurs no loss.

2 The model

2.1 Automated design of single-stage mechanisms

In this subsection, we review the relevant definitions and results from the single-stage AMD literature. In a single-stage AMD setting, we are given: 1) a finite set of outcomes O (payments to/from agents can be part of the outcome); 2) a finite set of N agents; 3) for each agent i, (a) a finite set of types Θ_i, (b) a probability distribution γ_i over Θ_i (in the case of correlated types, there is a single joint distribution γ over $\Theta_1 \times \ldots \times \Theta_N$), and (c) a utility function $u_i : \Theta_i \times O \rightarrow \mathbb{R}$;4) an objective function $g : \Theta_1 \times \ldots \times \Theta_N \times O \rightarrow \mathbb{R}$ whose expectation the designer wishes to maximize. Note that utility functions are parameterized by type; while the u_i are common knowledge, the types encode (private) preferences [18]. The restriction to a finite type space is somewhat limiting. However, continuous spaces can be handled via suitable discretization of the type space.[1] Possible designer objectives are many (e.g., *social welfare*, or maximizing the sum of agent utilities for the chosen outcome).

By the *revelation principle* [18], we can restrict attention to *truthful, direct revelation mechanisms*, where agents report their types directly and never have an incentive to misreport them. In general, mechanisms may choose the outcome randomly. Thus, a mechanism consists of a distribution selection function $p : \Theta_1 \times \ldots \times \Theta_N \rightarrow \Delta(O)$, where $\Delta(O)$ is the set of probability distributions over O. A mechanism is a *dominant strategy mechanism* if truthtelling is optimal regardless of what other agents report. In other words, for any agent i, type vector $(\theta_1, \ldots, \theta_i, \ldots, \theta_N)$, and alternative report $\hat{\theta}_i \in \Theta_i$, we have $E_{o|\theta_1,..,\theta_i,..,\theta_n} u_i(\theta_i, o) \geq E_{o|\theta_1,..,\hat{\theta}_i,..,\theta_n} u_i(\theta_i, o)$. If telling the truth is optimal only *given* that the other agents are truthful, we have a *Bayes-Nash equilibrium (BNE)* mechanism. That is, in a BNE mechanism, for any i, $\theta_i \in \Theta_i$, and alternative report $\hat{\theta}_i \in \Theta_i$, we have $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i} E_{o|\theta_1,..,\theta_i,..,\theta_n} u_i(\theta_i, o) \geq E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_N)|\theta_i} E_{o|\theta_1,..,\hat{\theta}_i,..,\theta_n} u_i(\theta_i, o)$.

In settings where participation is voluntary, the AMD formulation also includes participation (or *individual rationality*) constraints: no agent is worse off participating in the mechanism than not. Techniques for handling them in single-stage AMD can be applied to our multistage case without modification.

Given that the mechanism is allowed to choose the outcome at random, the problem of designing an optimal single-stage mechanism can be solved in polynomial time (given that the number of agents is constant) using linear programming [8]. The decision variables of that linear program are the following: for every type vector θ and every outcome o, there is a decision variable $p(\theta, o)$ that determines the probability of that outcome given that type vector. It is straightforward to check that the incentive compatibility constraints above, as well as the expectation of the objective, are linear functions of these variables, which gives us the linear program. Generating and solving this linear program is all that is required to have a basic approach to automatically designing single-stage mechanisms, and it is in fact the approach that we use in this paper to generate single-stage mechanisms.

2.2 Automated design of multistage mechanisms

In multistage AMD, the input includes—in addition to the input for single-stage AMD—a set of queries Q and a set of answers A. Assuming a single answer set (rather than distinct A_q for each query q) comes without loss of generality. One set of special interest is $A = \{\text{yes, no}\}$. Each query q is associated with a particular agent i (of whom q would be asked),[2] and the answer that the agent would give to q (when answering truthfully) is given by the function $a : Q \times \Theta_i \rightarrow A$, where $a(q, \theta_i)$ is i's truthful answer to query q when i's type is θ_i. This implies that there is only one truthful response to any $q \in Q$; thus, each query partitions the agent's type space. Upon receiving answer a to q from agent i, the mechanism can infer (assuming truthfulness) i's type is in $\{\theta_i \in \Theta_i : a(q, \theta_i) = a\}$.

