

ICE: An Expressive Iterative Combinatorial Exchange[‡]

Benjamin Lubin [†]

Adam I. Juda [†]

Ruggiero Cavallo

Sébastien Lahaie

Jeffrey Shneidman

David C. Parkes [†]

School of Engineering and Applied Sciences

Harvard University

Cambridge, MA 02138

BLUBIN@EECS.HARVARD.EDU

ADAMJUDA@POST.HARVARD.EDU

CAVALLO@EECS.HARVARD.EDU

SLAHAIE@EECS.HARVARD.EDU

JEFFSH@EECS.HARVARD.EDU

PARKES@EECS.HARVARD.EDU

Abstract

We present the design and analysis of the first fully expressive, iterative combinatorial exchange (ICE). The exchange incorporates a tree-based bidding language (*TBBL*) that is concise and expressive for CEs. Bidders specify lower and upper bounds in *TBBL* on their value for different trades and refine these bounds across rounds. These bounds allow price discovery and useful preference elicitation in early rounds, and allow termination with an efficient trade despite partial information on bidder valuations. All computation in the exchange is carefully optimized to exploit the structure of the bid-trees and to avoid enumerating trades. A proxied interpretation of a revealed-preference activity rule, coupled with simple linear prices, ensures progress across rounds. The exchange is fully implemented, and we give results demonstrating several aspects of its scalability and economic properties with simulated bidding strategies.

General Terms: Algorithms, Economics, Theory.

Keywords: Combinatorial exchange, Preference Elicitation, Price Discovery.

1. Introduction

Combinatorial exchanges combine and generalize two different mechanisms: double auctions and combinatorial auctions. In a double auction (DA), multiple buyers and sellers trade units of an identical good (McAfee, 1992). In a combinatorial auction (CA), a single seller has multiple heterogeneous items up for sale (de Vries & Vohra, 2003; Cramton et al., 2006). Each buyer in a CA may have complementarities (“I want *A* and *B*”) or substitutabilities (“I want *A* or *B*”) between goods, and is provided with an expressive bidding language to describe these preferences. A common goal in the design of both DAs and CAs is to implement the efficient allocation, which is the allocation that maximizes total value.

A combinatorial exchange (CE) (Parkes et al., 2001) is a combinatorial double auction that brings together multiple buyers and sellers to trade multiple heterogeneous goods. For example, in an exchange for wireless spectrum, a bidder may declare that he is willing to

[†]. Primary authors.

[‡]. An earlier version of this paper appeared in the Proc. ACM Conference on Electronic Commerce, 2005. The *TBBL* language is also described in a workshop paper (Cavallo et al., 2005)

pay US \$100 million for obtaining licenses for New York City, Boston, and Philadelphia in exchange for Washington DC. Unlike DAs, the participants in a CE can express complex valuations via expressive bids. Unlike CAs, a CE allows distributed initial ownership and handles bidders that are both buying and selling. For instance, a single bidder might want to perform a swap (“I will sell B if I can buy A ”), or a more complex contingent trade.

CEs have received attention in the context of wireless spectrum allocation (Cramton et al., ; Kwerel & Williams, 2002), for airport takeoff and landing slot allocation (Ball et al., 2006; Vossen & Ball, 2006), and for financial markets (Saatcioglu et al., 2001). In all of these domains there are incumbents with property rights, and it is necessary to facilitate a complex multi-way reallocation of resources. Another potential application domain for CEs is to resource allocation in shared distributed systems, such as PlanetLab (Fu et al., 2003). CEs can also find use for task allocation in robot teams, making them a potentially powerful tool to the multi-agent systems community (Dias et al., 2006; Bererton et al., 2003; Gerkey & Mataric, 2002). Finally, CEs have promise as mechanisms for expressive sourcing by multiple bid-takers, perhaps representing different profit centers within an organization; see Sandholm (2007) for related work on expressive sourcing using one-sided CAs.

This paper presents the design of the first *fully expressive, iterative* combinatorial exchange (ICE). The broad goal is to automate price discovery processes to enable efficient trade in complex environments. In designing an iterative exchange, we share the motivation of earlier work on iterative CAs: we wish to mitigate elicitation costs by focusing bidders, in this case through price discovery and activity rules, on relevant trades. Bidders will typically wish to reveal as little information as possible to avoid leaking information to competitors and because of difficult valuation problems; the valuation problem for even a single trade can present a challenging problem for a bidder in complex domains (Sandholm & Boutilier, 2006; Compte & Jehiel, 2007).

In describing the central *design principles* that support the ICE mechanism, we highlight the following:

- A bidder interacts with ICE by first defining a structured representation of his valuation for different trades. Defined in the tree-based bidding language (*TBBL*), this concisely defines the set of trades of interest to the bidder. The bidder must annotate the tree with initial lower and upper bounds on his value for different trades.
- Valuation bounds are used to construct a provisional trade and provisional payments in each round, and drive price feedback to bidders in the form of simple, linear prices. Each bidder receives, as a price quote, a price on each distinct item in the market and is required to update his valuation bounds to make precise which trade is most preferred given these prices.
- ICE is a hybrid design, in between a demand-revealing process and a direct-revelation mechanism, prices guide preference elicitation while directly-specified and expressive bids finally clear the exchange.

By design, ICE is agnostic to the particular rules used to define payments (as distinct from *price feedback* used to drive dynamics) once the exchange terminates. For concreteness, we have chosen to adopt the *Threshold* rule in defining final payments, which minimizes the

ex post incentive to manipulate across all budget-balanced payment rules (Parkes et al., 2001).¹ We highlight the following *technical contributions*:

- The tree-based bidding language (*TBBL*) supports generalized choose operators, valuation bounds, and atomic trades of fixed quantities of a single good on leaves. *TBBL* extends many earlier bidding languages for CAs, embracing bidders that wish to simultaneously buy and sell and providing exponentially more concise representations than OR^* and \mathcal{L}_{GB} (Nisan, 2006; Boutilier, 2002). The structure of the *TBBL* language can be directly encoded within a mixed-integer programming (MIP) formulation of the winner determination problem.
- Despite employing linear prices, the proxied design of ICE provides convergence to the efficient trade with straightforward bidders. A price-based proof using duality theory is available when prices are sufficiently accurate. Otherwise, a direct proof based on reasoning about the upper and lower valuation bounds is always available, even when the combinatorics of the instance preclude a price-based proof.
- Preference elicitation occurs through the combination of two novel activity rules. The first is a *modified revealed-preference activity rule* (MRPAR), and requires each bidder to make precise which trade is most preferred in each round. The second is a *delta improvement activity rule* (DIAR), and requires each bidder to refine his bid to improve price accuracy or prove that no improvement is possible.
- To compute approximate (linear) competitive equilibrium prices given *TBBL* bids we employ constraint generation, with heuristic seeding of constraints together with constraint pooling and optimistic validation to streamline the required computation. Lexicographical minimization and tie breaking to favor prices that approximate the payment rule and then to minimize the difference in prices across items ensures unique and informative prices.

The exchange is fully implemented in Java (with a C-based MIP solver sub-module). We present scalability results showing performance across a wide number of bidders, goods and valuation complexity and benchmarking to provide a qualitative understanding of the characteristics of our mechanism. In performing simulations, we employ a straightforward (non-strategic) bidding agent that adopts a heuristic to modify the bounds in its *TBBL* tree across rounds in meeting the activity rule and without precluding its true valuation. In describing the experimental results, we offer the following highlights:

- The exchange quickly converges to the efficient trade. For an example domain with 100 goods of 20 different types, and 8 bidders with valuation functions containing an average of 112 *TBBL* nodes, the system can prove it has the efficient trade in approximately 7 rounds.

1. The problem of designing truthful, approximately-efficient, and budget-balanced CEs remains unsolved, even for sealed-bid CEs. As and when progress is made on the problem for sealed-bid CEs, this can be directly and immediately leveraged within the ICE framework through appropriate modifications to the winner-determination and payment rules.

- The exchange permits bidders to leave large amounts of the uncertainty in their bids unrefined. In this same domain, we find that the bidders can leave upwards of 62% of their maximum attainable value undefined when the efficient trade is known, and 56% once final payments are determined.
- The price feedback in early rounds is informative. In this same domain, the trade that an average bidder demands at intermediate prices would on average achieve within 11% of the profit of the bidder’s best trade at the final prices.
- The exchange can scale to economically interesting problems. Through extensive use of parallel processing, these same domains complete in an average of 8.5 minutes on a 3.2GHz dual-processor dual-core workstation with 8GB of memory. This includes the time for all winner determination, pricing, and activity rules, as well as the time to simulate agent bidding strategies.

1.1 Related Work

Many ascending-price one-sided CAs are known in the literature (de Vries et al., 2007; Parkes & Ungar, 2000a; Wurman & Wellman, 2000; Mishra & Parkes, 2007; Ausubel & Milgrom, 2002). Direct elicitation approaches, in which bidders respond to explicit queries about their valuations, have also been proposed for one-sided CAs (Conen & Sandholm, 2001; Hudson & Sandholm, 2004; Lahaie & Parkes, 2004; Lahaie et al., 2005).

Of particular relevance to our work are the ascending CAs that are designed to work with simple prices on items (Dunford et al., 2003; Kwasnica et al., 2005). In computing linear prices, we generalize and extend these methods. Building on Rassenti et al. (1982), these earlier papers consider bids on bundles individually, and find prices that are exact on winning bids and minimize the pricing error to losing bids. Generalizing to the *TBBL* expressive language, we compute prices that minimize the worst-case pricing error over all *bidders* (rather than bids on individual trades), considering the most preferred trade consistent with the *TBBL* bid of each bidder. As in Dunford et al. (2003) and Kwasnica et al. (2005) we incorporate additional tie-breaking stages, in our case to lexicographically minimize the error and then to find prices that closely approximate the provisional *payments*. This latter step appears to be novel.

Linear prices are important in practical applications. Such prices are adopted by the FCC in their wireless spectrum auctions (Cramton, 2006), within clock auctions for the procurement of electricity generation (Cramton, 2003), and were an essential part of the proposed design for an airport landing slot auction at Laganardia airport (Ball et al., 2007).

The proxied architecture is inspired by Parkes and Ungar (2000b) and Ausubel and Milgrom (2002) in their work on proxied (ascending) CAs. ICE couples price discovery and a demand-revealing process in early rounds, with direct optimization based on reported valuations to finally clear the exchange. In this sense, ICE can be considered a two-sided generalization of the clock-proxy design of Ausubel et al. (2006), which has an initial stage of linear price discovery followed by a “best-and-final” sealed-bid stage. We compute linear price feedback based on a provisional clearing of the exchange in every round. In ensuring

progress, we adopt a variation on the clock-proxy auction’s revealed-preference activity rule. Activity rules have been shown to be very important in practice.²

Saatcioglu et al. (2001) study the use of sealed-bid combinatorial exchanges for the purpose of contingent trades in financial markets and consider aspects of expressiveness and winner determination. Parkes et al. (2001) studied sealed-bid combinatorial exchanges and introduced the Threshold payment rule. Subsequently, Krych (2003) demonstrated experimentally that the Threshold rule promotes efficient trade. See also Milgrom (2007) for a recent discussion. Dominant strategy DAs are known for unit demand environments (McAfee, 1992) and slightly more expressive domains (Babaioff & Walsh, 2005; Chu & Shen, 2007; Gonen et al., 2007). No dominant strategy mechanisms are known for the general CE problem. Exact efficiency together with budget balance is not possible because of the Myerson-Satterthwaite (1983) impossibility result.

Voucher-based schemes have been proposed as an alternative method to extend one-sided CAs to exchanges (Kwerel & Williams, 2002). Such mechanisms collect all goods from sellers and then run a one-sided auction in which sellers can “buy-back” their own goods with vouchers used to provide a seller with a share of the revenue collected on their own goods. Although voucher-based schemes can facilitate the design of exchanges through one-sided auction technology, the ICE design provides a significant improvement by providing equal and symmetric expressiveness to all participants.

We are not aware of any previous studies of fully expressive iterative CEs. Smith et al. (2002) have previously studied iterative CEs, but can handle only limited expressiveness and adopts a direct-query based approach with an enumerative internal data structure that does not scale. A novel feature in their earlier design (not supported here) is *item discovery*, where the items available to trade need not be known in advance.

Several bidding languages for CAs have previously been proposed, arguably the most compelling of which allow bidders to explicitly represent the logical structure of their valuation over goods via standard logical operators. We refer to these as “logical bidding languages” (Nisan, 2006). Closest in generality to *TBBL* is the \mathcal{L}_{GB} language (Boutilier & Hoos, 2001), which allows for arbitrarily nested levels, combining goods and trades by the standard propositional logic operators, and also provides a *k-of* operator, used to represent a willingness to pay for any k trades it quantifies over. Rothkopf et al. (1998) also describe a restricted tree-based bidding language. In a key insight, Boutilier (2002) specifies a MIP formulation for WD using \mathcal{L}_{GB} , and provides positive empirical performance results using a commercial solver, suggesting the computational feasibility of moving to this more expressive logical language. *TBBL* shares some structural elements with the \mathcal{L}_{GB} language but has important differences in its semantics. In \mathcal{L}_{GB} , the semantics are those of propositional logic, with the same items in an allocation able to satisfy a tree in multiple places. Although this can make \mathcal{L}_{GB} especially concise in some settings, the semantics that we propose provide representational locality, so that the value of one component in a tree can be understood independently from the rest of the tree.

2. For instance, the Milgrom-Wilson activity rule that requires a bidder to be active on a minimum percentage of the quantity of the spectrum for which it is eligible to bid is a critical component of the auction rules used by the FCC for wireless spectrum auctions (Milgrom, 2004).

1.2 Outline

Section 2 introduces preliminary concepts, defining the efficient trade, competitive equilibrium prices and the Threshold payment rule. Section 3 defines a sealed-bid CE, introducing *TBBL* and providing the MIP that is used to solve winner determination. Section 4 extends *TBBL* to allow for valuation bounds and defines the MRPAR and DIAR activity rules. The main theoretical results are also described. Section 5 provides our method to determine price feedback in each round. Section 6 gives a number of illustrative examples of the operation of ICE. Section 7 notes some systems issues in our implementation. Section 8 presents our experimental results. We conclude in Section 9. The Appendix provides an algorithm for each of the two activity rules together with details on the bidding logic used by simulated bidding agents.

