Regret-based Incremental Partial Revelation Mechanisms

Mark Crowley

Department of Computer Science
University of British Columbia

Game Theory and Decision Theory Reading Group - October 3, 2006

Outline

Introduction

- eliciting complete type information is increasingly difficult for complex domains
- define a system for describing mechanisms with partial revelation of types
- want revelation of type to be acquired incrementally
- use global regret to unify allocation and payment uncertainty
- want to approximate VCG payments without destroying incentive compatibility

Mechanism Design

Notation

$\mathbf{x} \in \mathbf{X}$ an outcome or allocation

$i \in n$ agent i

$t_i \in T_i$ type of agent i, encodes utility. $T = \{\text{set of all } T_i \text{ vectors}\}$

$v_i(\mathbf{x}; t_i)$ value to agent i of outcome $\mathbf{x}$ given type t_i

$SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$ Social Welfare is sum of all agents' values for the outcome given their type. $SW_{-i}(\mathbf{x}; t)$ is the SW based on the values of everyone except agent i

Mechanism Design

Notation

$\mathbf{x} \in \mathbf{X}$ an outcome or allocation

$i \in n$ agent i

$t_i \in T_i$ type of agent i, encodes utility.
$T = \{\text{set of all } T_i \text{ vectors}\}$

$v_i(\mathbf{x}; t_i)$ value to agent i of outcome $\mathbf{x}$ given type t_i

$SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$ Social Welfare is sum of all agents' values for the outcome given their type. $SW_{-i}(\mathbf{x}; t)$ is the SW based on the values of everyone except agent i

Mechanism Design

Notation

$\mathbf{x} \in \mathbf{X}$ an outcome or allocation

$i \in n$ agent i

$t_i \in T_i$ type of agent i, encodes utility. $T = \{\text{set of all } T_i \text{ vectors}\}$

$v_i(\mathbf{x}; t_i)$ value to agent i of outcome $\mathbf{x}$ given type t_i

$SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$ Social Welfare is sum of all agents' values for the outcome given their type. $SW_{-i}(\mathbf{x}; t)$ is the SW based on the values of everyone except agent i

Mechanism Design

Notation

$\mathbf{x} \in \mathbf{X}$ an outcome or allocation

$i \in n$ agent i

$t_i \in T_i$ type of agent i, encodes utility. $T = \{\text{set of all } T_i \text{ vectors}\}$

$v_i(\mathbf{x}; t_i)$ value to agent i of outcome $\mathbf{x}$ given type t_i

$SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$ Social Welfare is sum of all agents' values for the outcome given their type. $SW_{-i}(\mathbf{x}; t)$ is the SW based on the values of everyone except agent i

Mechanism Design

Notation

$\mathbf{x} \in \mathbf{X}$	an outcome or allocation
$i \in n$	agent i
$t_i \in T_i$	type of agent i, encodes utility. $T = \{\text{set of all } T_i \text{ vectors}\}$
$v_i(\mathbf{x}; t_i)$	value to agent i of outcome $\mathbf{x}$ given type t_i
$SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$	Social Welfare is sum of all agents' values for the outcome given their type. $SW_{-i}(\mathbf{x}; t)$ is the SW based on the values of everyone except agent i

Mechanism Design

Notation

$\mathbf{x} \in \mathbf{X}$ an outcome or allocation

$i \in n$ agent i

$t_i \in T_i$ type of agent i, encodes utility.
$T = \{\text{set of all } T_i \text{ vectors}\}$

$v_i(\mathbf{x}; t_i)$ value to agent i of outcome $\mathbf{x}$ given type t_i

$SW(\mathbf{x}; t) = \sum_i v_i(\mathbf{x}; t_i)$ Social Welfare is sum of all agents' values for the outcome given their type. $SW_{-i}(\mathbf{x}; t)$ is the SW based on the values of everyone except agent i

Mechanism Design

Definition

A mechanism consists of

$A = \prod_i A_i$ a set of actions

$x^* : A \rightarrow \mathbf{X}$ an allocation function

$p_i : A \rightarrow \mathcal{R}$ n payment functions

with a quasi-linear utility function

$$u_i(\mathbf{x}, p_i, t_i) = v_i(\mathbf{x}; t_i) - p_i$$

This induces a Bayesian game where each agent adopts a strategy $\pi_i : T_i \rightarrow A_i$ mapping each possible type to an action.

