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."

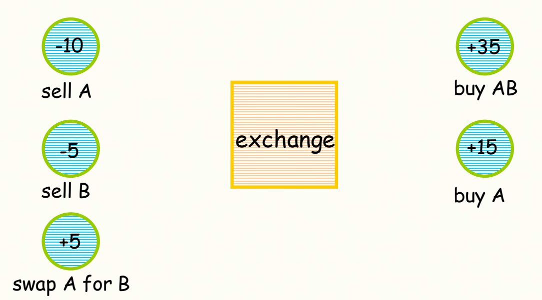
Combinatorial Auctions

- 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

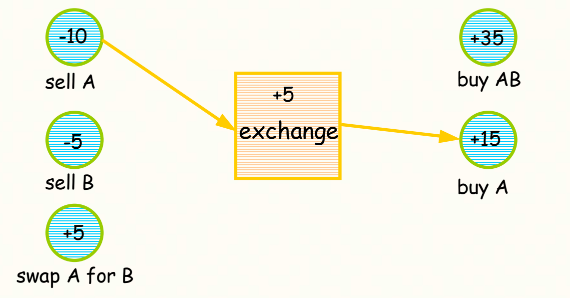
- Extension of Combinatorial Auctions
  - Multiple competitive buyers, sellers (or mixed)
- Expressive bids:
  - (sell [A,B] -$8) xor (sell[C,D] -$20)
  - (buy A) and (sell B) $5 [swap]
- Winner Determination is a combinatorial optimization problem
  - capture logical constraints in bids
  - maximize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

Exchange Example 1



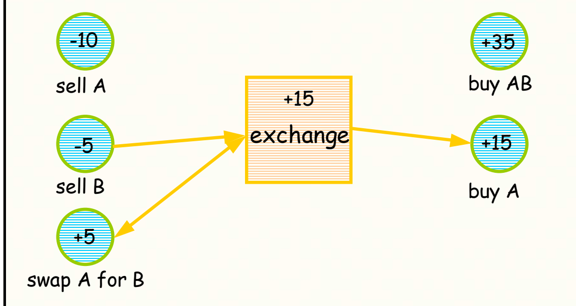
Harvard
School of Engineering
and Applied Sciences

Exchange Example 2



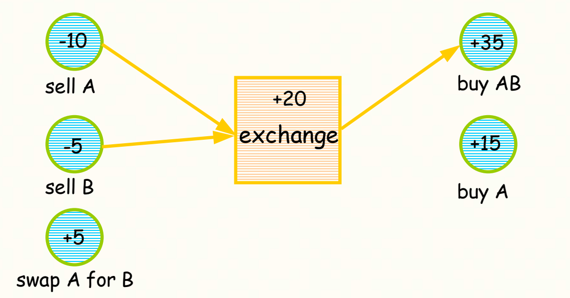
Harvard
School of Engineering
and Applied Sciences

Exchange Example 3



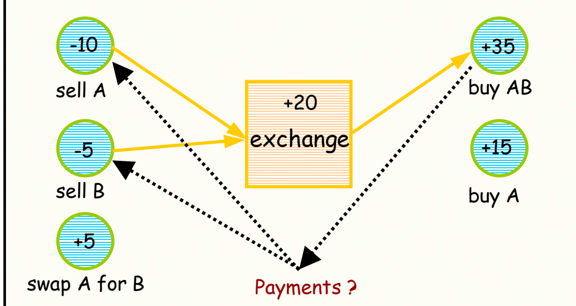
Harvard
School of Engineering
and Applied Sciences

Exchange Example 4



Harvard
School of Engineering
and Applied Sciences

Exchange Example 5



Harvard
School of Engineering
and Applied Sciences

Overview

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

Harvard
School of Engineering
and Applied Sciences

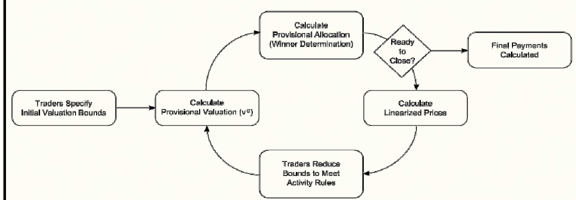
Related Work

- 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

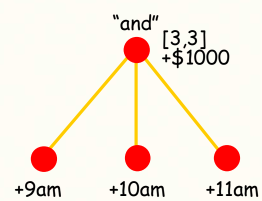
ICE Control Flow



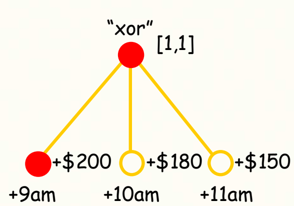
Tree-Based Bidding Language

- 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, Nisan00), L_{GB} (Boutilier & Hoos'02), although both captured with simple extensions (see Cavallo et al.'05)