2. Preliminaries

The basic environment considers a set of bidders, $N = \{1, \dots, n\}$, who are interested in trading multiple units of distinct, indivisible goods, where the set of different types of goods is denoted $G = \{1, \dots, m\}$. Each bidder has an initial endowment of goods and a valuation for different trades.

In the ICE mechanism, provisional trades and prices are reported to participants in each round, who are then required to update their valuation information (c.f. bids) to meet the activity rules. ICE is a *proxied design* in which proxy agents maintain upper and lower valuation bounds, and interact with bidders in guiding the refinement of these bounds. Each bidder must first provide an initial *TBBL* structure, state his initial allocation of items that are available for trade in the exchange, and ascribe initial upper- and lower bounds to the *TBBL* tree.

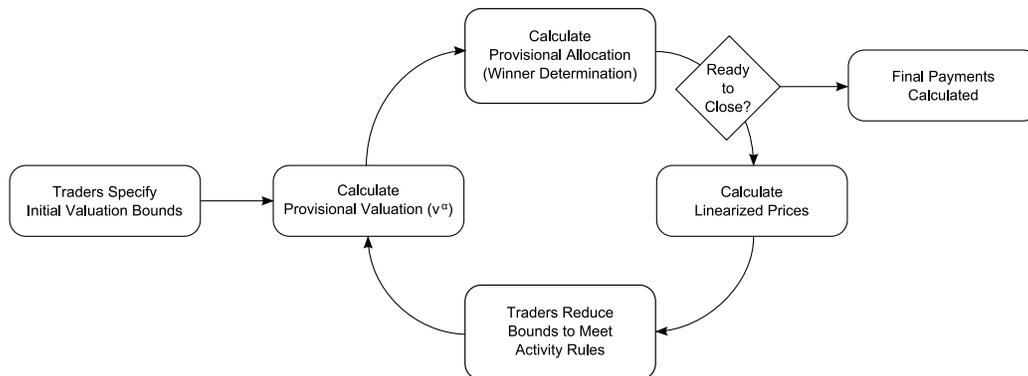


Figure 1: ICE system overview

See Figure 1. In each round, a buyer receives prices and a provisional trade and then interacts with his proxy agent in order to meet the activity rules. The proxy agents are responsible for enforcing these rules. Next, the exchange chooses a provisional valuation profile (denoted $v^\alpha = (v_1^\alpha, \dots, v_n^\alpha)$ in the figure), with the valuation v_i^α for each bidder picked to fall within the bidder's current valuation bounds (and to trend towards the lower valuation bound as progress is made towards determining the final trade). Then, the ex-

change computes a provisional trade and checks whether the conditions for moving to a last-and-final round are satisfied. ICE reports this provisional trade to the bidders via the proxy agents together with a set of prices, one for each good, that approximately balance supply and demand and also provide guidance about final payments. Finally, a new round begins with agents again modifying their reported bounds to meet the activity rules. At conclusion of the last-and-final round, the final payments and the final trade are computed in terms of *lower* valuations. The semantics are that these lower bounds guarantee that a bidder is willing to pay this amount (or receive a negative value as a payment), to complete the trade.

2.1 Definitions: Efficiency, Mechanisms, and Prices

Let $x^0 = (x_1^0, \dots, x_n^0)$ denote the initial endowment of goods, with $x_i^0 = (x_{i1}^0, \dots, x_{im}^0)$ and $x_{ij}^0 \geq 0$ for good $j \in G$ to indicate the number of units of good j initially held by bidder i . A trade $\lambda = (\lambda_1, \dots, \lambda_n)$ denotes the *change* in allocation, with $\lambda_i = (\lambda_{i1}, \dots, \lambda_{im})$ and $\lambda_{ij} \in \mathbb{Z}$ to denote the change in the number of units of item j to bidder i . Let $M = \sum_{i \in N} \sum_{j \in G} x_{ij}^0$ denote the total supply in the exchange. Write $i \in \lambda$ to denote that bidder i is active in the trade, i.e. trades at least one item.

2.1.1 THE EFFICIENT TRADE

Each bidder has a value $v_i(\lambda_i) \in \mathbb{R}$ for a trade λ_i . This value can be positive or negative, and represents the *change in value* between the final allocation $x_i^0 + \lambda_i$ and the initial allocation x_i^0 . The valuation and initial allocation information is private to each bidder, and we assume that there are no externalities, so that each bidder's value depends only on their individual trade. We assume *free disposal*, so that $v_i(\lambda'_i) \geq v_i(\lambda_i)$ for trade $\lambda'_i \geq \lambda_i$, i.e. for which $\lambda'_{ij} \geq \lambda_{ij}$ for all j . Let $v(\lambda) = \sum_i v_i(\lambda_i)$.

Utility is *quasi-linear*, with $u_i(\lambda_i, p) = v_i(\lambda_i) - p$ for trade λ_i and payment $p \in \mathbb{R}$. This implies that bidders are modeled as being risk neutral and assumes that there are no budget constraints. The payment, p , can be negative, indicating the bidder may receive a payment for the trade. We use the term *payoff* interchangeably with *utility*.

An instance of the CE problem is defined by tuple (v, x^0) , i.e. a valuation profile $v = (v_1, \dots, v_n)$ and an initial allocation $x^0 = (x_1^0, \dots, x_n^0)$. The *efficient trade*, λ^* , maximizes the total increase in value to bidders across all feasible trades:

Definition 1 *Given CE instance (v, x^0) , the efficient trade λ^* solves*

$$\text{EFF}(v, x^0) = \max_{(\lambda_1, \dots, \lambda_n)} \sum_i v_i(\lambda_i) \tag{1}$$

$$\text{s.t. } \lambda_{ij} + x_{ij}^0 \geq 0, \quad \forall i, \forall j \tag{2}$$

$$\sum_i \lambda_{ij} = 0, \quad \forall j \tag{3}$$

$$\lambda_{ij} \in \mathbb{Z}$$

Constraints (2) ensure that no bidder sells more items than it has in its initial allocation. By free disposal, we can impose strict balance in the supply and demand of goods at the solution in Constraints (3), i.e. we can allocate unwanted items to any bidder. We adopt

$\mathcal{F}(x^0)$ to denote the set of *feasible trades*, given these constraints and given an initial allocation x^0 , and $\mathcal{F}_i(x^0)$ for the set of feasible trades to bidder i . Note that valuation function v_i cannot be explicitly represented as a value for each possible trade because the number of trades scales as $O(m^n)$. The *TBBL* language (introduced in Section 3.1) leads to a concise formulation of the efficient trade problem as a mixed-integer program.

2.1.2 THE VCG MECHANISM AND THRESHOLD PAYMENTS

Mechanism design (MD) provides a methodology for implementing socially desirable outcomes in multi-agent systems with self-interested agents and private information about valuations.

In a *strategyproof* mechanism truthful reporting is a dominant-strategy equilibrium. The *Vickrey-Clarke-Groves* (VCG) mechanism (Krishna, 2002, e.g.) is a central positive result in MD, and provides a strategyproof and efficient mechanism, so that the efficient trade is implemented in equilibrium. In application to CEs, the VCG mechanism collects the following payment from each bidder:

$$p_{\text{vcg},i}(\hat{v}) = \hat{v}(\lambda_i^*) - (V(\hat{v}) - V_{-i}(\hat{v})), \quad (4)$$

where λ^* is the efficient trade, $V(\hat{v})$ is the reported value of this trade and $V_{-i}(\hat{v})$ is the reported value of the trade that would be selected without bidder i , i.e. the solution to $\text{EFF}(v_{-i})$ for $v_{-i} = (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$. Here we use \hat{v} to denote the *reported* valuations of bidders, and these need not be the same as the true valuations.

Let us refer to $\Delta_{\text{vcg},i}(\hat{v}) = V(\hat{v}) - V_{-i}(\hat{v})$ as the *VCG discount*. The problem with the VCG mechanism in the context of a CE is that it runs at a budget deficit, with:

$$\sum_i \Delta_{\text{vcg},i}(\hat{v}) > V(\hat{v}), \quad (5)$$

so that the total payments collected from bidders are negative. Exact budget-balance would require $\Delta_{\text{vcg},i}(\hat{v}) = V(\hat{v})$ for all reports \hat{v} .³

The *Threshold rule* (Parkes et al., 2001) addresses this budget deficit concern. The Threshold rule defines budget-balanced payments that minimize the maximal difference (across all bidders) between payments and the payments in the VCG outcome.

Definition 2 *The Threshold rule defines payments $p_{\text{thresh},i}(\hat{v}) = \hat{v}_i(\lambda_i^*) - \Delta_i(\hat{v})$, where $\Delta_i(\hat{v}) = (\Delta_1(\hat{v}), \dots, \Delta_n(\hat{v}))$ is set to minimize $\max_i(\Delta_{\text{vcg},i}(\hat{v}) - \Delta_i(\hat{v}))$ subject to $\Delta_i(\hat{v}) \leq \Delta_{\text{vcg},i}(\hat{v})$ for all i and $\sum_i \Delta_i(\hat{v}) \leq V(\hat{v})$.*

Threshold payments minimize the maximal *ex post* incentive to manipulate, while maintaining budget balance (Parkes et al., 2001). Experimental analysis has confirmed that Threshold payments promote allocative efficiency in a restricted and approximate Bayes-Nash equilibrium (Krych, 2003).

3. Consider a simple example in which a buyer has value \$8 for an item and a seller has value -\$4, where these values define the *change* in value to a bidder for the trade. The VCG payments are \$4 and -\$8 respectively and the exchange runs at a deficit of \$4. Equivalently, the VCG discount is \$4 to the buyer (= \$4 - 0) and \$4 to the seller (= \$4 - 0).

2.1.3 COMPETITIVE EQUILIBRIUM PRICES

Linear prices, $\pi = (\pi_1, \dots, \pi_m)$ define a price on each good, so that the price to bidder i on a trade, λ , is defined as $p^\pi(\lambda_i) = \sum_j \lambda_{ij} \pi_j = \lambda_i \cdot \pi$, play an important role in ICE. We adopt linear prices throughout.

Definition 3 *Linear prices π are competitive equilibrium prices for CE problem (v, x^0) if there is some feasible trade $\lambda \in \mathcal{F}(x^0)$ such that:*

$$v_i(\lambda_i) - p^\pi(\lambda_i) \geq v_i(\lambda'_i) - p^\pi(\lambda'_i), \quad \forall \lambda'_i \in \mathcal{F}_i(x^0), \quad \forall i, \quad (6)$$

where $\mathcal{F}(x^0)$ is the set of feasible trades given initial allocation x^0 and $p^\pi(\lambda_i) = \lambda_i \cdot \pi$. We say that such a trade, λ , is **supported** by prices π .

Theorem 1 *Any trade λ supported by competitive equilibrium prices π is an efficient trade.*

Proof: Fix instance (v, x^0) . Consider (λ, π) . By Eq. (6) we have

$$\sum_i v_i(\lambda_i) - \sum_i p^\pi(\lambda_i) \geq \sum_i v_i(\lambda'_i) - \sum_i p^\pi(\lambda'_i), \quad (7)$$

for all $\lambda'_i \in \mathcal{F}_i(x^0)$. We show that $\sum_i p^\pi(\lambda'_i) = 0$ for all $\lambda'_i \in \mathcal{F}_i(x^0)$ for prices $p^\pi(\lambda'_i) = \lambda'_i \cdot \pi$. For this, define the disaggregated trade $\Lambda_{ii'}(j) \in \mathbb{Z}$, such that $\Lambda_{ii}(j) = 0$ for all j and $\sum_{i' \neq i} \Lambda_{ii'}(j) = \lambda_{ij}$ for all $i \in N$, all $j \in G$. Where there are multiple such Λ , just break ties at random. Given this, $p_i(\lambda'_i) = \sum_j \lambda'_{ij} \pi_j = \sum_j \sum_{i' \neq i} \Lambda_{ii'}(j) \pi_j$, and so

$$\sum_i p_i(\lambda'_i) = \sum_i \sum_j \sum_{i' \neq i} \Lambda_{ii'}(j) \pi_j = \sum_j \left(\sum_i \sum_{i' \neq i} \Lambda_{ii'}(j) \right) \pi_j = 0, \quad (8)$$

where the final step follows because $\sum_i \sum_{i' \neq i} \Lambda_{ii'}(j) = \sum_i \lambda_{ij} = 0$ for all $j \in G$. This completes the proof. \square

In practice it is unlikely that exact, linear EQ prices exist in a CE. Instead, it is useful to define the concept of approximate EQ prices:

Definition 4 *Linear prices π are $\vec{\delta}$ -approximate competitive equilibrium prices for CE problem (v, x^0) , with respect to $\vec{\delta} = (\delta_1, \dots, \delta_n) \in \mathbb{R}_{\geq 0}^n$, if there is some feasible trade $\lambda \in \mathcal{F}(x^0)$ such that:*

$$v_i(\lambda_i) - p^\pi(\lambda_i) + \delta_i \geq v_i(\lambda'_i) - p^\pi(\lambda'_i), \quad \forall \lambda'_i \in \mathcal{F}_i(x^0), \quad \forall i. \quad (9)$$

At $\vec{\delta}$ -approximate EQ prices, every bidder i is within $\delta_i \geq 0$ of maximizing its utility at the prices, given trade λ . Let $C(\lambda, x^0, \vec{\delta})$ denote the *condition number* for instance λ given valuation profile v , initial allocation x^0 and price errors $\vec{\delta}$, and defined as

$$C(\lambda, x^0, \vec{\delta}) = \max_{\lambda' \in \mathcal{F}(x^0)} \sum_{i \in \lambda' \cup \lambda} \delta_i \quad (10)$$

Clearly, $C(\lambda, x^0, \vec{\delta}) \leq \sum_i \delta_i$. Say that a trade is *z-approximate* if the total value is within z of the total value of the efficient trade.