Mechanism Design

Definition

A mechanism consists of

$A = \prod_i A_i$ a set of actions

$x^* : A \rightarrow \mathbf{X}$ an allocation function

$p_i : A \rightarrow \mathcal{R}$ n payment functions

with a quasi-linear utility function

$$u_i(\mathbf{x}, p_i, t_i) = v_i(\mathbf{x}; t_i) - p_i$$

This induces a Bayesian game where each agent adopts a strategy $\pi_i : T_i \rightarrow A_i$ mapping each possible type to an action.

Mechanism Design

Definition

A mechanism consists of

$A = \prod_i A_i$ a set of actions

$x^* : A \rightarrow \mathbf{X}$ an allocation function

$p_i : A \rightarrow \mathcal{R}$ n payment functions

with a quasi-linear utility function

$$u_i(\mathbf{x}, p_i, t_i) = v_i(\mathbf{x}; t_i) - p_i$$

This induces a Bayesian game where each agent adopts a strategy $\pi_i : T_i \rightarrow A_i$ mapping each possible type to an action.

Mechanism Design

Definition

A mechanism consists of

$A = \prod_i A_i$ a set of actions

$x^* : A \rightarrow \mathbf{X}$ an allocation function

$p_i : A \rightarrow \mathcal{R}$ n payment functions

with a quasi-linear utility function

$$u_i(\mathbf{x}, p_i, t_i) = v_i(\mathbf{x}; t_i) - p_i$$

This induces a Bayesian game where each agent adopts a strategy $\pi_i : T_i \rightarrow A_i$ mapping each possible type to an action.

Mechanism Design

Definition

A mechanism consists of

$A = \prod_i A_i$ a set of actions

$x^* : A \rightarrow \mathbf{X}$ an allocation function

$p_i : A \rightarrow \mathcal{R}$ n payment functions

with a quasi-linear utility function

$$u_i(\mathbf{x}, p_i, t_i) = v_i(\mathbf{x}; t_i) - p_i$$

This induces a Bayesian game where each agent adopts a strategy $\pi_i : T_i \rightarrow A_i$ mapping each possible type to an action.

Mechanism Design: efficiency and equilibria

This paper focusses on mechanisms that implement social welfare maximization or efficient allocation.

$$x^*(\pi(t)) = \arg\max SW(\mathbf{x}; t)$$

where π_i are the strategies induced by the mechanism under equilibrium.
Other assumptions

- incentive compatible
- revelation principle allows them to assume $A_i = T_i$
- ex-post individually rational no agent is better off not playing even if they know everyone else's types
- ex-post equilibrium π_i is optimal for i even when they know everyone elses types

Partial Types and Iterative Querying

Notation

Q_i	set of queries that m can pose to agent i
$R_i(q_i)$	set of possible responses to $q_i \in Q_i$
$\theta_i \subseteq T_i$	the partial type for agent i. θ is the partial type vector for all agents. Since each response r tells us about agent i's type we also say $\theta_i(r)$ for $r \in R_i(q_i)$

Histories

A nonterminal history is a finite sequence of query/response pairs. A terminal history is a nonterminal history followed by an outcome $\mathbf{x} \in \mathbf{X}$.

Notation

$\mathcal{H} = \mathcal{H}_t \cup \mathcal{H}_n$

h_i	restrict to queries and responses involving agent i
$h^{\leq k}$	first k steps of history
h^k	k^{th} step in history
$a(h^k)$	the "action" at step k, query or outcome

Histories

A nonterminal history is a finite sequence of query/response pairs. A terminal history is a nonterminal history followed by an outcome $\mathbf{x} \in \mathbf{X}$.