A multistage mechanism M *corresponds* to a given single-stage mechanism S if, for each type vector θ reported by

[1]The discretization can be fixed in advance with an analysis of its impact on incentives and efficiency (as in recent research on limited revelation auctions [4]). Or, it may be optimized within the AMD model itself; this latter point is the subject of current research.

[2]In this paper we will restrict our attention to the case where we query one agent at a time; however, our approach is easily extended to settings where we query multiple agents at the same time. We note, however, that querying agents one at a time leads to the largest possible savings in the number of queries.

the agents, both M and S choose each outcome o with the same probability. Suppose M corresponds to some S where truth-telling is dominant. It is not hard to see that M has truthtelling as an *ex-post* equilibrium, regardless of the the results of previous queries revealed. (A vector of strategies is an ex-post equilibrium if for each agent, following the strategy is optimal regardless of the types of the other agents given their strategies.) That is, truth-telling is optimal (regardless of an agent's beliefs) whenever all other agents answer queries truthfully.This implies that we never need to randomize over query choice (though this no longer holds if S is not a dominant-strategy mechanism, as we will see later). Ex post implementation is weaker than dominant strategies, but stronger than BNE. Note that even if S is a dominant-strategy mechanism, M need not be: if an agent makes her answer dependent on the history of queries asked, another agent may have an incentive to lie about her type in order to influence which queries the former is asked.

For these reasons, apart from the last technical section in the paper, we focus exclusively on multistage mechanisms that correspond to dominant-strategy single-stage mechanisms. Thus, we can restrict ourselves to mechanisms that select the next query deterministically based on answers to prior queries; moreover, we need not worry about incentives.

Under these restrictions, a *multistage mechanism* is defined by: 1) a tree with nodes V and edges E; 2) for each internal (non-leaf) node v, an agent i and a query q to that agent; 3) a one-to-one correspondence between possible answers to the query at node v and children of node v; 4) for each node v and outcome o, a probability that, given that we reach v, we stop asking queries and choose outcome o. (In the case where v is a leaf, these probabilities must sum to one.) An *elicitation tree* is a multistage mechanism without outcome probabilities. We denote by I_v the *information set* at node v (i.e., the set of type vectors consistent with the answers that lead to v). We generally assume an elicitation tree is *complete*: I_l is a single type vector for any leaf l.[3]

We study several variants of multistage AMD. We consider the possibility starting with a *given* single-stage mechanism (e.g., computed by single-stage AMD software) and turning it into a corresponding multistage mechanism, as well as allowing our multistage perspective to influence the choice/design of the underlying single-stage mechanism. We also consider multistage design when the elicitation tree (hence the query order) is given beforehand, and when we impose no constraints on the form of the tree.

3 A small example

In this section, we illustrate various notions for automatically designing multistage mechanisms using a simple example. Suppose a divorcing couple jointly owns a painting, and an arbitrator has to decide the fate of the painting. There are 5 options: (1) the husband keeps the painting; (2) the wife keeps it; (3) the painting remains jointly owned, but is hung in a museum; (4) it is cut into pieces which are given to the husband; and (5) it is cut up with pieces given to the wife. The husband and wife each have two possible types: type L ("Low") is associated with relative indifference toward the painting, and type H ("High") with deep attachment. Each has type L with probability 0.8 and type H with probability 0.2. To maximize social welfare, the arbitrator would like to give the painting to whomever cares for it more; but since a party who cares little would prefer having it over not, the arbitrator must design appropriate incentives to ensure truthful reporting. The utility function for each party is the "same." Keeping the painting gives utility 2 (type L) or 100 (H). The other party getting the painting gives utility 0 (for either type). The museum outcome gives utility 1.5 (L) or 40 (H). Receiving pieces gives utility -9 while not even getting the pieces gives utility -10 (for either type).[4]

Our goal is to find a dominant-strategy (possibly randomized) mechanism (without payments) that maximizes expected social welfare. First we find the optimal single-stage mechanism. Solving this example using the methodology described earlier yields the following randomized mechanism (the probabilities are rounded):

	wife L	wife H
husband L	Museum	0.96 Wife keeps; 0.04 Husband gets pieces

husband H	0.96 Husband keeps;
	0.04 Wife gets pieces
		0.47 Husband keeps;
		0.40 Wife keeps;
		0.13 Wife gets pieces

In spite of the symmetry between the husband and the wife, the mechanism is asymmetric. Of course, other optimal solutions exist (e.g., where the roles of husband and wife are interchanged).