Theorem 2 *Any trade λ supported by $\vec{\delta}$ -approximate competitive equilibrium prices π is an $C(\lambda, x^0, \vec{\delta})$ -approximate efficient trade.*

Proof: Fix instance (v, x^0) and consider (λ, π) . For any trade $\lambda' \neq \lambda$ we have

$$\sum_{i \in \lambda \cup \lambda'} [v_i(\lambda_i) - p^\pi(\lambda_i) + \delta_i] \geq \sum_{i \in \lambda \cup \lambda'} [v_i(\lambda'_i) - p^\pi(\lambda'_i)], \quad (11)$$

by $\vec{\delta}$ -EQ prices. Since $\sum_{i \in \lambda \cup \lambda'} p^\pi(\lambda_i) = \sum_{i \in \lambda \cup \lambda'} p^\pi(\lambda'_i) = 0$ by a simple variation on the argument in the proof of Theorem 1, we have

$$\sum_i v_i(\lambda_i) + \sum_{i \in \lambda \cup \lambda'} \delta_i \geq \sum_i v_i(\lambda'_i) \quad (12)$$

Fix now $\lambda := \lambda^*$, for efficient trade, λ^* , then

$$\sum_i v_i(\lambda_i) + C(\lambda, x^0, \vec{\delta}) \geq \sum_i v_i(\lambda_i^*), \quad (13)$$

because $C(\lambda, x^0, \vec{\delta}) \geq \sum_{i \in \lambda \cup \lambda'} \delta_i$. □

Let δ -EQ prices, for some $\delta \in \mathbb{R}_{\geq 0}$ denote $\vec{\delta}$ -EQ prices where $\vec{\delta} = (\delta, \dots, \delta)$. In this case,

$$C(\lambda, x^0, \vec{\delta}) \leq \min(2A^\#(x^0), n)\delta \leq \min(2 \min(M, n), n)\delta = 2 \min(M, \frac{n}{2})\delta, \quad (14)$$

where $A^\#(x^0)$ is the maximal number of bidders that can trade in a feasible trade given x^0 , the second inequality follows because $A^\#(x^0) < \min(M, n)$; no more bidders can trade than there are number of goods to trade or bidders in the market.

We have the following simple corollary:

Corollary 1 *Any trade λ supported by δ -approximate EQ prices π is a $2 \min(M, \frac{n}{2})\delta$ -approximate efficient trade.*

3. Step One: A *TBBL*-Based Sealed-Bid Combinatorial Exchange

We first flesh out the details for a non-iterative, *TBBL*-based CE in which each bidder submits a sealed bid in the *TBBL* language.

3.1 Tree-Based Bidding Language

The tree-based bidding language (*TBBL*) is designed to be expressive and concise, entirely symmetric with respect to buyers and sellers, and to easily provide for bidders that are both buying and selling goods; i.e., ranging from simple swaps to highly complex trades.

Bids are expressed as annotated *bid trees*, and define a bidder's *change in value* for all possible trades. The main feature of *TBBL* is that it has a general "interval-choose" logical operator on internal nodes coupled with a rich semantics for propagating values within the tree. Leaves of the tree are annotated with traded items and all nodes are annotated with changes in values (either positive or negative). *TBBL* is designed such that these changes

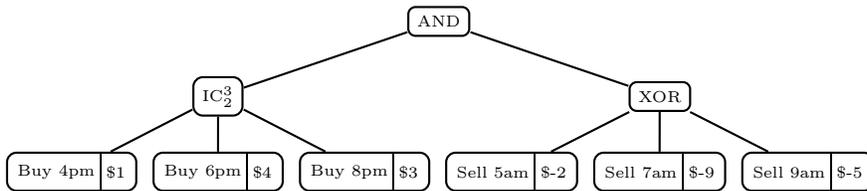


Figure 2: A simple *TBBL* tree for an airline interested in trading landing slots.

in value are expressed on *trades* rather than the total value of allocations.⁴ We remark that *TBBL* is exponentially more concise than OR^* and \mathcal{L}_{GB} for preferences that are realistic in important domains for CEs (Cavallo et al., 2005).

Consider bid tree T_i from bidder i . Write $\beta \in T_i$ to denote a node in the tree, and let $v_i(\beta) \in \mathbb{R}$ denote the value specified at node β (perhaps negative). Let $\text{Leaf}(T_i) \subseteq T_i$ be the subset of nodes representing the leaves of T_i and let $\text{Child}(\beta) \subseteq T_i$ denote the children of node β . All nodes except leaves are labeled with the *interval-choose* operator $\text{IC}_x^y(\beta)$. Each leaf β is labeled as a *buy* or *sell*, with units $q_i(\beta, j) \in \mathbb{Z}$ for the good j associated with leaf β , and $q_i(\beta, j') = 0$ otherwise.

The IC operator defines a range on the number of children that can be, and must be, satisfied for node β to be satisfied: an $\text{IC}_x^y(\beta)$ node (where x and y are non-negative integers) indicates that the bidder is willing to pay for the satisfaction of at least x and at most y of its children. With suitable values for x and y the operator can include many logical connectors. For instance: $\text{IC}_2^2(\beta)$ on node β with 2 children is equivalent to an AND operator; $\text{IC}_1^3(\beta)$ on node β with 3 children is equivalent to an OR operator; and $\text{IC}_1^2(\beta)$ on node β with 2 children is equivalent to an XOR operator. This equivalence implies that *TBBL* can directly express the XOR, OR and XOR/OR languages (Nisan, 2006).

More formally, the *satisfaction* of an $\text{IC}_x^y(\beta)$ node is defined by the following two rules:

R1 Node β with $\text{IC}_x^y(\beta)$ may be *satisfied* only if at least x and at most y of its children are *satisfied*.

R2 If some node, β , is *not satisfied*, then none of its children may be *satisfied*.

One can consider **R1** as a “first pass” that defines a set of candidates for satisfaction. This candidate set is then refined by **R2**. Besides defining how value is propagated, by virtue of **R2** our logical operators act as *constraints* on what trades are *acceptable* and provide necessary and sufficient conditions.⁵

Given a tree T_i , the (change in) value of a trade λ is defined as the sum of the values on all satisfied nodes, where the set of satisfied nodes is chosen to provide the *maximal* total value. In the simple case of unit quantities on leaves, every new item allocated in a trade

4. In working through numerous examples we frequently found it very cumbersome to capture even simple trades in languages that specified values on allocations, as is the case with all existing languages.

5. **R1** naturally generalizes the approach taken in \mathcal{L}_{GB} , where an internal node is satisfied according to its operator and the subset of its children that are satisfied. The semantics of \mathcal{L}_{GB} , however, treat logical operators only as a way of specifying when “added value” (positive or negative) results from attaining combinations of goods. In principle, any trade is “acceptable” to an \mathcal{L}_{GB} bid. By also adopting **R2** we take a different approach.

allows an additional buy leaf to be satisfied while every item sold in a trade requires that an additional sell leaf is considered satisfied.

Let $sat_i(\beta) \in \{0, 1\}$ denote whether node β in tree T_i of bidder i is satisfied, with $sat_i = \langle sat_i(\beta), \forall \beta \in T_i \rangle$. Solution sat_i is valid for tree T_i and trade λ_i , written $sat_i \in valid(T_i, \lambda_i)$, if and only if **R1** and **R2** hold for all nodes:

$$\sum_{\beta \in Leaf(T_i)} q_i(\beta, j) sat_i(\beta) \leq \lambda_{ij}, \quad \forall i \in N, \forall j \in G \quad (15)$$

$$IC_x(\beta) sat_i(\beta) \leq \sum_{\beta' \in Child(\beta)} sat_i(\beta') \leq IC_y(\beta) sat_i(\beta), \quad \forall \beta \in \{T_i \setminus Leaf(T_i)\}, \forall i \in N \quad (16)$$

A set of leaves are satisfied given trade λ_i if the total increase in quantity of each item is no greater than the total number of units awarded in the trade by Eq. (15). Free disposal is implicit here: it is allowed for the trade to assign additional units of an item over and above that required in the bid tree. This works for sellers as well as buyers: for sellers a trade is negative and this requires that the total number of items indicated sold in the tree is at least the total number sold as defined in the trade. Eq. (16) enforces the *interval-choose* constraints, by ensuring that no more and no less than the appropriate number of children is *satisfied* for any node that is *satisfied*. Further, this equation ensures not only this “downwards-propagation”, but also “upwards-propagation”: any time a node other than the root is satisfied then its parent must also be satisfied.

Given these constraints, the total value of trade λ_i , given bid-tree T_i from bidder i , is defined as the solution to an optimization problem:

$$\begin{aligned} v_i(T_i, \lambda_i) &= \max_{sat_i} \sum_{\beta \in T_i} v_i(\beta) sat_i(\beta) \\ &\text{s.t. (15), (16)} \end{aligned} \quad (17)$$

Given some tree T_i , is useful to adopt notation $\beta \in \lambda_i$ to denote a leaf $\beta \in T_i$ that is satisfied by trade λ_i .

Example 1 Consider an airline operating out of a slot-controlled airport that already owns several morning landing slots, but none in the evening. In order to expand its business the airline wishes to acquire at least two and possibly three of the evening slots. However, it needs to offset the cost of this purchase by selling one of its morning slots. Figure 2 shows a TBBL valuation tree for expressing this kind of swap.

3.2 Winner Determination

The problem of determining an efficient trade given bids is called the *winner determination* (WD) problem. A special case of this problem is equivalent to the weighted set-packing problem,⁶ and because of this it can be seen that the WD problem is an NP-hard optimization problem (Rothkopf et al., 1998). This is true even for one-sided CAs. On the other

6. For this, suppose there is a single seller willing to sell any number of some set of items and multiple buyers, where each buyer wants to buy a single trade of items (represented as a TBBL tree with a single AND node at the root and the value of the trade at the root.)

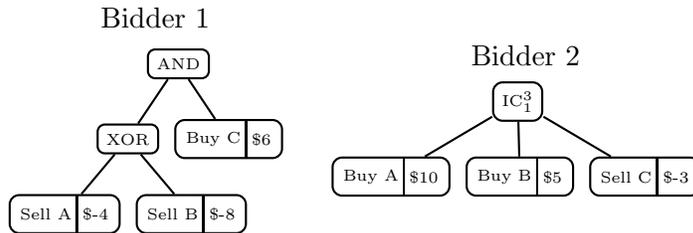


Figure 3: Two bidders and three items $\{A, B, C\}$. The efficient trade is for bidder 1 to sell A and buy C.

hand, it has been demonstrated that very large problem instances can be solved through tree search in economically feasible solution times (de Vries & Vohra, 2003; Boutilier, 2002; Sandholm, 2006), and in particular via a MIP formulation. We follow this route here.

We can formulate the WD problem for CEs as a MIP and solve with a commercial strength solver.⁷ Given bid trees $T = (T_1, \dots, T_n)$ and initial allocation x^0 , we formulate the WD problem as:

$$\begin{aligned} \text{WD}(T, x^0) : \max_{\lambda, \text{sat}} \sum_i \sum_{\beta \in T_i} v_i(\beta) \text{sat}_i(\beta) \\ \text{s.t. } (2), (3), \text{sat}_i(\beta) \in \{0, 1\}, \lambda_{ij} \in \mathbb{Z} \\ \text{sat}_i \in \text{valid}(T_i, \lambda_i), \quad \forall i, \end{aligned}$$

where $\text{sat} = (\text{sat}_1, \dots, \text{sat}_n)$ and sat_i defines the satisfaction of each node $\beta \in T_i$.

All goods are assigned by Eq. (2), but this does not affect the value of the solution by free-disposal and constraint (15), which allows for additional items to be allocated to any bidder. It is noteworthy that the tree structure is explicit in the MIP formulation: we have decision variables to represent the satisfaction of nodes, and capture the logic of the *TBBL* language through linear constraints. There are $O(nB + mn)$ variables, where B is the maximal number of nodes in any bid tree, and $O(mn)$ constraints.

3.3 Illustrative Example: *TBBL*, WD and Threshold

In this section, we provide a simple example that illustrates the various parts of the exchange covered thus far.

Example 2 Consider the two bidders in Figure 3. Bidder 1 will potentially sell one of his items (A or B) if he can get Bidder 2's item, C, at the right price. Bidder 2 is interested in buying one or both of Bidder 1's goods and also selling his own good. Lets consider each of the possible trades: If Bidder 1 trades A for C he gets \$2 of value and Bidder 2 gets \$7. If Bidder 1 trades B for C he gets \$-2 of value and Bidder 2 gets \$2. And if no trade occurs

7. In our case we adopt ILOG's CPLEX solver, www.ilog.com. Algorithms to optimally solve MIP formulations of optimization problems adopt branch-and-bound tree search with linear-programming (LP) relaxations used to generate admissible bounds, algebraic pre-processing, automatic cut generation to strengthen the LP bounds, branch heuristics and primal (local-search) heuristics. See Nemhauser and Wolsey (1999) for a good reference.

both bidders get \$0 value. Therefore the efficient trade is to swap A for C as determined by the MIP described in Section 3.2.

Because the efficient trade creates a surplus of \$9 and removing either bidder results in the null trade, both bidders have a Vickrey discount of \$9. Thus if we use VCG payments, Bidder 1 pays $\$2 - \$9 = \$-7$ and Bidder 2 pays $\$7 - \$9 = \$-2$ and the exchange runs at a deficit. The Threshold payment rule chooses payments that minimally deviate from VCG while maintaining budget balance. This minimization reduces the discounts to \$4.50, and thus Bidder 1 pays $\$2 - \$4.50 = \$-2.50$ and Bidder 2 pays $\$7 - \$4.50 = \$2.50$.

4. Step Two: Making the Exchange Iterative

Having defined a sealed-bid, *TBBL*-based exchange we can now modify the design to make it iterative, allowing bidders to tighten bounds on their valuations in response to price feedback provided in each round. Rather than provide an exact valuation for all interesting trades, a bidder can annotate a single *TBBL* tree with upper and lower bounds and refine this information in response to price feedback.