Notation

$\mathcal{H} = \mathcal{H}_t \cup \mathcal{H}_n$

h_i restrict to queries and responses involving agent i

$h^{\leq k}$ first k steps of history

h^k k^{th} step in history

$a(h^k)$ the "action" at step k, query or outcome

Histories

A nonterminal history is a finite sequence of query/response pairs. A terminal history is a nonterminal history followed by an outcome $\mathbf{x} \in \mathbf{X}$.

Notation

$\mathcal{H} = \mathcal{H}_t \cup \mathcal{H}_n$

h_i	restrict to queries and responses involving agent i
$h^{\leq k}$	first k steps of history
h^k	k^{th} step in history
$a(h^k)$	the "action" at step k, query or outcome

Histories

A nonterminal history is a finite sequence of query/response pairs. A terminal history is a nonterminal history followed by an outcome $\mathbf{x} \in \mathbf{X}$.

Notation

$\mathcal{H} = \mathcal{H}_t \cup \mathcal{H}_n$

h_i	restrict to queries and responses involving agent i
$h^{\leq k}$	first k steps of history
h^k	k^{th} step in history
$a(h^k)$	the "action" at step k, query or outcome

Histories

A nonterminal history is a finite sequence of query/response pairs. A terminal history is a nonterminal history followed by an outcome $\mathbf{x} \in \mathbf{X}$.

Notation

$\mathcal{H} = \mathcal{H}_t \cup \mathcal{H}_n$

h_i restrict to queries and responses involving agent i

$h^{\leq k}$ first k steps of history

h^k k^{th} step in history

$a(h^k)$ the "action" at step k, query or outcome

Histories

A nonterminal history is a finite sequence of query/response pairs. A terminal history is a nonterminal history followed by an outcome $\mathbf{x} \in \mathbf{X}$.

Notation

$\mathcal{H} = \mathcal{H}_t \cup \mathcal{H}_n$

h_i	restrict to queries and responses involving agent i
$h^{\leq k}$	first k steps of history
h^k	k^{th} step in history
$a(h^k)$	the "action" at step k, query or outcome

Incremental Mechanism

Definition

An incremental mechanism is a pair $M =< m, (p_i)_{i \leq n} >$

$m : \mathcal{H}_n \rightarrow Q \cup \mathbf{X}$ the entire history to this point determines the next action, a query or an allocation for each agent

$p_i : \mathcal{H}_t \rightarrow \mathcal{R}$ at the end the entire history maps to a payment for each agent

Definition

The revealed partial type of agent i is the cummulative restriction revealed by all of i's responses

$$\theta_i(h_i) = \cap_{j \leq k} \theta_i(r^j)$$

Incremental Mechanism

Definition

An incremental mechanism is a pair $M =< m, (p_i)_{i \leq n} >$

$m : \mathcal{H}_n \rightarrow Q \cup \mathbf{X}$ the entire history to this point determines the next action, a query or an allocation for each agent

$p_i : \mathcal{H}_t \rightarrow \mathcal{R}$ at the end the entire history maps to a payment for each agent

Definition

The revealed partial type of agent i is the cummulative restriction revealed by all of i's responses

$$\theta_i(h_i) = \cap_{j \leq k} \theta_i(r^j)$$

Incremental Mechanism

Definition

An incremental mechanism is a pair $M = < m, (p_i)_{i \leq n} >$

$m : \mathcal{H}_n \rightarrow Q \cup \mathbf{X}$ the entire history to this point determines the next action, a query or an allocation for each agent

$p_i : \mathcal{H}_t \rightarrow \mathcal{R}$ at the end the entire history maps to a payment for each agent

Definition

The revealed partial type of agent i is the cummulative restriction revealed by all of i's responses

$$\theta_i(h_i) = \cap_{j \leq k} \theta_i(r^j)$$

Incremental Mechanism

Definition

An incremental mechanism is a pair $M =< m, (p_i)_{i \leq n} >$

$m : \mathcal{H}_n \rightarrow Q \cup \mathbf{X}$ the entire history to this point determines the next action, a query or an allocation for each agent

$p_i : \mathcal{H}_t \rightarrow \mathcal{R}$ at the end the entire history maps to a payment for each agent

