ICE: Iterative Combinatorial Exchanges

Benjamin Lubin

In Collaboration with
David Parkes and Adam Juda

Early work: Giro Cavallo, Jeff Shneidman, Hassan Sultan, CS286r Spring 2004

Overview

- Introduction
- ICE
  - Bidding Language
  - Winner Determination
  - Payments
- Iteration
  - Activity Rules
  - Pricing
- experimentation
  - Implementation
  - Instances
  - Results
- Conclusion

Motivating Domains

- Landing Slots (FAA)
  - "Sell 8am slot and buy 4pm slot"
  - "Swap 2 LaGuardia slots for 3 at Newark"
  - Note: Ground assets also important
- Bandwidth (FCC)
  - "Buy one band, but only if I can get all the licenses for a complete region"
- Computational Resources (PlanetLab)
  - "Sell use of 32 nodes on Thursday and buy use of 24 nodes on Friday."

Combinatorial Auctions

- One Seller, many buyers (or reverse)
- Expressive/Concise bidding languages
  - Non-linear valuations on bundles
  - XOR, OR, OR*, L_{GB}, etc
- Winner determination
  - NP-hard (maximal weighted packing), but polynomial for subclasses
  - Branch-and-bound, branch-and-cut obtain guarantees on solution quality.
  - Approximation: LP-based, local search etc.
- Payments
  - First Price, VCG, Core

Combinatorial Exchanges

- Extension of Combinatorial Auctions
  - Multiple competitive buyers, sellers (or mixed)
- Expressive bids:
  - (sell [A,B] for $8) xor (sell [C,D] for $20)
  - (buy A) and (sell B) for $5 [swap]
- Winner Determination is a combinatorial optimization problem
  - capture logical constraints in bids
  - maximize "gains from trade"
- Payments: at final allocation what do you pay?
  - VCG fails Budget Balance \rightarrow Use Threshold Payments
  - Not strategyproof but mitigates incentives to manipulate
  - Core Constraints?

Exchange Example 1

- sell A
- buy AB
- sell B
- buy A
- swap A for B
Overview

- Introduction ✓
- ICE
  - Bidding Language
  - Winner Determination
  - Payments
- Iteration
  - Activity Rules
  - Pricing
- Experimentation
  - Implementation
  - Instances
  - Results
- Conclusion

Related Work

- Concise Combinatorial Languages
  - OR* (Nisan '00)
  - LGB (Boutilier & Hoos '01)
- Iterative Combinatorial Auctions
  - Linear (Gul & Stacchetti '00, Hoffman '01, Kwasnica et al. '05)
  - Non-Linear (Parkes & Ungar '00)
- Clock Proxy (Ausubel & Milgrom '04)
Exchange Properties

- First incremental and fully expressive two-sided combinatorial exchange.
- "Hybrid" Design
  - Incremental direct revelation of upper and lower bounds on trade values via expressive language.
  - "Last and Final" stage where the exchange clears and (Threshold) payments are determined.
  - Shares stylistic features with other "hybrid" designs such as clock-proxy for CAs (Ausubel et al.)
- Theoretical interest: efficiency results with linear prices used for preference elicitation

ICE Control Flow

Tree-Based Bidding Language

- Defines change in value for a trade; entirely symmetric for buyers and sellers
  - e.g., "sell AB, value -$100"; "buy A, value +$20"
  - bids: claim on increase in value from receiving an item
  - asks: claim on decrease in value from giving-up an item
  - mixed buy/sell in TBBL can have + or – values
- Generalizes XOR, OR, XOR/OR (Sandholm’99, Nisan’00).
- Conciseness incomparable with OR* (Fujishima et al’99, Nisan’00), LOG (Boutilier & Hoos’02), although both captured with simple extensions (see Cavallo et al.’05)

Example 1: “and”

Example 2: “xor”

Example 3: “xor of and”
Example 4: “choose”

- IC\([x, y]\): accept an allocation in which at least \(x\) and at most \(y\) of children are "satisfied"
  - IC\([all, all]\) \(\rightarrow\) AND
  - IC\([1, all]\) \(\rightarrow\) OR
  - IC\([1, 1]\) \(\rightarrow\) XOR

Choose 2 or 3

$200 +$180 +$150 +$220

Example 5: “swap”

Swap [2, 2] - $50

Example 6: “contingent sale”

And [2, 2] - $200

How to Solve Winner Determination?