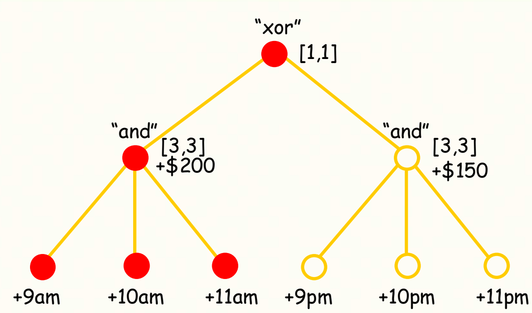
Example 1: "and"



Example 2: "xor"



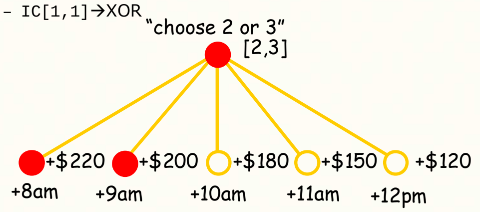
Example 3: "xor of and"



Harvard
School of Engineering
and Applied Sciences

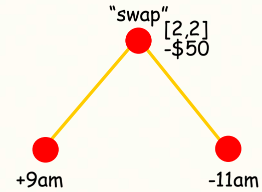
Example 4: "choose"

- `IC[x,y]`: accept an allocation in which *at least* *x* and *at most* *y* of children are "satisfied"
  - `IC[all,all]`→AND
  - `IC[1,all]`→OR
  - `IC[1,1]`→XOR



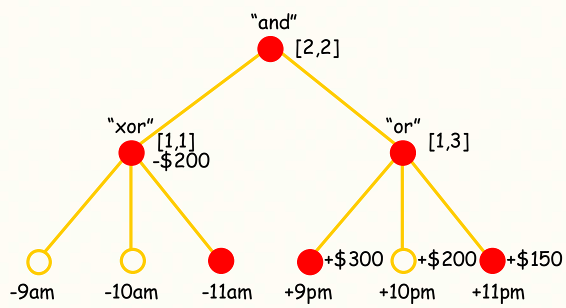
Harvard
School of Engineering
and Applied Sciences

Example 5: "swap"



Harvard
School of Engineering
and Applied Sciences

Example 6: "contingent sale"



Harvard
School of Engineering
and Applied Sciences

How to Solve Winner Determination?

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

- Winner determination:

$$\begin{aligned} \max \quad & \sum_i v_i(\lambda_i) \\ \text{s.t.} \quad & \left.\begin{aligned} &\lambda_{ij}+x^0_{ij} \geq 0, \ \forall i \ \forall j \\ &\sum_i \lambda_{ij} = 0, \ \forall j \end{aligned}\right\} \lambda \in \text{feas}(x^0) \\ & \lambda_{ij} \in Z \end{aligned}$$

Harvard
School of Engineering
and Applied Sciences

Possible formulation

- Construct "flat" representation of each agent's bids
- i.e., given tree T then for all $\lambda_i \in \Lambda_i = \text{Feas}(x^0)_i$, $\text{eval}(T,\lambda_i)$ and consider $v_i(\lambda_1)$ xor $v_i(\lambda_2)$ xor …

$$\begin{aligned} \max_{\{z(\lambda)\}} \quad & \sum_i \sum_{\lambda_i \in \Lambda_i} z_i(\lambda_i) v_i(\lambda_i) \\ \text{s.t.} \quad & \sum_{\lambda_i \in \Lambda_i} z_i(\lambda_i)\lambda_{ij} + x^0_{ij} \geq 0, \ \forall i, \forall j \\ & \sum_i \sum_{\lambda_i \in \Lambda_i} z_i(\lambda_i)\lambda_{ij} = 0, \ \forall j \\ & z_i(\lambda_i) \in \{0,1\}, \ \forall i, \forall \lambda_i \in \Lambda_i \end{aligned}$$

- Solve using branch-cut-and-bound (e.g. CPLEX)
- Problems?