Definition

The revealed partial type of agent i is the cummulative restriction revealed by all of i's responses

$$\theta_i(h_i) = \cap_{j \leq k} \theta_i(r^j)$$

Strategies

- An agent's strategy maps the agent's history, current query and their type into a response

$$\pi_i(h_i, q_i; t_i) \in R_i(q_i)$$

- Given a mapping, m, as well as all strategies, π, and types, t one specific history is induced $h(m, \pi, t)$
- A strategy is truthful iff for all t_i, q_i and h_i

$$t_i \in \theta_i(\pi_i(h_i, q_i; t_i))$$

Definition

A direct incremental mechanism relies only on revealed partial types rather than histories.

$$\left.\begin{array}{l} m(h) = m(h') \\ p_i(h) = p_i(h') \end{array}\right. \quad \text{if } \theta_i(h) = \theta_i(h') \text{ for all } i.$$

Denoted $m(\theta)$ and $p_i(\theta)$

Strategies

- An agent's strategy maps the agent's history, current query and their type into a response

$$\pi_i(h_i, q_i; t_i) \in R_i(q_i)$$

- Given a mapping, m, as well as all strategies, π, and types, t one specific history is induced $h(m, \pi, t)$
- A strategy is truthful iff for all t_i, q_i and h_i

$$t_i \in \theta_i(\pi_i(h_i, q_i; t_i))$$

Definition

A direct incremental mechanism relies only on revealed partial types rather than histories.

$$\begin{aligned} m(h) &= m(h') \\ p_i(h) &= p_i(h') \end{aligned} \qquad \text{if } \theta_i(h) = \theta_i(h') \text{ for all } i.$$

Denoted $m(\theta)$ and $p_i(\theta)$

Partial Revelation Mechanism

Definition

In a partial revelation mechanism there exists some terminal history, h, and some agent, i, s.t. $\theta_i(h_i)$ contains more than one type.

Once the history induced by π is terminal the utility can be expressed as

$$u_i(\pi_i, \pi_{-i}, t_i) = v_i(x^*(\theta(h)); t_i) - p_i(\theta(h))$$

Properties of the Mechanism

Definition

A direct mechanism $M =< m, p >$ is δ-allocation certain iff for all realizable terminal histories h, $\mathbf{x}^*(\theta(h))$

$$\forall t \in \theta(h), \forall \mathbf{x} \in \mathbf{X}, SW(\mathbf{x}^*(\theta(h)); t) \leq SW(\mathbf{x}; t) - \delta$$

Definition

A mechanism M is δ-efficient iff

- it is δ-allocation certain
- it is terminating

Regret Minimization

Minimizing *MMR* is hard, some factored forms help.

- Generalized additive independence (GAI) allows utility to be expresses as linear constraints.
- Optimization procedure allows the resulting linear, mixed-integer program to be solved by enumerating a small number of constraints.

A Regret Minimization Implementation

Current Solution Strategy (CSS) works by

- Given $\theta, \mathbf{x}$ and $\hat{\mathbf{x}}$.
- Each allocation is tied to some GAI factors
- Pick factor that has loosest bound among all the allocations
- Ask user queries this tighten bound
- For regret-based: After query compute $MMR(\theta)$ if $\leq \delta$ then terminate with $\mathbf{x}^*$, otherwise use $\mathbf{x}^*$ and $\hat{\mathbf{x}}$ for next round.

Regret can be made arbitrarily small, but not necessarily brought to zero with linear constraints.

Keeping them Honest

Definition

A partial VCG payment scheme is defined as $M =< m, (p_i^\top)_{i \leq n} >$ where

- m is δ- efficient
- $p_i^\top(\theta) = \max_{t_{-i} \in \theta_{-i}} p_i^v(\mathbf{x}^*(\theta), t_{-i})$

where p_i^v is the VCG payment scheme:

$$p_i^v(\mathbf{x}, t_{-i}) = \max_{\mathbf{x}_{-i}} SW_{-i}(\mathbf{x}_{-i}; t_{-i}) - SW_{-i}(\mathbf{x}; t_{-i})$$