ICE uses linear prices to drive preference elicitation across rounds through two price-based activity rules. The algorithm for determining prices in each round is presented until Section 5. For now we just assume that such prices exist. The first activity rule is the *Modified Revealed-Preference Activity Rule* (MRPAR).⁸ MRPAR by itself is sufficient to provide efficiency claims when prices are suitably accurate. The second activity rule is the *Delta Improvement Activity Rule* (DIAR) and is used to elicit value information in order to reduce pricing error. The use of a rule such as DIAR appears to be novel, and what is interesting here is that MRPAR coupled with DIAR ensures efficiency in all cases.

For the theoretical analysis of convergence to efficiency, we assume *straightforward bidders*, by which we mean *a bidder that always retains its true valuation within its valuation bounds*. (All results could equivalently be phrased in terms of efficiency claims with respect to reported valuations.) Section 4.1 gives a brief overview of a single round of ICE. Section 4.2 extends *TBBL* to allow for upper and lower value bounds. Sections 4.3 and 4.4 describe MRPAR and DIAR, with computational details presented in the Appendix. Section 4.5 provides a formal claim about the efficiency of ICE.

4.1 Overview

The exchange proceeds in rounds $t \in \{0, 1, \dots\}$, moving eventually to a last-and-final round. Each bidder maintains a lower and upper bound on his valuation, denoted \underline{v}_i and \bar{v} . For this we extend *TBBL* to allow nodes to be annotated with value bounds. Let $\text{WD}(v)$ denote the winner determination problem given valuation profile $v = (v_1, \dots, v_n)$. Recall that $v^\alpha = (v_1^\alpha, \dots, v_n^\alpha)$ is the provisional valuation profile, as selected by the exchange in some round based on reported bounds, and used to determine prices and drive progress.

ICE is parameterized by a target approximation error. In each round the exchange goes through the following steps:

8. MRPAR is loosely based around the activity rule advocated by Ausubel et al. (2006) for a clock-proxy auction for a one-sided CA. In that context, these authors argue that a RP activity rule is preferred to simpler quantity-based rules, for instance in removing incentives for “parking” on some large set of low price goods in order to retain the ability to snipe with large bid just as an auction is about to close.

1. If this is the last-and-final round then implement the trade that solves $\text{WD}(\underline{v})$ and collect Threshold payments defined on valuations \underline{v} . STOP.

ELSE,

2. Solve $\text{WD}(\underline{v})$ to obtain $\underline{\lambda}$. Use valuation bounds and prices to determine a bound ω^{eff} , on allocative efficiency of $\underline{\lambda}$. If this bound is within a target approximation error then the next round is last-and-final.
3. Set $\alpha \in [0, 1]$, with α trending to 1 as ω^{eff} trends to 1, and provisional valuation profile $v^\alpha = (v_1^\alpha, \dots, v_n^\alpha)$, where $v_i^\alpha(\lambda_i) = \alpha \underline{v}_i(\lambda_i) + (1 - \alpha) \bar{v}_i(\lambda_i)$, expressed with a *TBBL* tree in which the value on node $\beta \in T_i$ is $v_i^\alpha(\beta) = \alpha \underline{v}_i(\beta) + (1 - \alpha) \bar{v}_i(\beta)$.
4. Solve $\text{WD}(v^\alpha)$ to find *provisional trade* λ^α , and determine Threshold payments for provisional valuation profile, v^α .
5. Compute linear prices, $\pi \in \mathbb{R}_{\geq 0}^m$, that are approximate CE prices given valuations v^α and trade λ^α , breaking ties to best approximate the provisional Threshold payments and finally to minimize the difference in price between items.
6. Report (λ_i^α, π) to each bidder $i \in N$, and whether or not the next round is last-and-final.

In transitioning to the next round, the proxy agents are responsible for guiding bidders to make refinements to their lower and upper bound valuations to meet the *activity rules*.

Remark: In presenting the MRPAR and DIAR rules that define the preference elicitation process in ICE we will not specify explicit consequences of failing to meet an activity rule. One simple possibility is that the default action is to automatically set the upper valuation bound on every node in a bid tree to the maximum of the provisional price on a node and the lower-bound value on that node.⁹ This is entirely analogous to when a bidder in an ascending-clock auction stops bidding at a price: he is not permitted to bid at a higher price again in future rounds.

4.2 Extending *TBBL* to Allow Upper and Lower Bounds

We extend *TBBL* to allow bidder i to report a lower and upper value $(\underline{v}_i(\beta), \bar{v}_i(\beta))$ on each node $\beta \in T_i$, which in turn induce valuation functions $\underline{v}_i(T_i, \lambda_i)$ and $\bar{v}_i(T_i, \lambda_i)$, using the exact same semantics as in Eq. (17). The bounds on a trade can be interpreted as bounding the payment that the bidder considers acceptable. The bidder commits to complete the trade for a payment less than or equal to the lower-bound and to refuse to complete a trade for any payment greater than the upper-bound. The exact value, and thus true willingness-to-pay remains unknown except when $\underline{v}_i(\beta) = \bar{v}_i(\beta)$ on all nodes.

We say that bid-tree T_i for bidder i is *well-formed* if $\underline{v}_i(\beta) \leq \bar{v}_i(\beta)$ for all nodes $\beta \in T_i$. The following simple lemma is useful:

Lemma 1 *Given a well-formed tree, T , then $\underline{v}_i(T_i, \lambda_i) \leq \bar{v}_i(T_i, \lambda_i)$ for all trades.*

9. Here, the provisional price on a node is defined as the minimal total price across all feasible trades for which the subtree rooted at the node is satisfied.

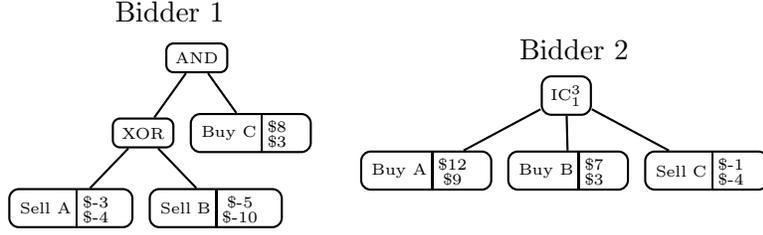


Figure 4: Two bidders, each with partial value information defined on their bid tree. One can already prove that the efficient trade is for bidder 1 to sell A and buy C .

Proof: Suppose there is some λ_i for which $\underline{v}_i(T_i, \lambda_i) > \bar{v}_i(T_i, \lambda_i)$. Then, $\max_{\text{sat}_i \in \text{valid}(T_i, \lambda_i)} \sum_{\beta \in T_i} \underline{v}_i(\beta) \text{sat}_i(\beta) > \max_{\text{sat}_i \in \text{valid}(T_i, \lambda_i)} \sum_{\beta \in T_i} \bar{v}_i(\beta) \text{sat}_i(\beta)$. But, this is a contradiction because the trade λ' that defines $\underline{v}_i(T_i, \lambda_i)$ is still feasible with upper bounds \bar{v}_i , and $\bar{v}_i(\beta) \geq \underline{v}_i(\beta)$ for all nodes β in a well-formed tree. \square

The efficient trade can often be determined with only partial information about agent valuations. Consider the following example.

Example 3 Consider Figure 4. The structure of our two bidders' tree is the same as in Example 2, but now the nodes are annotated with bounds. Let $x \in [3, 8]$ denote bidder 1's true value for "buy C " and $y \in [-4, -1]$ denote bidder 2's true value for "sell C ." The three feasible trades are: (1) trade A and C , (2) trade B and C , (3) no trade. The first trade is already provably efficient. Fixing x, y , its minimal value is $-4 + 9 + x - y$ and this is at least $-5 + 7 + x - y$, the value of the second trade. Moreover, its worst-case value is $-4 + 9 + 3 - 4 \geq 0$, the value of the null trade.

Given a bid tree T_i , it will be useful to define the **perturbed valuation with respect to a trade** λ_i , by assigning the following values to each node β :

$$\tilde{v}_i(\beta) = \begin{cases} \underline{v}_i(\beta) & , \text{ if } \beta \in \text{sat}(\lambda_i) \\ \bar{v}_i(\beta) & , \text{ otherwise.} \end{cases} \quad (18)$$

Intuitively, value \tilde{v}_i (where the dependence on λ_i is kept implicit in the notation) represents the worst-case valuation given that agent i is assigned trade λ_i and bid tree T_i .

4.3 Modified Revealed-Preference Activity Rule (MRPAR)

Activity rules in ICE are used for preference elicitation in each round of ICE.¹⁰ MRPAR is based on a simple idea. We require bidders to refine their valuation bounds in each round, so that there is some trade that is optimal (i.e., maximizes surplus) for the bidder given the current prices and for all possible valuations consistent with the bounds. In describing MRPAR (soon refined to δ -MRPAR), it is convenient to write $v'_i \in T_i$ for $TBBL$ tree T_i to

10. Without an activity rule, a rational bidder would likely wait until the very last moment to revise his valuation information, free-riding on the price discovery enabled by the bids of other participants. If every bidder were to behave this way then the exchange would reduce to a sealed-bid mechanism. This problem has been evocatively described as the "snake in the grass" problem. See Kwerel's forward in Milgrom (2004).

denote that valuation v'_i is consistent with the value bounds in the tree. If the bounds are tight everywhere, then v'_i is exactly the valuation function defined by tree T_i .

A simple RPAR rule imposes constraints on bid tree T_i given prices π . It requires that *there is enough information in valuation bounds to establish that one trade is weakly preferred to all other trades at the prices*, so that:

$$\exists \check{\lambda}_i \in \mathcal{F}_i(x^0) \text{ s.t. } v'_i(\check{\lambda}_i) - p^\pi(\check{\lambda}_i) \geq v'_i(\lambda'_i) - p^\pi(\lambda'_i), \quad \forall v'_i \in T_i, \forall \lambda'_i \in \mathcal{F}_i(x^0) \quad (\text{RPAR})$$

Note that a bidder can always meet this rule by defining an *exact* valuation \hat{v}_i and tight value bounds on every node in his bid tree; in this case, trade $\lambda_i \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} [\hat{v}_i(\lambda_i) - p^\pi(\lambda_i)]$ satisfies RPAR.

Throughout this section, when we discuss the accuracy of prices we mean the accuracy of prices given the current provisional valuations and provisional trade. To understand a subtle challenge in using RPAR to establish the efficiency of the provisional trade in ICE, let us first suppose a good case, which is provided when the prices π are *strict* EQ prices for $(v^\alpha, \lambda^\alpha)$, which is defined so that,

$$v_i^\alpha(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) > v_i^\alpha(\lambda'_i) - p^\pi(\lambda'_i), \quad \forall \lambda'_i \in \mathcal{F}_i(x^0), \forall i \in N. \quad (19)$$

Theorem 3 *If prices π are strict EQ prices for provisional valuation profile v^α and trade λ^α , and every bidder i retains v_i^α in his bid tree after meeting RPAR, then trade λ^α is efficient when bid trees contain true bidder valuations.*

Proof: Fix bidder i . Let $\check{\lambda}_i$ denote the trade that satisfies RPAR. Because v_i^α is consistent with the revised bid tree of bidder i , we have:

$$v_i^\alpha(\check{\lambda}_i) - p^\pi(\check{\lambda}_i) \geq v_i^\alpha(\lambda'_i) - p^\pi(\lambda'_i), \quad \forall \lambda'_i \in \mathcal{F}_i(x^0). \quad (20)$$

Moreover, we must have $\check{\lambda}_i = \lambda_i^\alpha$, because $v_i^\alpha(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) > v_i^\alpha(\lambda'_i) - p^\pi(\lambda'_i)$ by the strictness of prices. Instantiating RPAR with this trade, and with true valuations $v_i \in T_i$ (since bidders are straightforward), we have:

$$v_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) \geq v_i(\lambda'_i) - p^\pi(\lambda'_i), \quad \forall \lambda'_i \in \mathcal{F}_i(x^0), \quad (21)$$

from which prices p^π are EQ prices with respect to true valuations. The efficiency claim then follows from the welfare theorem, Theorem 1. \square

In particular, the provisional trade is efficient given strict EQ prices when every bidder meets the rule without modifying his bounds in any way. However, to see that strict EQ prices are required for this result, consider the following example:

Example 4 *The TBBL trees shown in Figure 5 will have no trade occur at the truthful valuation (which is indicated in bold between the value bounds). However, suppose $\alpha = 0$ so that at the provisional valuations it is efficient for A to be traded. Prices $\pi = (6, 2)$ are EQ (but not strict EQ) prices given v^α and λ^α , with the buyer indifferent between buying A and buying B and the seller indifferent between selling A , selling A and B , or making no sale. The buyer passes RPAR without changing his bounds because the bounds already establish that he (weakly) prefers A to B , and prefers A to no trade, at all possible valuations. Similarly, the seller passes RPAR without changing his bounds because the bounds establish that he weakly prefers no trade to selling any combination of A and B given the current prices. Thus, we have no activity even though the current provisional trade is inefficient.*

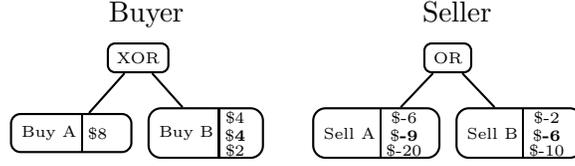


Figure 5: An Example to illustrate the failure of a simple RPAR rule. True values are shown in bold and are such that the efficient outcome is no trade.