Harvard
School of Engineering
and Applied Sciences

A Better Formulation

Agent problem. Given λ_i

$$\begin{aligned} \max_{\text{sat}_i\{\beta\}} \quad & \sum_{\beta \in T} v_i(\beta)\, \text{sat}_i(\beta) \\ \text{s.t.} \quad & \sum_{\beta \in \text{Leaf}(i)} q_{ij}(\beta)\, \text{sat}_i(\beta) \leq \lambda_{ij} \ \forall j \quad (3) \\ & IC_{x,i}(\beta)\text{sat}_i(\beta) \leq \sum_{\beta' \in \text{child}(\beta)} \text{sat}_i(\beta') \\ & \quad \leq IC_{y,i}(\beta)\text{sat}_i(\beta), \ \forall \beta \notin \text{Leaf}(i) \quad (4) \end{aligned}$$

Denote this $\text{VAL}_i(\lambda_i)$

\# vars = $|T|$

\# constraints = $m+|T|$

☺

- Joint problem. Find $\lambda=(\lambda_1,\ldots,\lambda_n)$

$$\begin{aligned} \max_\lambda \quad & \sum_i \text{VAL}_i(\lambda_i) \\ \text{s.t.} \quad & \lambda_{ij}+x^0_{ij} \leq 0, \ \forall i, \forall j \quad (1) \\ & \sum_i \lambda_{ij} \leq 0, \ \forall j \quad (2) \\ & \lambda_{ij} \in Z, \ \forall i, \forall j \end{aligned}$$

\# vars = m x n

\#constraints = m x n + n

- Roll into a single program

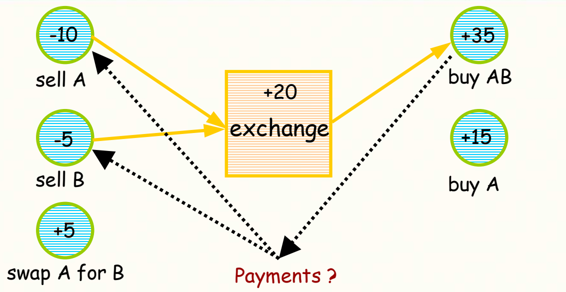
$$\begin{aligned} \max_{\lambda,\text{sat}} \quad & \sum_i \sum_{\beta \in T_i} v_i(\beta)\text{sat}_i(\beta) \\ \text{s.t.} \quad & (1), (2), \{(3)_1,\ldots,(3)_n\}, \{(4)_1,\ldots,(4)_n\} \end{aligned}$$

\# vars = (m x n) + (n x |T|)

\#constraints = m x n + n + n(m+|T|)

☺

Payments Redux

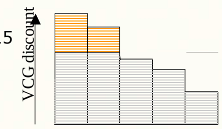


Formulate this problem as one of *dividing surplus*, s.t. each agent's payment is value $v_i(\lambda_i) - \Delta_i$ and $\sum_i \Delta_i = V^*$

Harvard School of Engineering and Applied Sciences

Payments: VCG & Threshold

- VCG Payments:
  - VCG discount: $\Delta_{vcg,i} = V^* - V^{-i}$
  - Agent 1 pays -10–(20-15)=-15
  - Agent 2 pays -5–(20-5)=-20
  - Agent 3 pays 35–(20-15)=30
  - Deficit: 30-20-15 = -5



- Threshold Payments:
  - Payments $v_i(\lambda^*)-\Delta_i$ Choose discounts Δ_i to:

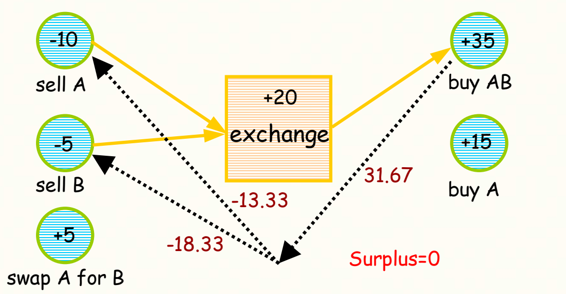
    min $\{\max \Delta_{vcg,i}-\Delta_i\}$

    s.t. $\sum_i \Delta_i <= V^*$ and $\Delta_i <= \Delta_{vcg,i}$
  - $\Delta_1 = 3.33$ $\Delta_2 = 13.33$ $\Delta_3 = 3.33$
  - Agent 1 pays –13.33
  - Agent 2 pays –18.33
  - Agent 3 pays 31.67
  - ex post regret = $\Delta_{vcg,i}-\Delta_i = 1.67$

[PKE '01]