Now we consider how to turn this single-stage mechanism into a corresponding multistage mechanism (i.e., with the same outcome probabilities). First, suppose the elicitation tree is given, with the wife's type elicited first. Fig. 1 shows the optimal multistage mechanism. (Why this is so will become apparent.) This mechanism saves one query with probability $0.2 \cdot 0.4 = 0.08$.

If the elicitation tree (query order) is not fixed, the optimal mechanism is that given in Fig. 2. It turns out that greater savings can be obtained by eliciting the husband's type first: this mechanism saves a query with probability $0.2 \cdot (0.04 + 0.47) = 0.10$. (This is due to the asymmetry of the single-stage mechanism from which we are starting.)

Suppose queries to the husband are slightly more expensive than those to the wife, so that we would rather save on husband queries (unlike the previous mechanism). If we allow a different optimal single-stage mechanism, namely the analog of the one above with the husband and wife roles switched (which remains optimal due to the problem symmetry), then the optimal multistage mechanism that corresponds to this

[3]This does not imply that the mechanism will ask all queries and uniquely determine a type vector: the concrete outcome probabilities, specifically, the possibility of terminating at an interior node will typically preclude this.

[4]This problem has some similarity to King Solomon's dilemma; however, when that dilemma is discussed in the economics literature [23], it is assumed that there is only one rightful mother, and both women know who it is—unlike our problem, where the agents do not know each others' types.

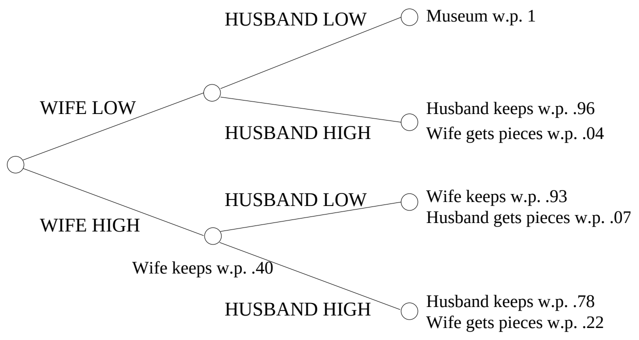

Figure 1: The optimal elicitation tree given the single-stage mechanism and given that the wife is queried first. When an internal node has a probability-outcome pair associated with it, we terminate early at that node with that probability, with that outcome; with the remaining probability, we move on to the next query.

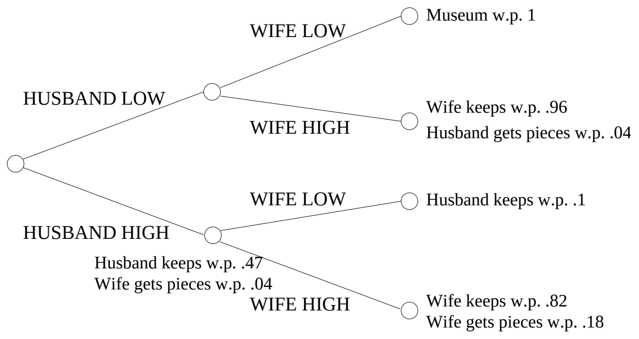

Figure 2: The optimal mechanism when asking the husband first.

saves a query to the husband (rather than the wife) with probability 0.10, giving greater cost savings.

Finally, if we are willing to sacrifice optimality of the single-stage mechanism to obtain greater query savings, this may again change the mechanism. For example, if we make the cost of querying sufficiently large, it will be optimal to not ask any queries, and always choose the same outcome.

One interesting additional motivation for automatically designing multistage mechanisms is that the tree-based representation of a multistage mechanism may be *easier to understand* for a human than the tabular form of a single-stage mechanism—especially if the tree is relatively small.

The trees in the figures are reminiscent of decision trees in machine learning [20] and boolean function representation. In decision tree learning, the tree classifies an example based on the answers to a sequence of queries about features of the example. Similarly, a multistage mechanism determines the outcome for a particular type vector based on the answers to a sequence of queries about the agents' types. There are significant differences, however, in the form of the trees. For example, decision trees typically do not use randomization in determining the classification. Moreover, decision trees never determine the classification at an internal node of the tree, whereas we probabilistically terminate the querying process in our trees.[5] Our tree transformations do have an analog in decision tree learning, where reordering is often considered for the purposes of simplicity and generalization [6]; but again the motivation, details, and meaning of such transformations are quite distinct.