The modified revealed-preference activity rule (δ -MRPAR) addresses this issue and also supports claims about approximate efficiency given approximate EQ prices. For this purpose, let $\theta_i^\pi(\lambda_i, \lambda'_i, v'_i) = v'_i(\lambda_i) - p^\pi(\lambda_i) - (v'_i(\lambda'_i) - p^\pi(\lambda'_i))$, i.e. the profit to i for trade λ_i over λ'_i given v'_i and prices π . We have:

Definition 5 Given provisional trade λ^α , linear prices π , and accuracy parameter $\delta \geq 0$, δ -MRPAR requires that every bidder i refines his value bounds so that his TBBL tree T_i satisfies:

$$\theta^\pi(\lambda_i^\alpha, \lambda'_i, v'_i) \geq -\delta, \quad \forall v'_i \in T_i, \forall \lambda'_i \in \mathcal{F}_i \quad (22)$$

or, that there is some $\check{\lambda}_i \in \mathcal{F}_i(x^0)$ such that

$$\theta^\pi(\check{\lambda}_i, \lambda'_i, v'_i) \geq 0, \quad \forall v'_i \in T_i, \forall \lambda'_i \in \mathcal{F}_i(x^0) \quad (23)$$

$$\theta^\pi(\check{\lambda}_i, \lambda_i^\alpha, v'_i) > \delta, \quad \forall v'_i \in T_i \quad (24)$$

This δ -MRPAR rule imposes constraints on bid tree T_i , this time given prices π and also the provisional trade λ_i^α . Phrasing to allow for the rule to be interpreted with and without the δ relaxation, each bidder must adjust his valuation bounds to establish that the provisional trade is [within δ of] maximizing profit for all possible valuations (Eq 22), or some other trade satisfies RPAR (Eq. 23) and is strictly preferred [by at least δ] to the provisional trade (Eq. 24). Just as for RPAR, one can show that a bidder can always meet δ -MRPAR (for any δ) by defining an exact valuation.¹¹

In the special case of $\delta = 0$, the δ -MRPAR rule simplifies, and constrains T_i given (π, λ_i^α) to require that there exists some $\check{\lambda}_i \in \mathcal{F}_i(x^0)$ such that:

$$\theta^\pi(\check{\lambda}_i, \lambda'_i, v'_i) \geq 0, \quad \forall v'_i \in T_i, \forall \lambda'_i \in \mathcal{F}_i(x^0) \quad (25)$$

$$\mathbf{and, either } \check{\lambda}_i = \lambda_i^\alpha \quad \mathbf{or, } \theta_i^\pi(\check{\lambda}_i, \lambda_i^\alpha, v'_i) > 0, \quad \forall v'_i \in T_i. \quad (26)$$

Eq. (25) is nothing more than RPAR, and the additional requirements enforce that the trade is either λ_i^α or strictly preferred to λ_i^α . The need to show a strict preference in Eq. (26) prevents the deadlock shown in Example 4. The seller has shown only a *weak* preference for not trading over selling A. MRPAR's requirement for strictness in the case that $\check{\lambda}_i \neq \lambda_i^\alpha$ will force the seller to show that he strictly prefers $\check{\lambda}_i$, in this case by reducing the upper-bounds on both A and B; thus MRPAR ensures progress.

11. Let v_i denote this valuation. If δ -MRPAR is not satisfied via Eq. (22) then $\check{\lambda}_i \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} [v_i(\lambda_i) - p^\pi(\lambda)]$ will satisfy δ -MRPAR. This satisfies Eq. (23) by construction. Now, let λ'_i denote the trade with $v_i(\lambda'_i) - p^\pi(\lambda'_i) > v_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) + \delta$. We have $v_i(\lambda_i) - p^\pi(\lambda_i) \geq v_i(\lambda'_i) - p^\pi(\lambda'_i) > v_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) + \delta$, and Eq. (24).

Theorem 4 *If prices π are δ -approximate EQ prices for provisional valuation profile v^α and trade λ^α , and every bidder i meets δ -MRPAR without precluding v_i^α from its updated bid tree, then λ^α is a $2 \min(M, \frac{n}{2})\delta$ -approximate efficient trade when bid trees contain true bidder valuations.*

Proof: Fix bidder i . By δ -EQ, we have $\theta^\pi(\lambda_i^\alpha, \lambda'_i, v_i^\alpha) \geq -\delta$ for all $\lambda'_i \in \mathcal{F}_i(x^0)$. Consider any $\tilde{\lambda}_i \neq \lambda_i^\alpha$. Because v_i^α remains in the bid tree, we must have $\theta^\pi(\tilde{\lambda}_i, \lambda_i^\alpha, v_i^\alpha) \leq \delta$ and δ -MRPAR cannot be satisfied via Eqs. (23) and (24). Therefore, δ -MRPAR is satisfied for every bidder via Eq. (22) and with provisional trade λ^α the satisfying trade. Therefore we prove that prices, π , are δ -approximate EQ prices for all valuations, and including the true valuation since bidders are straightforward and this is within their bounds. The efficiency of the trade follows from Corollary 1. \square

Let us suppose for the moment that δ -MRPAR is the only activity rule in ICE, and the exchange terminates as soon as prices are δ -accurate and v^α is retained in the bid tree by all bidders in meeting the activity rule, or when quiescence is reached and no bidder refines its bounds in meeting the rule. In this variation, provisional trade λ^α is the trade finally implemented. We have the following as an immediate corollary:

Corollary 2 *ICE with δ -MRPAR is $2 \min(M, \frac{n}{2})$ -efficient when prices are δ -accurate upon termination and bidders are straightforward.*

In particular, ICE has this efficiency property when prices are δ -accurate in every round.

Example 5 *To illustrate the δ -MRPAR rule consider a single bidder with a valuation tree as in Figure 6(a). Suppose the provisional trade λ_i^α allocates A to the bidder, and with prices $\pi_A = 3, \pi_B = 4$ and $\delta = 2$. Here the bidder has satisfied δ -MRPAR because the guaranteed $\$2-\$3=\$-1$ payoff from A is within δ of the possible $\$5-\$4=\$1$ payoff from B . Now consider Figure 6(b), with a relaxed upper-bound on “buy B ” of $\$8$. Now the bidder fails δ -MRPAR because the guaranteed $\$-1$ payoff from A is not within δ of the possible payoff from B of $\$8-\$4=\$4$. Let $[x, v]$ and $[w, y]$ denote the lower and upper bounds, on “buy A ” and “buy B ” respectively, as revised in meeting the rule. To pass the rule, the bidder has two choices:*

- *Demonstrate λ_i^α is the best response. To do so the bidder will need to adjust x and y to make $x - 3 \geq y - 4 - 2 \Rightarrow y - x \leq 3$; e.g. values $x = \$2, y = \5 solve this, as in Figure 6(a), as do many other possibilities.*
- *OR Demonstrate that another trade (e.g. “buy B ”) is more than $\$2$ better than λ_i^α , i.e., $w - 4 > v - 3 + 2 \Rightarrow w - v > 3$, and “buy B ” is weakly better than the null trade, i.e., $w - 4 \geq 0$. For instance, if the bidder’s true values are $v_A = \$3, v_B = \8 then $x \leq 3 \leq v$ and $w \leq 8 \leq y$ and the rule cannot be satisfied in the first case. But, the buyer can establish that “buy B ” is its best-response, e.g. by setting $v = \$4, w = \7 , or $v = \$3, w = \6 .*

With only δ -MRPAR, it is quite possible for ICE to get stuck, with all bidders satisfying the activity rule without changing their bounds but with the prices less than δ accurate. This shortcoming is addressed with the DIAR rule in Section 4.4.

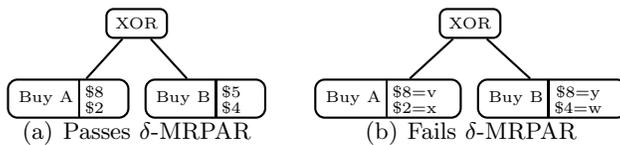


Figure 6: δ -MRPAR where the provisional trade is “Buy A”, $\pi_A = 3, \pi_B = 4$ and $\delta = 2$

Remark: Computation and Bidder Feedback. The definition of MRPAR naively suggests that checking for compliance requires explicitly considering all valuations $v'_i \in T_i$ and all trades $\lambda'_i \in \mathcal{F}_i(x^0)$. Fortunately, this is not necessary. We present in the Appendix a method to check MRPAR given prices π , provisional trade λ_i^α and bid tree T_i by solving three MIPs. Moreover, the solution to these MIPs also provides nice feedback for bidders. ICE can automatically identify a set of nodes at which a bidder needs to increase his lower bound and a set of nodes at which a bidder needs to decrease his upper bound in meeting MRPAR. This detail is also provided in the Appendix.

4.4 Delta Improvement Activity Rule (DIAR)

The second activity rule (DIAR) is introduced to ensure that ICE is always δ -efficient, even when δ -accurate EQ prices do not exist upon quiescence of δ -MRPAR. Thus, the role of DIAR is to handle price inaccuracies.

Given prices π and provisional trade λ_i^α , the main focus of DIAR is the following upper-bound $\bar{\delta}_i^k$, on the amount by which prices π might misprice some trade $\lambda_i^k \in \mathcal{F}_i(x^0)$ with respect to bidder i 's true valuation:

$$\bar{\delta}_i^k = \max_{v'_i \in T_i} [v'_i(\lambda_i^k) - p^\pi(\lambda_i^k) - (v'_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))]. \quad (27)$$

We refer to $\bar{\delta}_i^k$ on trade λ_i^k , which depends on prices π , bid tree T_i and provisional trade λ_i^α , as the **DIAR error**. The error bounds the true additional payoff that the bidder could achieve from trade λ_i^k over trade λ_i^α . If we order trades, $\lambda_i^1, \lambda_i^2, \dots$, so that λ_i^1 has maximal DIAR error, then $\bar{\delta}_i^1 \geq \delta_i$, where $\delta_i = \max_{\lambda'_i \in \mathcal{F}_i(x^0)} [v'_i(\lambda'_i) - p^\pi(\lambda'_i) - (v'_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))]$ is the pricing error with respect to the provisional trade and provisional valuation profile. This is the error that the pricing algorithm is designed to minimize in each round, and the same error that is used in Corollary 1 in reference to δ -accurate prices. Thus, we see that the maximal DIAR error also bounds the pricing error.

Loosely, DIAR requires that a bidder reduces the DIAR error $\bar{\delta}_i^j$, on some trade λ_i^j , prioritizing trades with large error, or proves that no improvement is possible (by providing exact value information where relevant in his bid tree.) Figure 7 illustrates the difference between MRPAR and DIAR. A bidder can satisfy MRPAR, by making it clear that the lower bound on payoff from some trade is greater than the upper bound on all other trades, while leaving a great amount of uncertainty about their maximal value on the preferred trade. DIAR requests that a bidder also refine this upper bound if it is leading to price inaccuracy. DIAR is parameterized by some $\epsilon \geq 0$. We refer to the formal rule as ϵ -DIAR:

Definition 6 *To satisfy ϵ -DIAR given provisional trade λ_i^α and prices π , then the bidder must modify its valuation bounds to establish some trade, $\lambda_i^j \in \mathcal{F}_i(x^0)$, for which*

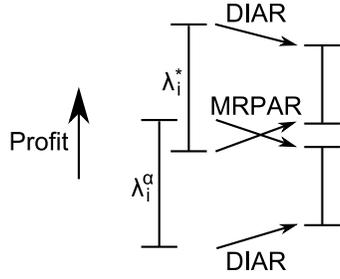


Figure 7: Stylized effect of MRPAR and DIAR on the bounds of the λ_i^α and λ_i^* trades

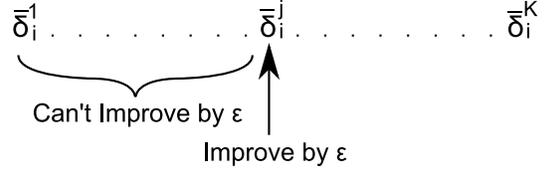


Figure 8: Trades for bidder i , ordered with DIAR error reducing from left to right. The bidder must reduce, by at least ϵ , the DIAR error on the trade with the greatest error for which this is possible and prove (via valuation bounds) that it is impossible to improve by ϵ any trades with larger error.

(a) the DIAR error on some trade λ_i^j has been reduced by at least ϵ , **and** (b) for all $1 \leq k < j$, trade $\bar{\delta}_i^k$ **cannot** be improved by ϵ

or, (c) if no such trade λ_i^j exists, then the bidder must establish that $\bar{\delta}_i^k$ cannot be improved by ϵ on **any** trade.

DIAR places constraints on the bid tree of a bidder given prices π , the provisional trade λ_i^α , and also in relation to the bid tree in the previous round. This context of the previous bid tree, and requiring an improvement in the DIAR error where possible, is what ensures progress. The rule is illustrated in Figure 8. In practice, we define the ϵ parameter to be large at the start and smaller towards the end. A large ϵ value leads to larger adjustments when an adjustment is possible while a small ϵ leads to slower, but more reliable, progress. We find that a simple rule:

$$\epsilon := \frac{\gamma}{n} \sum_i \sum_{\beta \in T_i} \frac{\bar{v}_{i \in N}(\beta) - v_i(\beta)}{|T_i|}, \quad (28)$$

for $\gamma \approx 0.5$ works well.

Example 6 Consider the tree in Figure 9(a) when the provisional trade is “buy A”, prices $\pi = (\$4, \$5, \$6)$ and DIAR parameter, $\epsilon = 1$. The DIAR error on each trade, defined via Eq. (27), and listed in decreasing order, are:

$$\begin{aligned} C \rightarrow \bar{\delta}^1 &= (\$10 - \$6) - (-\$2) = \$6 \\ B \rightarrow \bar{\delta}^2 &= (\$8 - \$5) - (-\$2) = \$5 \\ \emptyset \rightarrow \bar{\delta}^3 &= (\$0 - \$0) - (-\$2) = \$2 \\ A \rightarrow \bar{\delta}^4 &= (\$2 - \$4) - (-\$2) = \$0, \end{aligned}$$

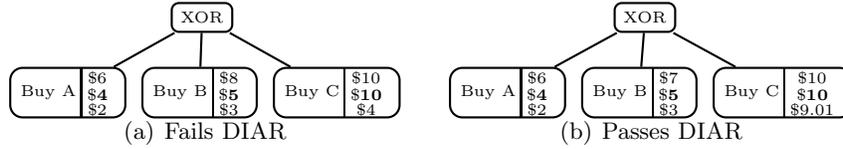


Figure 9: Respecting DIAR where the provisional trade is “Buy A”, $\pi_A = 4, \pi_B = 5, \pi_C = 6$ and $\epsilon = 1$

where $-\$2 = \$2 - \$4$ is the worst-case profit from the provisional trade. Now, we see that $\bar{\delta}^1$ can not be made smaller by lowering the upper-bound on leaf “buy C” because this bound is already tight against the truthful value of \$10. Instead the bidder must demonstrate that a decrease of $\epsilon = 1$ is impossible by raising the lower bound on “buy C” to 9.01. However $\bar{\delta}^2$ can be decreased by $\epsilon = 1$, by reducing the upper-bound on “buy B” from 8 to 7, giving us the tree in Figure 9(b).