Harvard School of Engineering and Applied Sciences

Threshold Payments Example



Harvard School of Engineering and Applied Sciences

Overview

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

Harvard School of Engineering and Applied Sciences

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

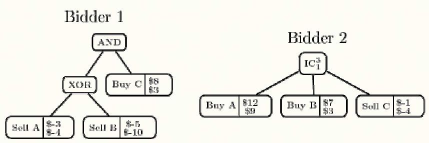
Harvard School of Engineering and Applied Sciences

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

Harvard School of Engineering and Applied Sciences

TBBL Bounds Example

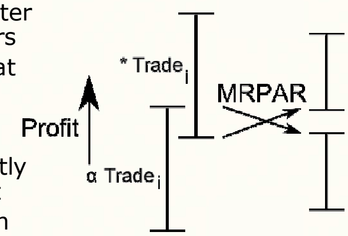


Two bidders, each with partial value information defined on their bid tree. One can already prove that the efficient trade is for bidder 1 to sell A and buy C.

Harvard School of Engineering and Applied Sciences

MRPAR Activity Rule

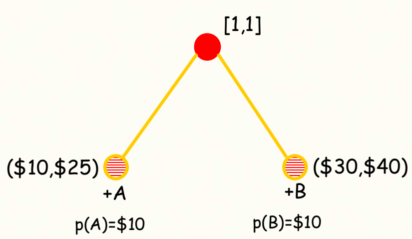
- Show one trade is weakly better then all others
- And show that this trade is either the provisional trade or strictly better then it
- Exchange can verify with 3 MIPs



Harvard School of Engineering and Applied Sciences

RPAR 1

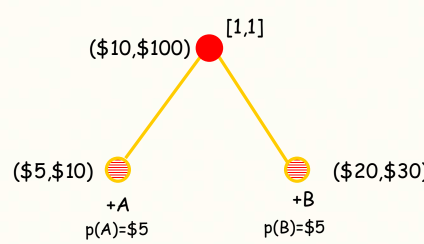
- $\pi_L(+B) = 30 - 10 = 20$, $\pi_U(+A) = 25 - 10 = 15$
- Enough information



Harvard School of Engineering and Applied Sciences

RPAR 2

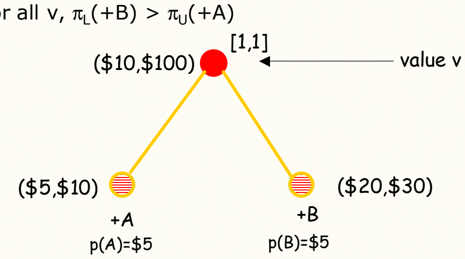
- $\pi_L(+B) = 30 - 5 = 25 < \pi_U(+A) = 110 - 5 = 105$
- Not enough information ??



Harvard School of Engineering and Applied Sciences

RPAR 3

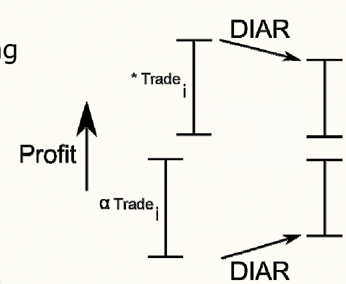
- $\pi_L(+B)=20+v-5$
- $\pi_U(+A)=10+v-5$
- For all v, $\pi_L(+B) > \pi_U(+A)$



Harvard School of Engineering and Applied Sciences

ε-DIAR Activity Rule

- Reduce the linear pricing error to within ε, or show that you can't
- Exchange can verify with 2 MIPs



MRPAR+DIAR Activity Rule Properties

Harvard School of Engineering and Applied Sciences

- Guaranteed progress in a given round
- Can lower bound EFF($\underline{\lambda}$)

$$\mathrm{EFF}(\underline{\lambda}) = \frac{\sum_i v_i(\underline{\lambda}_i)}{\sum_i v_i(\lambda_i^*)} = \frac{v(\underline{\lambda})}{v(\lambda^*)} \geq \Delta^*$$

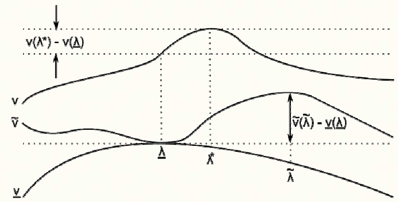
  - via *linear* prices (when sufficiently accurate)
  - otherwise directly via bounds on TBBL trees
- Thus despite linear prices:
  - **Theorem**. For straightforward bidders MRPAR and ε-DIAR cause the exchange to terminate with a trade that is within a target efficiency error Δ^* as $\varepsilon \to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