- Goods: \(\{1, \ldots, m\}\)
- Agents: \(\{1, \ldots, n\}\)
- Trades: \(\lambda \in \mathbb{Z}^{m \times n}\)
- Initial allocation: \(x^0 \in \mathbb{Z}^{m \times n}\)
- Final allocation:
  - (change in) value: \(v_i(\lambda_i)\)
  - Winner determination:

\[
\begin{align*}
\max \sum_i v_i(\lambda_i) \\
\text{s.t. } \sum_j \lambda_{ij} x_{ij} \geq 0, \forall i, v_j \\
\sum_i \lambda_{ij} = 0, v_j \\
\lambda_{ij} \in \mathbb{Z}, \forall i, v_j
\end{align*}
\]

A Better Formulation

Agent problem. Given \(\lambda\)

\[
\begin{align*}
\max \sum_i v_i(\lambda_i) \\
\text{s.t. } \sum_j \lambda_{ij} x_{ij} \geq 0, \forall i, v_j \\
\sum_i \lambda_{ij} = 0, v_j \\
\lambda_{ij} \in \mathbb{Z}, \forall i, v_j
\end{align*}
\]

Denote this VAL(\(\lambda_i\))

\# vars = |T| 
\# constraints = m + |T|
Payments Redux

Formulate this problem as one of *dividing surplus*, s.t. each agent's payment is value $v_i(\lambda_i) - \Delta_i$ and $\sum_i \Delta_i = V^*$.

Threshold Payments

Example

Surplus = 0

Why Iterative?

• Agents find it difficult to determine their preferences
  - Want to allow approximate information about the complete valuation function
• Iteration allows for price feedback to focus agents on the right part of their value space

From CE to ICE

• A TBBL bid is now annotated with lower and upper bounds on value
• Key idea: clear based on "optimistic" values in early rounds, ... "pessimistic values" in later rounds
  - provides early price discovery
• Bidders tighten bounds across rounds
• Linear prices drive activity, elicitation
TBBL Bounds Example

Two hidden nodes with partial value information defined on their bid tree. One can already prove that pruned trade in the folder 1 to all A and buy C.

MRPAR Activity Rule

- Show one trade is weakly better than all others
- And show that this trade is either the provisional trade or strictly better then it
- Exchange can verify with 3 MIPs

RPAR 1

- $\pi_1(+B) = 30 - 10 = 20$, $\pi_2(+A) = 25 - 10 = 15$
- Enough information

- ($10, 25$)
- $p(A) = 10$
- $p(B) = 10$

RPAR 2

- $\pi_1(+B) = 30 - 5 = 25 < \pi_2(+A) = 110 - 5 = 105$
- Not enough information ??

- ($10, 100$)
- $p(A) = 5$
- $p(B) = 5$

RPAR 3

- $\pi_1(+B) = 20 + v - 5$
- $\pi_2(+A) = 10 + v - 5$
- For all $v$, $\pi_1(+B) > \pi_2(+A)$
- $\pi_1(+B) > \pi_2(+A)$

- ($10, 100$)
- $p(A) = 5$
- $p(B) = 5$

v-DIAR Activity Rule

- Reduce the linear pricing error to within $\varepsilon$, or show that you can’t
- Exchange can verify with 2 MIPs

- $\text{Profit}$
- $\text{DIAR}$
MRPAR + DIAR Activity

Rule Properties

- Guaranteed progress in a given round
- Can lower bound \( \text{EFF}(\lambda) \)
  \[
  \text{EFF}(\lambda) = \sum_i v_i(\lambda_i^*) - \frac{v(\lambda)}{\pi(\lambda)} \geq \Delta^* \]

  - via linear prices (when sufficiently accurate)
  - otherwise directly via bounds on TBBL trees
- Thus despite linear prices:
  - **Theorem.** For straightforward bidders MRPAR and \( \varepsilon \)-DIAR cause the exchange to terminate with a trade that is within a target efficiency error \( \Delta^* \) as \( \varepsilon \to 0 \)