Consider instance (v, x^0) . Say that a node $\beta \in T_i$ in the bid tree of bidder i is *interesting* when the node is satisfied in some feasible trade.

Theorem 5 *When ICE incorporates DIAR, then a straightforward bidder must eventually reveal complete value information on all interesting nodes in his bid tree as $\epsilon \rightarrow 0$.*

Proof: Fix provisional trade λ_i^α and consider trade, $\lambda_i^1 \in \mathcal{F}_i(x^0) \neq \lambda_i^\alpha$, with the maximal DIAR error. Continue to assume straightforward bidders. Recall that $v_i(\beta)$ denotes a bidder’s true value on node β in his *TBBL* tree. By case analysis on nodes $\beta \in T_i$, meeting the DIAR rule on this trade as $\epsilon \rightarrow 0$, requires:

- (i) Nodes $\beta \in \lambda_i^1 \setminus \lambda_i^\alpha$. Decrease the upper-bound to $v_i(\beta)$, the true value, to reduce the error. Increase the lower-bound to $v_i(\beta)$ to prove that further progress is not possible.
- (ii) Nodes $\beta \in \lambda_i^\alpha \setminus \lambda_i^1$. Increase the lower-bound to $v_i(\beta)$, the true value, to reduce the error. Decrease the upper-bound to $v_i(\beta)$ to prove that further progress is not possible.
- (iii) Nodes $\beta \in \lambda_i^\alpha \cap \lambda_i^1$. No change is required.
- (iv) Nodes $\beta \notin \lambda_i^1 \cup \lambda_i^\alpha$. No change is required.

Continue to fix some λ_i^α , and consider now the impact of DIAR as $\epsilon \rightarrow 0$ and as the rule is met for successive trades, moving from λ_i^1 to λ_i^2 and onwards. Eventually, the value bounds on all nodes $\beta \notin \lambda_i^\alpha$ but *in at least one other feasible trade* are driven to truth by (i), and the value bounds on all nodes $\beta \in \lambda_i^\alpha$ but *not in at least one other feasible trade* are driven to truth by (ii). Noting that the null trade is always feasible, the bidder will ultimately reveal complete value information except on nodes that are not satisfied in any feasible trade. \square

DIAR is ultimately coupled within ICE with MRPAR, and combined with methods to prove approximate efficiency, one leveraging approximate EQ prices and another direct method based on lower and upper valuation bounds. Thus, the DIAR rules does not imply

that bidders will reveal full information. Rather, the presence of DIAR ensures both good performance in practice as well as good theoretical properties.

In an interesting variation, DIAR could be used only in rounds in which the price error for the provisional valuation and trade is greater than the error associated with δ -MRPAR. This is because δ -MRPAR is sufficient for approximate efficiency when prices are accurate enough. In our experiments we enable DIAR in all rounds of ICE, and it fires in parallel with δ -MRPAR. In practice, we see that most of the progress in refining valuation information occurs due to MRPAR, and that *all* the progress in early rounds occurs due to MRPAR. Experimental support for this is provided in Section 8.

Remark: Computation and Bidder Feedback. We present in the Appendix a method to check ϵ -DIAR given prices π , provisional trade λ_i^α , the bidder’s bid tree from the past round and proposed new bid tree by solving two MIPs. Moreover, the solution to these MIPs also provides nice feedback for bidders. ICE can automatically identify the trade – and in turn the corresponding nodes in the bid tree – for which the bidder must provide more information.

4.5 Bounding Efficiency, Last-and-Final and Termination

Let t denote the current round in ICE. The round starts with the announcement of prices, denote them π^t , and the provisional trade. The round ends with every bidder having met the δ -MRPAR and ϵ -DIAR activity rules.

At the closing of each round, ICE makes a determination about whether to move to the last-and-final round. Bidders are notified when this occurs. The last-and-final round provides a final opportunity for bidders to update their lower valuation bound information (without exceeding their upper bounds). The exchange finally terminates with the efficient trade and Threshold payments, as defined with respect these lower bounds.

The transition into the last-and-final round occurs when the trade that is optimal given the revised lower bound valuations, $\underline{\lambda} \in \arg \max_{\lambda \in \mathcal{F}(x^0)} \sum_i v_i(\lambda_i)$, is within a *target approximation error* $\Delta^* \in (0, 1]$ of the efficient trade, i.e.

$$\text{EFF}(\underline{\lambda}) = \frac{\sum_i v_i(\underline{\lambda}_i)}{\sum_i v_i(\lambda_i^*)} = \frac{v(\underline{\lambda})}{v(\lambda^*)} \geq \Delta^* \quad (29)$$

Of course, the true valuation v_i of bidder i is not known by the exchange! However, we can provide two methods to bound the efficiency of the pessimistic trade. The first is price-based and uses duality theory. The second is determined by direct reasoning about the bounds on bidder valuations.

A price-based proof of error. We have already seen that this bound can sometimes be established via prices. Fix some $\delta \geq 0$. For v^α denoting the provisional valuation profile at the start of round t , and λ^α the corresponding provisional trade, we know that if,

- (a) bidders meet δ -MRPAR while leaving v^α within their bounds,
- (b) prices π^t were δ -approximate EQ prices for v^α and λ^α , and
- (c) λ^α is equal to $\underline{\lambda}$, i.e. the efficient trade given the refined lower bound valuations,

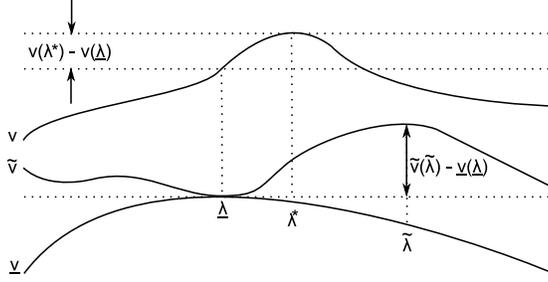


Figure 10: Determining an efficiency bound based on valuation bounds.

then trade $\underline{\lambda}$ is a $2 \min(M, \frac{n}{2})\delta$ -approximation to the efficient trade λ^* by Corollary 1. We have $\sum_i v_i(\underline{\lambda}_i) + 2 \min(M, \frac{n}{2})\delta \geq \sum_i v_i(\lambda_i^*)$, and then,

$$\text{EFF}(\underline{\lambda}) = \frac{\sum_i v_i(\underline{\lambda}_i)}{\sum_i v_i(\lambda_i^*)} \geq 1 - \frac{2 \min(M, \frac{n}{2})\delta}{\sum_i v_i(\lambda_i^*)} \geq 1 - \frac{2 \min(M, \frac{n}{2})}{\max_{\lambda \in \mathcal{F}(x^0)} \sum_i \bar{v}_i(\lambda)} \delta, \quad (30)$$

which we define as ω^{price} , and can be compared in value with the target approximation error, Δ^* . Conditioned on (a-c) being met, so that the bound is available, it will satisfy $\omega^{\text{price}} \geq \Delta^*$ for a small enough δ parameter. When the bound is not available we set $\omega^{\text{price}} := 0$.

A direct proof of error. We also provide a complementary, direct, method to establish a bound on the error of $\underline{\lambda}$ by working with the refined valuation bounds upon the closing of round t . Given the valuation bounds, we can establish the following bound,

$$\text{EFF}(\underline{\lambda}) = \frac{v(\underline{\lambda})}{v(\lambda^*)} \geq \min_{v' \in T, \lambda' \in \mathcal{F}(x^0)} \left[\frac{v'(\underline{\lambda})}{v'(\lambda')} \right] = \min_{\lambda' \in \mathcal{F}(x^0)} \left[\frac{\tilde{v}(\underline{\lambda})}{\tilde{v}(\lambda')} \right] = \frac{v(\underline{\lambda})}{\tilde{v}(\tilde{\lambda})}, \quad (31)$$

which we define as ω^{direct} , and can be compared with the target approximation error, Δ^* . Notation $\tilde{v} = (\tilde{v}_1, \dots, \tilde{v}_n)$, where \tilde{v}_i is the *perturbed valuation with respect to trade $\underline{\lambda}_i$* , as defined in Section 4.2. Trade $\tilde{\lambda}$ is the trade that maximizes $\sum_i \tilde{v}_i(\lambda_i)$ across all feasible trades. The first inequality holds because the domain of the minimization includes $v \in T$ and trade $\lambda' = \lambda^*$. The first equality holds because for any $\lambda' \neq \underline{\lambda}$, the worst-case efficiency for $\underline{\lambda}$ occurs when the value $v' \in T$ is selected to minimize the value on nodes $\underline{\lambda} \setminus \lambda'$, maximize the value on nodes $\lambda' \setminus \underline{\lambda}$, and minimize the value on shared nodes, $\lambda' \cap \underline{\lambda}$. Whatever the choice of λ' , this valuation is provided through perturbed valuation \tilde{v} . For the final equality, $\tilde{v}(\underline{\lambda}) = v(\underline{\lambda})$ by definition, and the optimal trade λ' is that which maximizes the value of the denominator, i.e. trade $\tilde{\lambda}$. Figure 10 schematically illustrates the various trades and values used in this bound, and in particular provides some graphical intuition for why $\tilde{v}(\tilde{\lambda}) - v(\underline{\lambda}) \geq \tilde{v}(\lambda^*) - v(\underline{\lambda}) = \max_{v' \in T} [v'(\lambda^*) - v'(\underline{\lambda})] \geq v(\lambda^*) - v(\underline{\lambda})$.

Moving to last-and-final. Given the above methods we can establish an approximation error $\omega^{\text{eff}} = \max(\omega^{\text{price}}, \omega^{\text{direct}})$. The rules of ICE state that the exchange moves to the last and final round, for target approximation error Δ^* , when either of the following hold:

- (a) error bound $\omega^{\text{eff}} \geq \Delta^*$
- (b) there is no trade even at optimistic valuations.

Combining this with Theorem 5, we immediately get our main result.

Theorem 6 *When ICE incorporates (approximate) MRPAR and ϵ -DIAR, then the exchange terminates with a trade that is within target approximation error Δ^* as $\epsilon \rightarrow 0$ and for straightforward bidders.*

Empirical support for the quality of the price-based bound and the direct efficiency bounds is provided in Section 8.

Setting provisional valuations. The bound $\omega^{\text{eff}} \in [0, 1]$ on efficiency is also used to define the α parameter, which is used in turn to define the provisional valuation v^α and to determine the provisional trade, and price feedback, to provide at the start of the next round. Recall that as $\alpha \rightarrow 1$, then $v^\alpha \rightarrow \underline{v}$. A simple approach that works well is to define,

$$\alpha := \max(0.5, \omega^{\text{eff}}). \quad (32)$$

Lower bound 0.5 is a useful heuristic for early rounds when ω^{eff} is likely to be small, making ICE adopt a provisional valuation in the middle of the valuation bounds when not much is known.

Remark. A simple variation on ICE could be defined to require progress in terms of the accuracy of *payments* before moving to last-and-final, with a bound on the payment accuracy computed in an analogous way to that on efficiency. Whether this is required in practice is likely domain-specific and to depend on a number of factors, including: (a) whether the payments tend to be accurate anyway, by the time the trade is approximately accurate; and (b) the impact that this design decision has on strategic behavior.

5. Step Three: Linear Price Feedback

In this section, we present the method used in ICE to compute linear, approximate competitive equilibrium prices in each round. The pricing step is the most computationally intensive of all steps in ICE and the computation is heavily optimized. We first define the prices via exponentially-sized linear programs (LPs). Constraint generation is used to find solutions to these otherwise intractable LPs, a technique that relies upon a restricted form of the winner-determination problem to identify the new constraints to introduce.

We compute prices π at the end of some round in ICE according to the following rules:

- I: Accuracy.** First, we compute prices that minimize the maximal error in the best-response constraints across all bidders.
- II: Fairness.** Second, we break ties to prefer prices that minimize the maximal deviation from Threshold payments across all bidders.
- III: Balance.** Third, we break ties to prefer prices that minimize the maximal price across all items.

Taken together, these steps are designed to promote the informativeness of prices in driving progress across rounds. By fairness, we prefer prices that better reflect the Threshold payments that are finally implemented at termination. Balance is well motivated in domains where items are more likely to be similar in value than dissimilar, preferring prices to be similar across items and rejecting extremal prices.

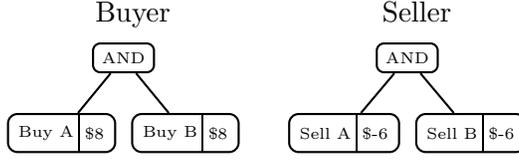


Figure 11: A simple example to illustrate pricing. ACC prices AB between \$12 and \$16, FAIR narrows this to \$14 and BAL requires $A = \$7, B = \7

Example 7 Consider the example in Figure 11 with one buyer interested in buying AB and one seller interested in selling AB . Here the buyer's and seller's values for each item are 8 and -6 respectively. The efficient outcome given these values is for the trade to complete. ACC requires $12 \leq \pi_A + \pi_B \leq 16$, and thus allows a range of prices. The Threshold payment splits the difference, so that the buyer pays 14 to the seller and so FAIR adds the constraint $\pi_A + \pi_B = 14$. Finally, BAL requires $\pi_A = \pi_B = 7$.