Bounding Efficiency

Harvard School of Engineering and Applied Sciences

- 'maximal improvement' valuation enables us to bound efficiency

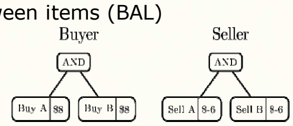
$$\mathrm{EFF}(\underline{\lambda}) = \frac{v(\underline{\lambda})}{v(\lambda^*)} \geq \min_{v' \in T, \lambda' \in \mathcal{F}(x^0)} \left[\frac{v'(\underline{\lambda})}{v'(\lambda')}\right] = \min_{\lambda' \in \mathcal{F}(x^0)} \left[\frac{\tilde{v}(\underline{\lambda})}{\tilde{v}(\lambda')}\right] = \frac{\underline{v}(\underline{\lambda})}{\tilde{v}(\tilde{\lambda})}$$



Pricing

Harvard School of Engineering and Applied Sciences

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

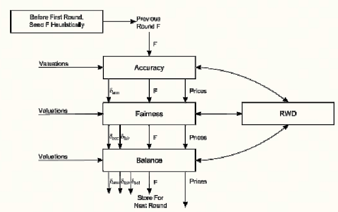


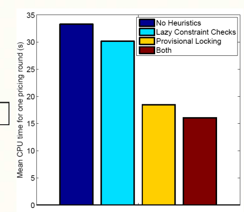
ACC: AB is between $12 and $16
FAIR: AB=$14
BAL: A=$7, B=$7

Computing Prices

Harvard School of Engineering and Applied Sciences

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





Constraint Generation

Harvard School of Engineering and Applied Sciences

- Accuracy for example:

$$\delta_{\mathrm{acc}} = \min_{\pi, \delta_{\mathrm{acc}}} \delta_{\mathrm{acc}}$$

$$\text{s.t.} \quad v_i^{\alpha}(\lambda_i') - \sum_j \pi_j \lambda_{ij}' \leq v_i^{\alpha}(\lambda_i^{\alpha}) - \sum_j \pi_j \lambda_{ij}^{\alpha} + \delta_{\mathrm{acc}}, \ \forall i, \forall \lambda_i' \in \mathbb{F}_i$$

$$\delta_{\mathrm{acc}} \geq 0, \quad \pi_j \geq 0, \forall j \in G.$$

WD:
$\max_{\lambda,sat} \sum_i \sum_{\beta \in T_i} v_i(\beta) sat_i(\beta)$
s.t. (1), (2), $\{(3)_1, \ldots, (3)_n\}$, $\{(4)_1, \ldots, (4)_n\}$

→

RWD: (for each agent)
$\max_{sat} \sum_{\beta \in T} v_i(\beta) sat_i(\beta) - \sum_{\beta \in leaf(T)} \pi_{good(\beta)} q_\beta sat_i(\beta)$
s.t. (1), (2), $\{(3)_1, \ldots, (3)_n\}$, $\{(4)_1, \ldots, (4)_n\}$

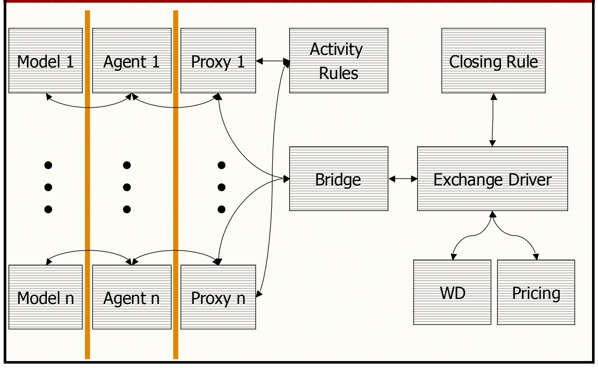
Check:
$v^{\alpha}(\lambda') - p(\lambda') \leq v^{\alpha}(\lambda^{\alpha}) - p(\lambda^{\alpha}) + \delta_{acc}$