Bounding Efficiency

- 'maximal improvement' valuation enables us to bound efficiency

Pricing

- Linear prices minimize distance:
  - To competitive equilibrium (ACC)
  - To provisional final payments (FAIR)
  - Between items (BAL)

  ACC: AB is between $12 and $16
  FAIR: AB=$14
  BAL: A=$7, B=$7

Computing Prices

- Lexicographic within each stage
  - Most expensive step
  - Constraint Generation
  - Heuristics to speed search

Overview

- Introduction ✔
- ICE ✔
  - Bidding Language
  - Winner Determination
- Payments
- Iteration ✔
  - Activity Rules ✔
  - Pricing ✔
- Experimentation
  - Implementation
  - Instances
- Results
- Conclusion

Constraint Generation

- Accuracy for example:
  \[
  \delta_{\text{acc}} = \min \sum \delta_{\text{acc}}^i \delta_{\text{acc}}^j
  \]

  \[
  \begin{align*}
  \text{s.t.} & \quad \sum_j \sigma_j p_j^i \leq v(\lambda^i) - \sum_j \sigma_j p_j^i + \delta_{\text{acc}}, \forall i, \forall j \\
  \text{and} & \quad \gamma_{\text{acc}} \geq 0, \forall i, \forall j \in G_i
  \end{align*}
  \]

  WD: \( \max_{i, j} \sum \beta \cdot v(\beta) \cdot \text{sat}(\beta) \)
  \[\text{s.t.} \ (1), (2), (3), (4) \]

  RWD: (for each agent)
  \[\max_i \sum_j \varepsilon_j \cdot v(\beta) \cdot \text{sat}(\beta) - \sum_j \gamma_j \cdot q_j \cdot \text{sat}(\beta) \]

  \[\text{s.t.} \ (1), (2), (3), \text{and} \ (4) \]

  Check:
  \[v(\lambda^i) - p(\lambda^i) \leq v(\lambda^i) - p(\lambda^i) + \delta_{\text{acc}}\]
• Thousands of distinct but related MIPs
  – Massive multi-threading/parallelization
  – Modular and hierarchical MIP "code generator"
  – Concise & parallel CPLEX/LPSolve wrapper

• Numerical precision a big practical issue

---

**Generator**

- Create d copies of each good type
- Assign these to the agents
- Recursively build a tree for agents
  - 1st phase: exponential growth
  - 2nd phase: triangle distribution of width
    over depth
  - Internal nodes: Draw Y between 1 and
    [children], X between 1 and Y
  - Leaf nodes: assign buy or sell and then
    choose a good accordingly
  - Draw value for each node from a internal,
    buy, or sell distribution respectively

---

**Agent and Good Scalability**

- Polynomial in agents
- Phase transition behavior in good types

---

**Tree Size Scalability**

- Polynomial in size of tree
Activity Rules

Efficiency Bound

• MRPAR: ‘main rocket’
  DIAR: ‘course correction’
• Efficiency bound effective

Information Revelation

• Bounds retain ‘slack’

Price Quality

• Prices converge quickly
• Low regret (best trade at intermediate prices compared to final prices)

Pricing Error

• Linear prices have low error

Results Summary

• **Fast**: 100 goods in 20 types, 8 bidders each with ~112 TBBL nodes, converges to efficient trade in ~7 rounds
• **Elicitation efficient**: Around 62% “value uncertainty” retained in final bid-trees.
• **Informative**: The best trade for a bidder at intermediate prices within 11% of the profit it would get from its best trade at final prices.
• **Scalable**: 8.5 minutes on 3.2GHz, dual-processor, dual-core, 8GB memory (including agent simulation)

Conclusion

• ICE showcases a “hybrid” design in which linear prices guide elicitation but exchange clears based on expressive bids.
• Linear prices can be generated for expressive languages (e.g. TBBL) and coupled to any (e.g. Threshold) payment rule.
• Threshold payment scheme is “maximally” truthful when participants guaranteed non-negative profit at reported values and the budget is balanced.
• Experiments show that ICE converges quickly, and that it is efficient, informative and scalable
For more information:
- http://www.eecs.harvard.edu/~blubin/ice
- blubin {at} eecs {dot} harvard {dot} edu