5.1 The Three Pricing Stages and Lexicographical Refinement

Leaving the idea of lexicographical minimization to one side for a moment, we first define the objectives of each of the three stages, and provide an example. Given provisional trade, λ^α and valuation profile, v^α , we define maximally accurate EQ prices as those that solve the following LP:

$$\begin{aligned}
 \delta_{\text{acc}}^* &= \min_{\pi, \delta_{\text{acc}}} \delta_{\text{acc}} && \text{[ACC]} \\
 \text{s.t. } & v_i^\alpha(\lambda'_i) - \sum_j \pi_j \lambda'_{ij} \leq v_i^\alpha(\lambda_i^\alpha) - \sum_j \pi_j \lambda_{ij}^\alpha + \delta_{\text{acc}}, \quad \forall i, \forall \lambda'_i \in \mathcal{F}_i(x^0) && (33) \\
 & \delta_{\text{acc}} \geq 0, \quad \pi_j \geq 0, \forall j \in G.
 \end{aligned}$$

These prices minimize the worst-case short fall in payoff for trade λ^α across all bidders, i.e. minimize the maximal value of $\theta^\pi(\lambda_i^*, \lambda_i^\alpha, v_i^\alpha)$, where $\lambda_i^* = \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} [v_i^\alpha(\lambda_i) - p^\pi(\lambda_i)]$. Second, we break the remaining ties to prefer *fair* prices: choosing prices that minimize the worst-case error in utility for each bidder with respect to the utility that would be achieved given Threshold payments at provisional valuation profile v^α . The fairness tie-breaking method is formulated as the following LP:

$$\begin{aligned}
 \delta_{\text{fair}}^* &= \min_{\pi, \delta_{\text{fair}}} \delta_{\text{fair}} && \text{[FAIR]} \\
 \text{s.t. } & v_i^\alpha(\lambda'_i) - \sum_j \pi_j \lambda'_{ij} \leq v_i^\alpha(\lambda_i^\alpha) - \sum_j \pi_j \lambda_{ij}^\alpha + \delta_{\text{acc}}^*, \quad \forall i, \forall \lambda'_i \in \mathcal{F}_i(x^0) && (34)
 \end{aligned}$$

$$\delta_{\text{fair}} \geq U_{\text{vcg}, i} - \sum_j \pi_j \lambda_{ij}^\alpha, \quad \forall i \quad (35)$$

$$\delta_{\text{fair}} \geq \sum_j \pi_j \lambda_{ij}^\alpha - U_{\text{vcg}, i}, \quad \forall i \quad (36)$$

$$\delta_{\text{fair}} \geq 0, \quad \pi_j \geq 0, \quad \forall j \in G,$$

where $U_{\text{vcg},i}$ denotes the utility to bidder i in the VCG outcome given valuations v_i^α and trade λ_i^α . The objective here is the same as in the Threshold payment rule: minimize the maximal error between bidder payoff (at v^α) for the provisional trade and the VCG payoff (at v^α). Rather than compute the Threshold payments, and then the utility, and then define prices to best approximate the Threshold payments this is handled directly through this VCG-based formulation.

Third, we break the remaining ties to prefer *balanced* prices: choosing prices that minimize the maximal price across all items. For this, we solve a lexicographic sequence of LPs, alternating between minimization and maximization of all prices not yet pinned down. For instance, the minimization LP is formulated as follows (the maximization LP is an exact mirror and omitted in the interest of space):

$$\begin{aligned} & \min_{\pi, \delta_{\text{bal}}} \delta_{\text{bal}} && \text{[BAL]} \\ \text{s.t.} \quad & v_i^\alpha(\lambda'_i) - \sum_j \pi_j \lambda'_{ij} \leq v_i^\alpha(\lambda_i^\alpha) - \sum_j \pi_j \lambda_{ij}^\alpha + \delta_{\text{acc}}^*, \quad \forall i, \forall \lambda' \in \mathcal{F}(x^0) \end{aligned} \quad (37)$$

$$\delta_{\text{fair}}^* \geq U_{\text{vcg},i} - \sum_j \pi_j \lambda_{ij}^\alpha, \quad \forall i \quad (38)$$

$$\delta_{\text{fair}}^* \geq \sum_j \pi_j \lambda_{ij}^\alpha - U_{\text{vcg},i}, \quad \forall i \quad (39)$$

$$\delta_{\text{bal}} \geq \pi_j, \quad \forall j \quad (40)$$

$$\pi_j \leq \min \max \delta_{\text{bal}}^* + \epsilon, \quad \forall j \in G \quad (41)$$

$$\pi_j \geq \max \min \delta_{\text{bal}}^*, \quad \forall j \quad (42)$$

$$\delta_{\text{bal}} \geq 0, \pi_j \geq 0, \quad \forall j \in G,$$

Here, $\min \max \delta_{\text{bal}}^*$ and $\max \min \delta_{\text{bal}}^*$ represent the minimum δ_{bal} value and the maximum δ_{bal} value from all previous BAL maximization and minimization MIPs respectively. These two constraints serve to bind the prices into a progressively narrowing range that tends towards the center.

5.1.1 LEXICOGRAPHICAL REFINEMENT

For all three pricing stages, we also perform lexicographical refinement (with respect to bidders in ACC and FAIR, and with respect to goods in BAL). In addition to further improving the quality of the prices this also ensures uniqueness. For instance, in ACC we successively minimize the maximal error across all bidders. Given an initial solution we first “pin down” the error on all bidders for whom constraint (44) is binding. For such a bidder i , the constraint is replaced with

$$v_i^\alpha(\lambda'_i) - \sum_j \pi_j \lambda'_{ij} \leq v_i^\alpha(\lambda_i^\alpha) - \sum_j \pi_j \lambda_{ij}^\alpha + \delta_{\text{acc},i}^*, \quad \forall \lambda'_i \in \mathcal{F}(x^0), \quad (43)$$

and ACC is resolved with variable δ_{acc} included in Constraints (44) for bidders not yet pinned-down, so that further progress is made in lexicographically minimizing the maximal error. Eventually, the sequence of ACC LPs terminate with a best-case error $\delta_{\text{acc},i}^*$ defined for each bidder.

Consequently, ACC passes a *vector*, $\delta_{\text{acc}}^* = (\delta_{\text{acc},1}^*, \dots, \delta_{\text{acc},n}^*)$, to FAIR, rather than a single value. Constraints (34) in FAIR are replaced with an individualized constraint, reflecting $\delta_{\text{acc},i}^*$, for each bidder. A similar lexicographical optimization process is then used for FAIR, with constraints (35) and (36) replaced with $\delta_{\text{fair},i}^* \geq U_{\text{vcg},i} - \sum_j \pi_j \lambda_{ij}^\alpha$ and $\delta_{\text{fair},i}^* \leq \sum_j \pi_j \lambda_{ij}^\alpha - U_{\text{vcg},i}$ as bidders i are “pinned down” and the payment error $\delta_{\text{fair},i}^*$ is lexicographically minimized for each bidder.

Upon termination of FAIR, the vector, $\delta_{\text{acc}}^* = (\delta_{\text{acc},1}^*, \dots, \delta_{\text{acc},n}^*)$, is passed together with $\delta_{\text{fair}}^* = (\delta_{\text{fair},1}^*, \dots, \delta_{\text{fair},n}^*)$ from FAIR, into BAL. In addition to constraints (37), constraints (38) and (39) are also individualized in BAL, to capture the values $\delta_{\text{fair},i}^*$ for each bidder. Consequently when BAL proceeds as described above, the quoted prices $\pi^* = (\pi_1^*, \dots, \pi_m^*)$ are lexicographically determined for all goods G . Finally, the price error given provisional valuation profile v^α and trade λ^α is defined as $\delta := \max_i \delta_{\text{acc},i}^*$.

5.2 Computation: Constraint Generation and Optimizations

Having defined the prices in ICE we now explain how they are computed through constraint generation. We also mention the use of heuristic seeding, explain how constraints are pooled across rounds, and also mention the use of provisional locking and lazy constraint checks to speed up computation.

Problems ACC, FAIR and BAL all have an exponential number of constraints because the price accuracy constraints (33) are defined for all trades $\lambda' \in \mathcal{F}(x^0)$ and all bidders i . It is infeasible to even write this problem down. Rather than solve it explicitly we use *constraint generation* (Bertsimas & Tsitsiklis, 1997, e.g.) and dynamically generate a sufficient subset of constraints. Constraint generation (CG) considers a relaxed program that only contains a manageable subset of the constraints, and solves this to optimality. Given a solution to this relaxed program, a subproblem is used to either prove that the solution is optimal to the full program, or find a “violated constraint” in the full problem that is then introduced and the (now strengthened) relaxed program resolved.

We illustrate this process for ACC. Let \mathbb{F}_i denote a manageable subset of all possible feasible trades to bidder i . Then, a relaxed version (called *ACC'*) is formulated by substituting constraints (33) with:

$$v_i^\alpha(\lambda'_i) - \sum_j \pi_j \lambda'_{ij} \leq v_i^\alpha(\lambda_i^\alpha) - \sum_j \pi_j \lambda_{ij}^\alpha + \delta_{\text{acc}}, \quad \forall i, \forall \lambda'_i \in \mathbb{F}_i, \quad (44)$$

where \mathbb{F}_i is a set of trades that are feasible for bidder i given the other bids. We shortly explain how a new violated constraint, or proof of correctness, is generated. Note that we use the same constraint sets across all three stages of pricing, augmenting this one pool of constraints as necessary when solving each stage. We also maintain the constraints across rounds of the exchange. This control flow is illustrated in Figure 12.

How can we identify violated constraints? Let π^* denote the solution to relaxed problem, *ACC'*. We can solve n subproblems, one for each bidder, of form:

$$\max_{\lambda'_i \in \mathcal{F}(x^0)} v_i^\alpha(\lambda'_i) - \sum_j \pi_j^* \lambda'_{ij}, \quad [\text{R-WD}(i)]$$

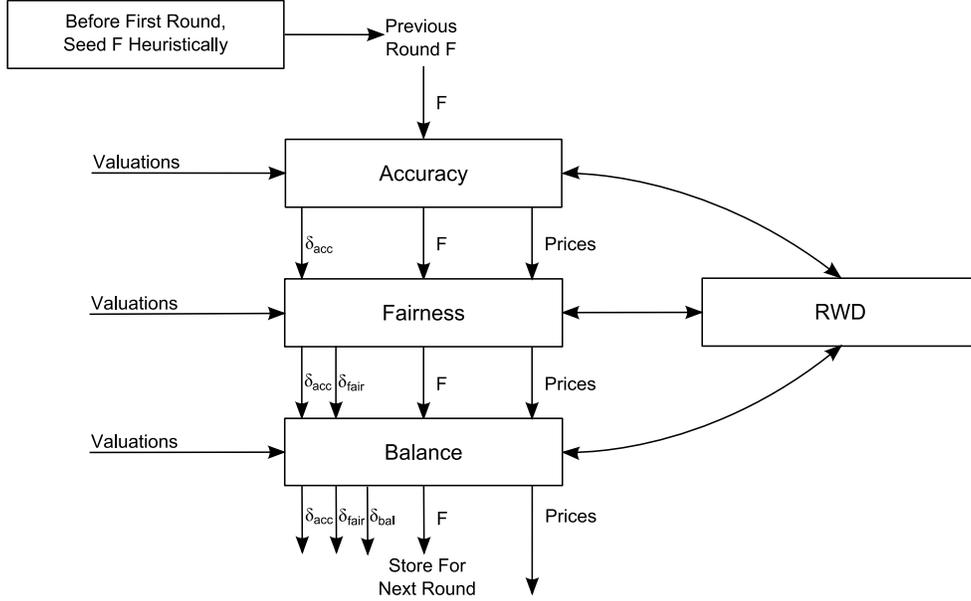


Figure 12: Using constraint generation to calculate prices. The constraint set (indicated F in the figure) is maintained across rounds and propagated across the stages. A restricted WD problem (RWD) checks for violated constraints.

to check whether solution (π^*, δ_{acc}^*) to ACC' is also a feasible solution to ACC. In R-WD(i), the “restricted winner-determination problem for bidder i ”, the objective is to determine a most preferred trade for bidder i at candidate prices π^* .

Let $\check{\lambda}_i$ denote a solution to R-WD(i). We check condition:

$$v_i^\alpha(\check{\lambda}_i) - \sum_j \pi_j^* \check{\lambda}_{ij} \leq v_i^\alpha(\lambda_i^\alpha) - \sum_j \pi_j^* \lambda_{ij}^\alpha + \delta_{acc}^*, \quad (45)$$

and if this condition holds for all bidders i , then solution (π^*, δ_{acc}^*) is optimal for the full program ACC. Otherwise, trade $\check{\lambda}_i$ is added to constraint set, \mathbb{F}_i , for the bidders i for which constraint (45) is violated and we re-solve ACC' with the new set of constraints.

Problem R-WD(i) is a specialization of the WD problem, in which the objective is to maximize the payoff of a single bidder, rather than the total value across all bidders. It is solved as a MIP, by rewriting the objective in WD(T, x^0) as $\max\{v_i(\beta) \cdot sat_i(\beta) - \sum_j p_j^* \cdot \lambda_{ij}\}$ for bidder i . Thus, the structure of *TBBL* is exploited in generating new constraints, because this subproblem can be solved as a concise MIP. The trees submitted by bidders $i' \neq i$ are used to define the space of feasible trades.

We now mention the use of heuristic seeding, the streamlining of computation, and some aspects of providing numerical stability.

Heuristic seeding of constraint sets. By maintaining the set of constraints generated in earlier rounds we place most of the computational cost on computing prices in the very first round of ICE, when the constraint sets, \mathbb{F} , are empty. To speed-up computation we use a heuristic method to seed the constraint sets. We heuristically guess what the prices should

be, and then solve a number of R-WD(i) subproblems to back out appropriate trades to add to \mathbb{F}_i . There are many possible ways to heuristically guess appropriate prices, including many methods that are domain specific. For now, we use a heuristic that sets the price on a good to the average contribution to the value of the provisional trade for each good traded, subject to several simplifying assumptions regarding what else is activated in the trees.