Keeping them Honest

Definition

A partial VCG payment scheme is defined as $M =< m, (p_i^\top)_{i \leq n} >$ where

- m is δ- efficient
- $p_i^\top(\theta) = \max_{t_{-i} \in \theta_{-i}} p_i^v(\mathbf{x}^*(\theta), t_{-i})$

where p_i^v is the VCG payment scheme:

$$p_i^v(\mathbf{x}, t_{-i}) = \max_{\mathbf{x}_{-i}} SW_{-i}(\mathbf{x}_{-i}; t_{-i}) - SW_{-i}(\mathbf{x}; t_{-i})$$

Keeping them Honest

Definition

A partial VCG payment scheme is defined as $M =< m, (p_i^\top)_{i \leq n} >$ where

- m is δ- efficient
- $p_i^\top(\theta) = \max_{t_{-i} \in \theta_{-i}} p_i^v(\mathbf{x}^*(\theta), t_{-i})$

where p_i^v is the VCG payment scheme:

$$p_i^v(\mathbf{x}, t_{-i}) = \max_{\mathbf{x}_{-i}} SW_{-i}(\mathbf{x}_{-i}; t_{-i}) - SW_{-i}(\mathbf{x}; t_{-i})$$

Payment Range (SW-CSS)

Theorem

Let M have a δ-efficient allocation function and use partial VCG payments. Then M is a δ-efficient, δ-ex post individually rational, $(\delta + \epsilon(\mathbf{x}^*(\theta)))$-ex post incentive compatible mechanism, where $\epsilon(\mathbf{x}) = \max_i \epsilon_i(\mathbf{x})$, and:

$$\epsilon_i(\mathbf{x}) = \max_{t'_{-i} \in \theta_{-i}} p_i^V(\mathbf{x}, t'_{-i}) - \min_{t_{-i}} p_i^V(\mathbf{x}, t_{-i})$$

- SW is within δ of optimal
- Lying about your type can gain you at most $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$
- Cannot gain more than δ ex post by not participating

Payment Range (SW-CSS)

Theorem

Let M have a δ-efficient allocation function and use partial VCG payments. Then M is a δ-efficient, δ-ex post individually rational, $(\delta + \epsilon(\mathbf{x}^*(\theta)))$-ex post incentive compatible mechanism, where $\epsilon(\mathbf{x}) = \max_i \epsilon_i(\mathbf{x})$, and:

$$\epsilon_i(\mathbf{x}) = \max_{t'_{-i} \in \theta_{-i}} p_i^V(\mathbf{x}, t'_{-i}) - \min_{t_{-i}} p_i^V(\mathbf{x}, t_{-i})$$

- SW is within δ of optimal
- Lying about your type can gain you at most $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$
- Cannot gain more than δ ex post by not participating

Payment Range (SW-CSS)

Theorem

Let M have a δ-efficient allocation function and use partial VCG payments. Then M is a δ-efficient, δ-ex post individually rational, $(\delta + \epsilon(\mathbf{x}^*(\theta)))$-ex post incentive compatible mechanism, where $\epsilon(\mathbf{x}) = \max_i \epsilon_i(\mathbf{x})$, and:

$$\epsilon_i(\mathbf{x}) = \max_{t'_{-i} \in \theta_{-i}} p_i^v(\mathbf{x}, t'_{-i}) - \min_{t_{-i}} p_i^v(\mathbf{x}, t_{-i})$$

- SW is within δ of optimal
- Lying about your type can gain you at most $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$
- Cannot gain more than δ ex post by not participating

Payment Range (SW-CSS)

Theorem

Let M have a δ-efficient allocation function and use partial VCG payments. Then M is a δ-efficient, δ-ex post individually rational, $(\delta + \epsilon(\mathbf{x}^*(\theta)))$-ex post incentive compatible mechanism, where $\epsilon(\mathbf{x}) = \max_i \epsilon_i(\mathbf{x})$, and:

$$\epsilon_i(\mathbf{x}) = \max_{t'_{-i} \in \theta_{-i}} p_i^V(\mathbf{x}, t'_{-i}) - \min_{t_{-i}} p_i^V(\mathbf{x}, t_{-i})$$

- SW is within δ of optimal
- Lying about your type can gain you at most $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$
- Cannot gain more than δ ex post by not participating

Payment Elicitation (P-CSS)

Problem: If $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$ is too loose then it may not induce truthfulness.