Overview

Harvard School of Engineering and Applied Sciences

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

Architecture



Harvard
School of Engineering
and Applied Sciences

Implementation

- 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

Component	Purpose	Lines
Agent	Strategic behavior and information revelation decisions	2001
Model	XML support to load goods and true valuations	1353
Bidding Language	Implements *TBBL*	2497
Exchange Driver & Communication	Controls exchange, and coordinates agent behavior	1322
Activity Rule Engine	MRPAR, DIAR and TPAR	1280
Closing Rule Engine	Checks for termination condition	550
WD Engine	Logic for WD	685
Pricing Engine	Logic for three pricing stages	1317
MIP Builders	Translates from engines into our optimization APIs	782
Pricing Builders	Used by the pricing stages	584
WD Builders	Used by WD, activity & closing rules, pricing	840
Framework	Wire above components together	891
Instrumentation	Gather data for analysis	1751
JOpt	Our Optimization API wrapping CPLEX	2178
Instance Generator	Random Problem Generator	497

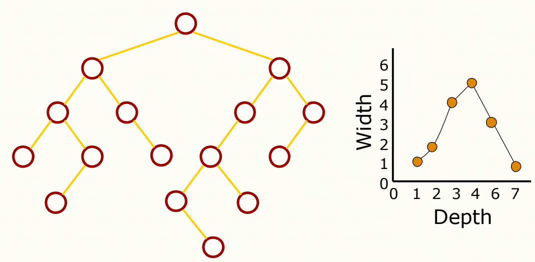
Harvard
School of Engineering
and Applied Sciences

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

Harvard
School of Engineering
and Applied Sciences

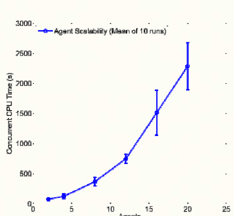
Generator Phase Example

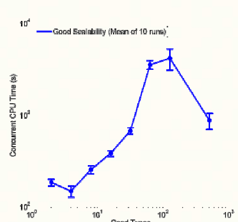


Harvard
School of Engineering
and Applied Sciences

Agent and Good Scalability

- Polynomial in agents
- Phase transition behavior in good types

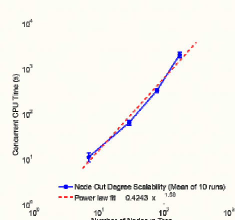


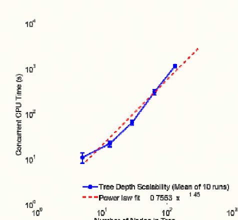


Harvard
School of Engineering
and Applied Sciences

Tree Size Scalability

- Polynomial in size of tree

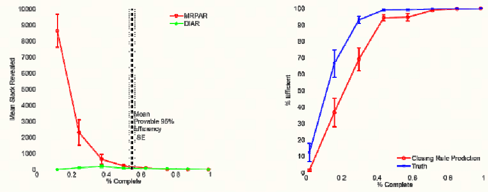




Activity Rules Efficiency Bound

Harvard School of Engineering and Applied Sciences

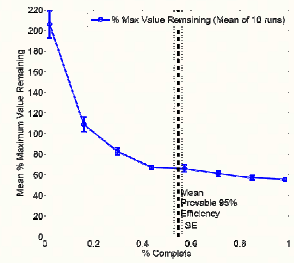
- MRPAR: 'main rocket'
  DIAR: 'course correction'
- Efficiency bound effective



Information Revelation

Harvard School of Engineering and Applied Sciences

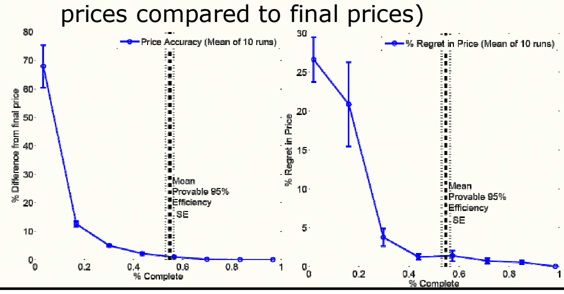
- Bounds retain 'slack'



Price Quality

Harvard School of Engineering and Applied Sciences

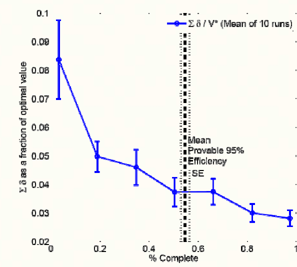
- Prices converge quickly
- Low regret (best trade at intermediate prices compared to final prices)



Pricing Error

Harvard School of Engineering and Applied Sciences

- Linear prices have low error



Results Summary

Harvard School of Engineering and Applied Sciences

- **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

Harvard School of Engineering and Applied Sciences

- 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

Harvard
School of Engineering
and Applied Sciences

Fin

- For more information:
  - http://www.eecs.harvard.edu/~blubin/ice
  - blubin {at} eecs {dot} harvard {dot} edu