4 Converting a single-stage mechanism into a multistage mechanism

In this section we develop methods for converting a given (e.g., automatically designed) single-stage mechanism into an equivalent multistage mechanism which saves on elicitation costs. The analog of this in decision trees would be constructing a (short) decision tree, given that, for each complete instantiation of all the features, we have already decided how we want to classify examples with that instantiation. In the first subsection we develop methods for the case where the elicitation tree (query order) is given. In the second subsection we generalize the approach to the case where the elicitation tree is not given, but can be chosen endogenously.

4.1 Given elicitation tree

We first solve the simplest of our problems: converting a single-stage mechanism into the most efficient multistage mechanism for a given elicitation tree. This problem can be motivated by considering exogenous constraints on query order (e.g., agents available at different times, or when the optimal ordering is readily available). More importantly, this setting serves as a stepping stone to more general techniques below. Our key technique is to "propagate up" probability from the leaves to internal nodes where this is possible.

Lemma 1 *Let multistage mechanism M correspond to single-stage mechanism S. Suppose that for some internal node v in the elicitation tree (with exit probability e_v) and outcome o, all the leaves of the subtree T_v rooted at v assign a probability of at least $p > 0$ to outcome o (and none of the internal nodes of T_v have any exit probability). Then the following modification M' of M corresponds to S: (1) At node v, exit with o with probability $(1 - e_v)p$; (2) Subtract p from the probability assigned to o at each leaf of T_v; (3) Divide all the outcome probabilities at leaves T_v by $1 - p$.*

Proof: Consider the probability $p(\theta, o')$ that outcome o' will be selected given type vector θ in M'. If θ does not lead to v, clearly $p(\theta, o')$ is the same in M and M'; so assume that it does. If $o' = o$ (the outcome we exit early with), then the probability of selecting o' at v is now the early-exit probability p, plus the probability that we do not exit early but choose outcome o' later, which is $(1 - p)(p(\theta, o')^{old} - p)/(1 - p) = p(\theta, o')^{old} - p$. Hence the total probability is $p(\theta, o')^{old}$; i.e., it did not change. If $o' \neq o$, then the probability of selecting o' at v is the probability that we do not exit early with o and choose outcome o' later, which is $(1 - p)(p(\theta, o')^{old})/(1 - p) = p(\theta, o')^{old}$. Hence for any θ, M and M' select o' with the same probability. ■

[5]Randomized decision trees in circuit complexity literature can in some sense be viewed in this light, though the representation is usually as a distribution of "deterministic" trees.

We note that the ability to propagate probability up in this manner even when the distributions at the leaves are not identical makes this different from the standard framework in communication complexity theory. (In addition, we may have a restricted query language, and we have a prior distribution over the inputs.)

If we propagate up as much probability as possible, we obtain the optimal mechanism (for a given S and tree):

Theorem 1 *Suppose we apply Lemma 1 to every pair (v, o), starting at the root and working our way down to the leaves. Then the resulting multistage mechanism saves the most queries (or, in the case of different query costs, the greatest query cost) among multistage mechanisms corresponding to the given single-stage mechanism and the given elicitation tree.*

As an example, we derive the mechanism of Fig. 2. We start from a mechanism that saves no queries (Fig. 3).

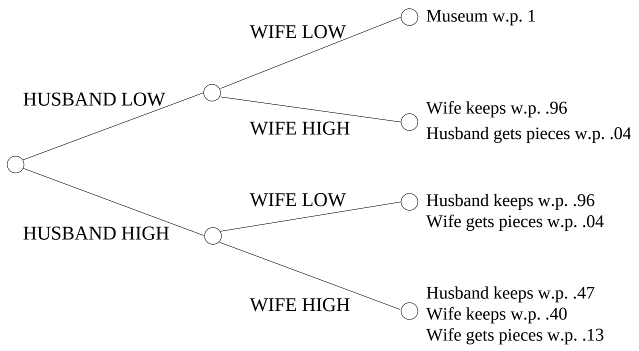

Figure 3: Multistage mechanism that saves no queries at all.