Solution: Second phase of elicitation to determine payments. Goal is to reduce ϵ to a predetermined, type-independent value.

Define: $t^\top_{-i}$ and $t^\perp_{-i}$ types define the max and min payments for i in $\mathbf{x}^*$. $\mathbf{x}^\top_{-i}$ and $\mathbf{x}^\perp_{-i}$ are the optimal allocations under those types.

$$\begin{aligned}\epsilon_i(\mathbf{x}^*) = {} & SW_{-i}(\mathbf{x}^\top_{-i}; t^\top_{-i}) - SW_{-i}(\mathbf{x}^*; t^\top_{-i}) \\ & - SW_{-i}(\mathbf{x}^\perp_{-i}; t^\perp_{-i}) + SW_{-i}(\mathbf{x}^*_{-i}; t^\perp_{-i})\end{aligned}$$

Payment Elicitation (P-CSS)

Problem: If $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$ is too loose then it may not induce truthfulness.

Solution: Second phase of elicitation to determine payments. Goal is to reduce ϵ to a predetermined, type-independent value.

Define: $t^{\top}_{-i}$ and $t^{\perp}_{-i}$ types define the max and min payments for i in $\mathbf{x}^*$. $\mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$ are the optimal allocations under those types.

$$\epsilon_i(\mathbf{x}^*) = SW_{-i}(\mathbf{x}^{\top}_{-i}; t^{\top}_{-i}) - SW_{-i}(\mathbf{x}^*; t^{\top}_{-i}) - SW_{-i}(\mathbf{x}^{\perp}_{-i}; t^{\perp}_{-i}) + SW_{-i}(\mathbf{x}^*_{-i}; t^{\perp}_{-i})$$

Payment Elicitation (P-CSS)

Problem: If $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$ is too loose then it may not induce truthfulness.

Solution: Second phase of elicitation to determine payments. Goal is to reduce ϵ to a predetermined, type-independent value.

Define: $t^{\top}_{-i}$ and $t^{\perp}_{-i}$ types define the max and min payments for i in $\mathbf{x}^*$. $\mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$ are the optimal allocations under those types.

$$\epsilon_i(\mathbf{x}^*) = SW_{-i}(\mathbf{x}^{\top}_{-i}; t^{\top}_{-i}) - SW_{-i}(\mathbf{x}^*; t^{\top}_{-i}) - SW_{-i}(\mathbf{x}^{\perp}_{-i}; t^{\perp}_{-i}) + SW_{-i}(\mathbf{x}^*_{-i}; t^{\perp}_{-i})$$

Payment Elicitation (P-CSS)

Problem: If $\gamma = (\delta + \epsilon(\mathbf{x}^*(\theta)))$ is too loose then it may not induce truthfulness.

Solution: Second phase of elicitation to determine payments. Goal is to reduce ϵ to a predetermined, type-independent value.

Define: $t^{\top}_{-i}$ and $t^{\perp}_{-i}$ types define the max and min payments for i in $\mathbf{x}^*$. $\mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$ are the optimal allocations under those types.

$$\begin{aligned}\epsilon_i(\mathbf{x}^*) = {} & SW_{-i}(\mathbf{x}^{\top}_{-i}; t^{\top}_{-i}) - SW_{-i}(\mathbf{x}^*; t^{\top}_{-i}) \\ & - SW_{-i}(\mathbf{x}^{\perp}_{-i}; t^{\perp}_{-i}) + SW_{-i}(\mathbf{x}^*_{-i}; t^{\perp}_{-i})\end{aligned}$$

Another way to think about it

- Design so far
  - one round of elicitation to reduce allocation uncertainty and choose $\mathbf{x}^*$
  - another round to reduce manipulability and payment uncertainty in $\mathbf{x}^*$
- But the true type of the agent is unique and with $\mathbf{x}^*$ determines both efficiency and payment uncertainty. They are not independent.
- Objective is not to reduce uncertainty but to reduce manipulability.