At the node after the husband reports "high", the husband keeps the painting with $p \geq .47$ in all subsequent leaves. So we can propagate this probability up (Fig. 4).

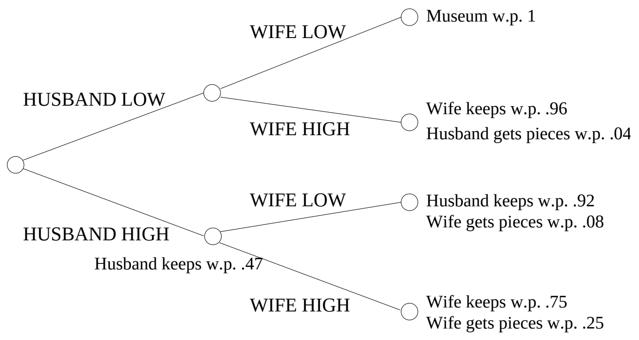

Figure 4: Some probability propagated up.

At the same node, the wife gets the pieces of the painting with $p \geq .08$ in all subsequent leaves. Propagating this up results in the mechanism of Fig. 2.

The following corollary characterizes the probability of exiting early at or before a given node. This will be helpful in our use of the "propagating probabilities up" technique within all of the algorithms discussed later in the paper.

Corollary 1 *In a multistage mechanism that saves a maximum number of queries, for any type vector θ such that node v will be reached if the mechanism does not exit early, the probability that we will reach v and not exit early at v, given that θ is the type vector, is $1 - \sum_{o \in O} \min_{\theta \in I_v} p(\theta, o)$. Hence, given that θ is the type vector and we have not exited early at or before node v, and we transition from node v to node w, the probability of exiting early at node w is*
$1 - (1 - \sum_{o \in O} \min_{\theta \in I_w} p(\theta, o))/(1 - \sum_{o \in O} \min_{\theta \in I_v} p(\theta, o))$.

4.2 Endogenously determined elicitation tree

In this section we develop methods for converting a single-stage mechanism into a multistage one, *without constraints on the elicitation tree (query order).* We first provide a greedy algorithm, and show two ways in which it can "fail" (i.e., yield an arbitrarily small fraction of the query savings available). We then give an optimal dynamic program.

Greedy algorithm

Our greedy algorithm chooses the query at each stage so as to maximize the probability of being able to exit immediately after this query given the preceding queries and responses. Letting $U(I, q, a_q)$ denote the information state that results from being in information state I and then receiving answer a to query q, we define the algorithm as follows.

Definition 1 *The* greedy algorithm *chooses the query to ask at node v from the set*
$\arg\max_{q \in Q} \sum_{a \in A} P(a|I_v, q) \sum_{o \in O} \min_{\theta \in U(I_v, q, a)} p(\theta, o)$.

The greedy algorithm does what we intend:

Theorem 2 *The greedy algorithm chooses a query that maximizes the probability of exiting immediately after it.*

Theorem 3 *The greedy algorithm chooses the query for node v in time $O(|Q| \cdot |A| \cdot |O| \cdot |\Theta|)$.*

Unfortunately, the greedy algorithm can be arbitrarily far from optimal (even when all queries have equal cost):

Proposition 1 *There exist single-stage mechanisms S for which the greedy algorithm achieves only an arbitrarily small fraction of the possible query savings (even when S is deterministic, there are only three players, two types per player, and three outcomes; alternatively, even when priors over types are uniform, there are only three players, two types per player, and five outcomes).*

This is a worst-case result; it is likely that the greedy algorithm will perform quite well in practice.

Dynamic programming algorithm

Unlike the greedy algorithm, the dynamic program must build the entire tree. The program works by computing, for *every* possible information state I, the minimum possible expected number of queries $n(I)$ from that point on, *given* that we have not exited early. As before, let $U(I, q, a)$ be the information state that results from receiving answer a to q at

I. Let $e(I, q, a)$ be the probability of exiting immediately after receiving answer a to q at I, given that we did not exit early at I. By Corollary 1, we can compute $e(I, q, a)$ as $1 - \frac{1 - \sum_{o \in O} \min_{\theta \in U(I,q,a)} p(\theta, o)}{1 - \sum_{o \in O} \min_{\theta \in I} p(\theta, o)}$. We obtain the recurrence

$$n(I) = min_{q \in Q} c(q) + \sum_{a \in A} P(a|I, q)(1 - e(I, q, a))n(U(I, q, a))$$

Using the fact that $n(\{\theta\}) = 0$ for every type vector θ, we use this recurrence to compute the value of $n(I)$ for every I, starting with the small I and working up to larger ones.