Manipulability

The ammount an agent can manipulate the mechanism by lying is

$$\alpha_i(\mathbf{x}^*, t) = \max_{\hat{\mathbf{x}}}[v_i(\hat{\mathbf{x}}; t_i) - p_i^V(\hat{\mathbf{x}}; t_{-i})] - v_i(\mathbf{x}^*; t_i) + p_i^\top(\mathbf{x}^*; \theta_{-i})$$

The worst-case manipulability of the mechanism is $\alpha = \max_t \max_i \{\alpha_i(\mathbf{x}^*, t)\}$. If this holds then M is α-manipulable.

Theorem

Let M be an α-manipulable mechanism using partial VCG payments. Then M is α-efficient, α-ex post individually rational, and α-ex post incentive compatible.

Substrategies for Elicitation

We have defined the follow CSSs

SW-CSS maximizaing social welfare

PS-CSS reducing payment uncertainty

Now we define one more

M-CSS reducing manipulability

- When computing $\mathbf{x}^*$ to minimize α we get $\mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.
- M-CSS asks a query for the associated parameter in GAI model with the largest gap
- This performs poorly, reduces uncertainty on payments for unrealized allocations.

Substrategies for Elicitation

We have defined the follow CSSs

- **SW-CSS** maximizaing social welfare
- **PS-CSS** reducing payment uncertainty

Now we define one more

- **M-CSS** reducing manipulability
  - When computing $\mathbf{x}^*$ to minimize α we get $\mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.
  - M-CSS asks a query for the associated parameter in GAI model with the largest gap
  - This performs poorly, reduces uncertainty on payments for unrealized allocations.

Substrategies for Elicitation

We have defined the follow CSSs

- **SW-CSS** maximizaing social welfare
- **PS-CSS** reducing payment uncertainty

Now we define one more

- **M-CSS** reducing manipulability
  - When computing $\mathbf{x}^*$ to minimize α we get $\mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.
  - M-CSS asks a query for the associated parameter in GAI model with the largest gap
  - This performs poorly, reduces uncertainty on payments for unrealized allocations.

Elicitation Strategies

Three strategies using the substrategies

two phase (2P) Standard. Run SW-CSS untill $\delta = 0$ yielding $\mathbf{x}^*$. Then run P-CSS until $\delta + \epsilon$ is small.

α-two phase (α2P) Just like 2P but terminating instead when α is below some bound.

common-hybrid (CH) Let A be the set of GAI parameters for SW-regret allocations $\mathbf{x}$ and $\hat{\mathbf{x}}$. Let B be the set of GAI parameters for the manipulability allocations $\mathbf{x}, \hat{\mathbf{x}}, \mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.

1. If A and B have any common parameters, query those with the largest gap
2. Otherwise choose parameters from SW-CSS and M-CSS. Bias towards SW-CSS early on.

Elicitation Strategies

Three strategies using the substrategies

two phase (2P) Standard. Run SW-CSS untill $\delta = 0$ yielding $\mathbf{x}^*$. Then run P-CSS until $\delta + \epsilon$ is small.

α-two phase (α2P) Just like 2P but terminating instead when α is below some bound.

common-hybrid (CH) Let A be the set of GAI parameters for SW-regret allocations $\mathbf{x}$ and $\hat{\mathbf{x}}$. Let B be the set of GAI parameters for the manipulability allocations $\mathbf{x}, \hat{\mathbf{x}}, \mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.

1. If A and B have any common parameters, query those with the largest gap
2. Otherwise choose parameters from SW-CSS and M-CSS. Bias towards SW-CSS early on.

Elicitation Strategies

Three strategies using the substrategies

two phase (2P) Standard. Run SW-CSS untill $\delta = 0$ yielding $\mathbf{x}^*$. Then run P-CSS until $\delta + \epsilon$ is small.