Theorem 4 *The dynamic programming algorithm computes the value of $n(I)$ for all I in time $O(|Q| \cdot |A| \cdot |O| \cdot |\Theta| \cdot 2^{|\Theta|})$.*

We can retrieve the optimal multistage mechanism from this as follows: when we arrive at information state I and do not exit early, choose a query from $\arg\min_{q \in Q} \sum_{a \in A} P(a|I, q)(1 - e(I, q, a))n(U(I, q, a))$.

5 Designing optimal multistage mechanisms

So far we have discussed how a given single-stage mechanism can be converted into an equivalent multistage mechanism. Here we will no longer take the single-stage design as a constraint. We develop a method for designing the single-stage mechanism in such a way that we get large savings in queries when we transform it into a multistage mechanism using the techniques described earlier. We focus on the case where the elicitation tree (query order) is given. It turns out that, using Corollary 1, we can directly integrate the eventual query savings into the linear programming formulation for AMD described earlier.

We say that node v is *on the elicitation path* for type vector θ if θ would lead us to ask the query at v (given that we do not exit early). For every internal node v in the tree, we add a term to the AMD objective (which maximizes the designer's objective) that indicates the probability of saving the query corresponding to this node.[6] (We say that we *save* the query corresponding to v when v is on the elicitation path, but we exit early at or before v.) Thus, the term in the objective for v is $c(v)P(v)e(v)$ where $c(v)$ is the cost of the query at node v, $P(v)$ is the probability of v being on the elicitation path, and $e(v)$ is the probability that we will exit early at or before v, given that v is on the elicitation path. $P(v)$ is a constant, but $e(v)$ is a variable that depends on how we set the outcome probabilities for the leaves. Specifically, by Corollary 1, we know that $e(v) = \sum_{o \in O} \min_{\theta \in S_v} p(\theta, o)$. The $\min$ operator is not linear, so we cannot add this expression to the LP objective directly. We work around this by letting $e(v) = \sum_{o \in O} e(v, o)$, where $e(v, o)$ is the probability of exiting early at or before v with outcome o, given that v is on the elicitation path. Then, for every $o \in O$ and $\theta \in S_v$, we add the constraint $e(v, o) \leq p(\theta, o)$.

[6]Suitable scaling to ensure commensurability with the designer's objective is straightforward; however, this does assume query costs can be accounted for additively.

Because linear programs can be solved to optimality in polynomial time, and the formulation above is polynomial in the number of outcomes and the number of types per agent (but not in the number of agents), the following theorem follows immediately:

Theorem 5 *The extension of the single-stage AMD formulation described above computes the optimal multistage mechanism for the given elicitation tree, taking query costs into account, in time polynomial in the number of outcomes and the number of types per agent (but not in the number of agents).*

This also begets an (inefficient) algorithm for generating the optimal multistage mechanism when neither the single-stage mechanism nor the elicitation tree is given: apply the above algorithm to every possible elicitation tree.

6 Mechanisms without dominant strategies

So far, we have restricted our study to multistage mechanisms whose single-stage correspondents have truth-telling as a dominant strategy. As discussed, this is helpful because in such multistage mechanisms, telling the truth is an ex-post equilibrium, so we need not worry that information revealed to agents by the mechanism will introduce strategic behavior. Nevertheless, we may also be interested in converting single-stage mechanisms that do not have dominant strategies, such as BNE mechanisms, to multistage mechanisms (e.g., because such mechanisms can achieve a higher objective value than dominant-strategy mechanisms).

Here we present initial results on converting BNE mechanisms into multistage mechanisms. These results are negative: they show that restricting ourselves to particular natural classes of multistage mechanisms may come at a loss of optimality. Thus, to design optimal multistage mechanisms, we need to search a broader space of mechanisms.

Proposition 2 *Even when the primary objective is social welfare and we use BNE as our solution concept, there exist settings in which immediately revealing the result of every query incurs a loss in objective value.*

The next result that we establish is that restricting ourselves to mechanisms that always choose the next query deterministically can come at a loss.