α-two phase (α2P) Just like 2P but terminating instead when α is below some bound.

common-hybrid (CH) Let A be the set of GAI parameters for SW-regret allocations $\mathbf{x}$ and $\hat{\mathbf{x}}$. Let B be the set of GAI parameters for the manipulability allocations $\mathbf{x}, \hat{\mathbf{x}}, \mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.

1. If A and B have any common parameters, query those with the largest gap
2. Otherwise choose parameters from SW-CSS and M-CSS. Bias towards SW-CSS early on.

Elicitation Strategies

Three strategies using the substrategies

two phase (2P) Standard. Run SW-CSS untill $\delta = 0$ yielding $\mathbf{x}^*$. Then run P-CSS until $\delta + \epsilon$ is small.

α-two phase (α2P) Just like 2P but terminating instead when α is below some bound.

common-hybrid (CH) Let A be the set of GAI parameters for SW-regret allocations $\mathbf{x}$ and $\hat{\mathbf{x}}$. Let B be the set of GAI parameters for the manipulability allocations $\mathbf{x}, \hat{\mathbf{x}}, \mathbf{x}^{\top}_{-i}$ and $\mathbf{x}^{\perp}_{-i}$.

1. If A and B have any common parameters, query those with the largest gap
2. Otherwise choose parameters from SW-CSS and M-CSS. Bias towards SW-CSS early on.

Results

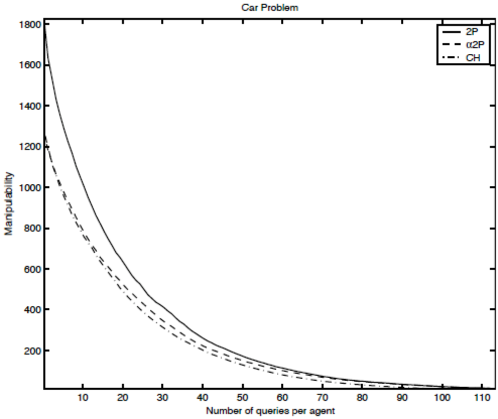

Figure 1: Car Rental Problems. Average of 40 runs. 2 sellers, 1 buyer; 13 factors/agent; 1-4 variables/factor; 2-9 values/variable. 825 parameters total.

Results

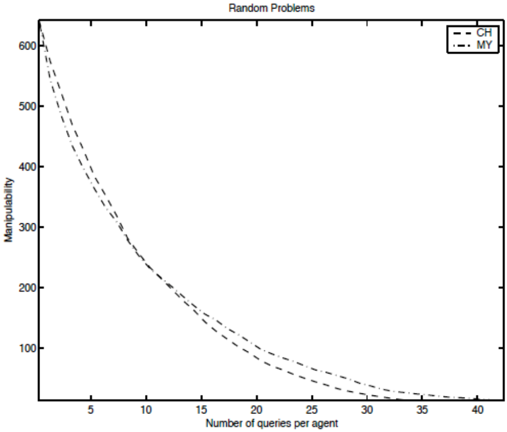

Figure 2: Small Problems. Average of 40 runs. 2 sellers, 1 buyer; 3 factors/agent; 2 variables/factor; 3 values/variable. 81 parameters total.

Results

- α2P and CH have better anytime performance than 2P
- 2P and α2P reach zero manipulability in 110 queries. CH does it in 95.
- Only 8% of the utility parameters were queried by CH.
- On average 92% of the uncertainty remains while other methods that halve uncertainty get down to 64% uncertainty but remain far from reaching zero-manipulability

Conclusion

- They showed how to use min-max regret to make allocations with type uncertainty
- They introduced regret-based, incremental, partial revelation mechanisms
- They argued for reducing manipulability rather than type uncertainty as a more efficient approach
- If gain from manipulation is low and cost is high the result is practical, exact incentive compatibility even though formally it is only approximately incentive compatible.

Bibliography

Darius Braziunas and Craig Boutilier.
Local utility elicitation in gai models.
In *UAI-05*, pages 42–49, 2005.

Nathanael Hyafil and Craig Boutilier.
Regret-based incremental partial revelation mechanisms.
In *AAI-06*, 2006.

Questions?