Proposition 3 *There exist settings in which:*

1. *The primary objective is social welfare;*
2. *The optimal single-stage BNE incentive compatible mechanism is unique;*
3. *The unique optimal (in terms of query savings) elicitation tree to ask the queries for this mechanism is not (even BNE) incentive compatible;*
4. *There exists an elicitation tree for this mechanism that randomizes over the next query selected, is (BNE) incentive compatible, and has almost the same query savings as the optimal elicitation tree (and thus strictly greater query savings than any deterministic (BNE) incentive-compatible elicitation tree for this mechanism).*

A potential alternative to randomization by the mechanism is to obtain the randomization from mixed (i.e., randomized) strategies of the agents in mechanisms that are not truthful direct-revelation mechanisms.

7 Conclusions

We extended the constrained-optimization based techniques for automated mechanism design to the design of multistage mechanisms, allowing reduction in elicitation burden by querying agents sequentially, and only querying them for information that is relevant given previous query responses. We focused primarily on the design of multistage mechanisms that correspond to dominant-strategy single-stage mechanisms, since these ensure truth-telling is an *ex-post* equilibrium (no matter what is revealed about other agents' answers). We described several techniques for converting single-stage mechanisms into multistage, both with and without fixed elicitation trees, and also showed how to augment single-stage AMD to produce single-stage mechanisms that can be maximally exploited in the conversion to multistage. Finally, we presented negative results on the design of multistage mechanisms that do not correspond to dominant-strategy single-stage mechanisms.

References

[1] K. Arrow. The property rights doctrine and demand revelation under incomplete information. In M Boskin, ed., *Economics and human welfare*. Academic Press, New York, 1979.

[2] G. Bahar and M. Tennenholtz. Sequential-simultaneous information elicitation in multi-agent systems. *IJCAI*, pp.923–928, 2005.

[3] A. Blumberg and A. Shelat. Searching for stable mechanisms: Automated design for imperfect players. *AAAI*, pp.8–13, 2004.

[4] L. Blumrosen and N. Nisan. Auctions with severely bounded communication. *FOCS*, pp.406–415, 2002.

[5] L. Blumrosen, N. Nisan, and I. Segal. Multi-player and multi-round auctions with severely bounded communication. *ESA*, pp.102–113, 2003.

[6] M. Bohanic and I. Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15:223–250, 1994.

[7] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

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

[9] V. Conitzer and T. Sandholm. Vote elicitation: Complexity and strategy-proofness. *AAAI*, pp.392–397, 2002.

[10] V. Conitzer and T. Sandholm. Computational criticisms of the revelation principle. *LOFT*, 2004.

[11] C. d'Aspremont and L. Gérard-Varet. Incentives and incomplete information. *J. Public Economics*, 11:25–45, 1979.

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

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

[14] R. Jurca and B. Faltings. Minimum payments that reward honest reputation feedback. *ACM-EC*, 2006.

[15] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[16] Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions. In *Theoretical Aspects of Rationality and Knowledge*, pages 169–182, 2001.

[17] Anton Likhodedov and Tuomas Sandholm. Mechanism for optimally trading off revenue and efficiency in multi-unit auctions. *ACM-EC*, 2004. Short paper.

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

[19] E. Maskin and J. Riley. Optimal multi-unit auctions. In F. Hahn, ed., *The Economics of Missing Markets, Information, and Games*, Ch.14, pp.312–335. Clarendon Press, Oxford, 1989.

[20] S. K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[21] R. Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[22] D. Parkes. Iterative combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, eds., *Combinatorial Auctions*, Ch.3. MIT Press, 2006.

[23] M. Perry and P. J. Reny. A general solution to King Solomon's dilemma. *Games and Economic Behavior*, 26:279–285, 1999.

[24] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. *AAAI*, pages 256–262, 1993.

[25] Tuomas Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000.

[26] T. Sandholm and C. Boutilier. Preference elicitation in combinatorial auctions. In P. Cramton, Y. Shoham, R. Steinberg, eds., *Combinatorial Auctions*, Ch.10, pp.233–263. MIT Press, 2006.

[27] Tuomas Sandholm and Andrew Gilpin. Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. *AAMAS*, pages 1127–1134, 2006.

[28] R. Smorodinsky and M. Tennenholtz. Sequential information elicitation in multi-agent systems. *UAI*, pp.528–535, 2004.

[29] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J. Finance*, 16:8–37, 1961.