Streamlining of computation. Within a given pricing stage (i.e., ACC, FAIR or BAL) we must perform CG together with lexicographical refinement. We have streamlined this computation. There are two key features of our approach:

- *Provisional Locking.* There are two types of mathematical programs we solve in pricing: LPs that solve for prices and IPs that generate additional constraints. The LPs can be solved significantly faster than the IPs, but several IPs can be run in parallel because they optimize over a single agent rather than globally. As a consequence, rather than performing our lexicographic ordering in each stage by locking one value at a time, we greedily run a sequence of LPs without CG, provisionally locking down at least $m_{\text{lock}} \geq 1$ values (e.g., δ_{acc}^* values in ACC, for a sequence of agents), before checking for violating constraints. The best choice for m_{lock} depends on the relative speed with which the LPs and IPs can be solved. In our experiments, we have found that choosing m_{lock} to be aggressive (i.e. close to 100% of the number of values), so that nearly all decisions are made before checking the constraints, works well.

- *Lazy Constraint Checks.* When performing CG, we choose not to check the validity of current prices for every bidder every time. Instead, once a bidder i has been found to pass, we optimistically assume that the error δ_i computed for that bidder remains valid, only running the expensive IP check for the bidders which continue to fail. Only once there are no such failing bidders, do we perform an additional check of *all* bidders to ensure that the prices have remained stable enough for the validity of the skipped bidders to have been maintained. If this check fails for any bidder then CG must continue. Eventually, when all m_{lock} bidders pass simultaneously, we make the m_{lock} provisional locks permanent (e.g. locking down the target payoff errors for m_{lock} bidders), and continue on for the next m_{lock} provisional locks in the lexicographic refinement.

Issues with numerical stability. Careful tuning is required to ensure that appropriate “ ϵ -constants” are included to relax the various constraints appropriately and in a way that ensures feasibility, without undermining necessary preconditions for future computation. Because of the repeated optimizations performed, it is easy for what seems to be a relaxation in one context to actually cause an infeasibility in a later context by permitting a solution that should otherwise be eliminated. For example, when pinning-down bidders for the purpose of lexicographical refinement we relax the associated bidder-constraints with a small $\epsilon_{\text{num}} > 0$.

6. Extended Illustrative Examples

In this section illustrate the behavior of the exchange on simple examples. These examples are provided to give a qualitative feel for its behavior. To construct the examples we

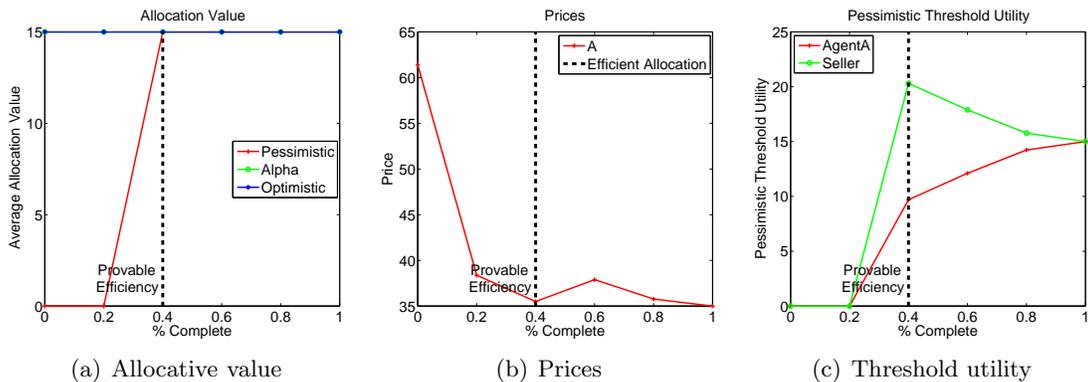


Figure 13: Seller: -\$20, Buyer: \$50

populate ICE with very simple, automated bidding agents. A detailed explanation of the operation of these bidding agents is provided in the Appendix.

In both this section, and also in presenting our main experimental results, we do *not* move to a last-and-final round. Rather, the bidding agents are programmed to continue to improve their bids past the round at which efficiency is already proved – when a last-and-final round would ordinarily be declared – and until payments are within some desired accuracy tolerance. We do this to avoid the need to program agents with a strategy for how to bid in the last-and-final round.

First we consider an example with a single seller with a value of -\$20 for a sale and a single buyer with a value of \$50 for obtaining the item. In Figure 13(a) we can see the truthful value of the pessimistic trade in the various rounds of the exchange. Initially no trade occurs, resulting in no net value for either bidder. Then, in the 3rd round enough information has been revealed to the exchange for the trade to switch to the efficient trade. We indicate the first round where the pessimistic trade can be proved to be 95% efficient by a dotted vertical line in the plots.

As the exchange progresses, the linear prices are calculated so as to explain the current α -trade, as shown in Figure 13(b). These prices converge towards the optimal \$35 over time. In Figure 13(c) we see the Threshold utility to each agent at the pessimistic trade, were the exchange to have stopped in a given round. The graph shows that at the point where the exchange has determined the efficient trade it still does not have enough information to calculate accurate Threshold payments (and thus utilities). However with a few additional rounds, we can see the Threshold utilities converge to \$15, an even split of the \$30 of surplus in this example.

For the next three examples, we assume a no-reserve seller of two items **A** and **B**. First we consider the case where there are three buyers. AgentA demands **A** with a value of \$8, AgentB demands **B** with a value of \$8, and AgentAB demands **A AND B** with a value of \$10. Figure 14(a) shows that very quickly the exchange discovers the correct trade. A price between \$5 and \$8 will be accurate in this situation, and we can see that the prices in Figure 14(b) meet this condition. Fairness drives the prices towards \$6, which will be the eventual threshold payments to AgentA and AgentB. And Balance ensures that the prices remain the same for the two items. In Figure 14(c) we see the Threshold payments

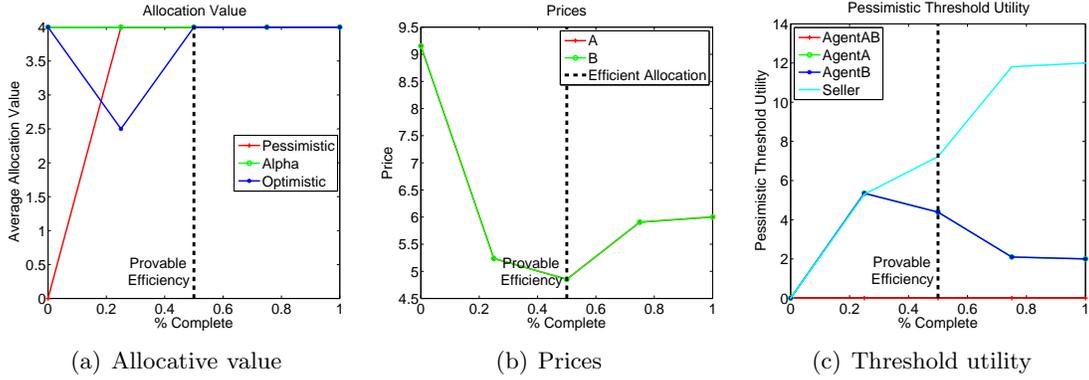


Figure 14: AgentA: **A** \$8, AgentB: **B** \$8, AgentAB: **A AND B** \$10.

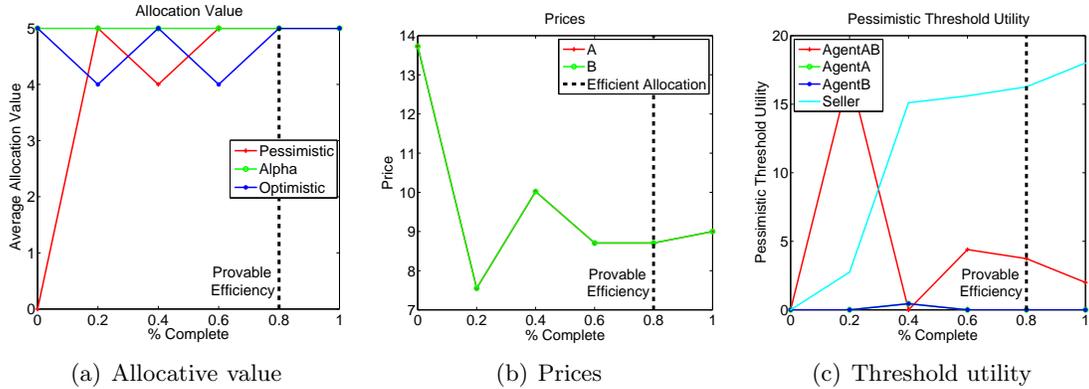


Figure 15: AgentA: **A** \$8, AgentB: **B** \$8, AgentAB: **A AND B** \$20.

eventually do reach \$6 causing AgentA and AgentB to have Threshold utilities of \$2, and the Seller to obtain a utility of \$12.

Next we consider an identical case, but where AgentAB has a value of \$20 instead. Now the goods will be allocated to this bidder instead of the other two. Consider the truthful value of the pessimistic trade through the rounds of the exchange shown in Figure 15(a). Here we can see that the exchange guess the correct trade in the second round (though it can't yet prove it), but further revelation fools it into thinking AgentB should be allocated **B** in round three. By the fourth round the correct trade has been reacquired. The prices in Figure 15(b) track the current α -trade and converge to a price of \$9, the eventual threshold payment for each item. Turning to Figure 15(c), we can see this payment corresponds to a Threshold utility of \$18 for the seller, \$2 for AgentAB and \$0 for AgentA and AgentB.

Next we consider the case of XOR valuations. Specifically, AgentAOverB will value **A** at \$10 XOR **B** at \$8. AgentBOverA will value **A** at \$8 XOR **B** at \$10. And a third buyer, AgentLowAB will value **A** at \$6 XOR **B** at \$6. Figure 16(a) shows that the exchange finds the correct trade (**A** to AgentAOverB, and **B** to AgentBOverA) in the third round. However the incorrect α -trade of **B** to AgentAOverB (The symmetry is broken randomly) in the second round causes prices to be biased towards that trade. It takes several rounds before enough information is revealed to generate Fair prices, near \$8.66.

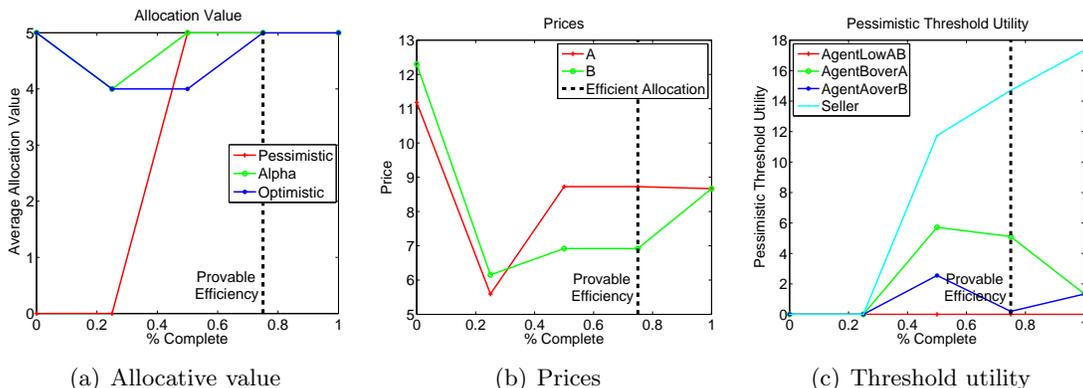


Figure 16: AgentAOverB: **A** \$10 XOR **B** \$8, AgentBOverA: **A** \$8 XOR **B** \$10, AgentLowAB **A** \$6 XOR **B** \$6

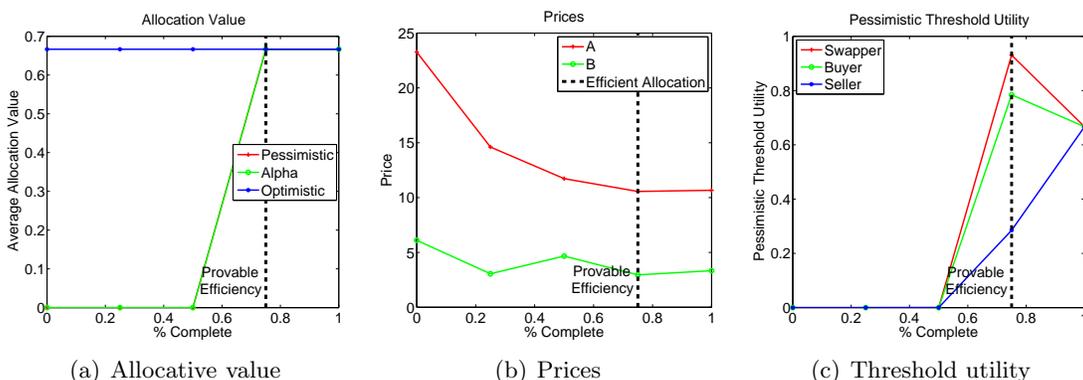


Figure 17: Seller **A** -\$10, Swapper: swap **A** for **B** \$8, Buyer **B** \$4

Lastly, we show a simple swapping example. Here we have a Seller offering **A** for a reserve of \$10, a Swapper who is willing to pay \$8 if he can swap **A** for his **B**, and a Buyer willing to pay \$4 for **B**. In this more complex example, it takes 4 rounds, as illustrated in Figure 17(a), for a trade to be found in the pessimistic trade. Revelation drives progress towards a completed trade, and as we can see in Figure 17(b), this is reflected in falling prices on the goods. Thus we can see that the price feedback is providing accurate information to the participants: only when the price eventually becomes low enough do the buying bidders actually want a trade to occur – and that is also when the exchange’s provisional trade switches. It is also worth noting that the greater valuations the Seller and Swapper place on good **A** result in a net higher price than that for good **B**. Figure 17(c) shows convergence to a Threshold utility of \$0.66 for all three agents, as they will equally split a surplus of \$2.

7. Systems Infrastructure

ICE is approximately 20,000 lines of extremely tight Java code, broken up into the functional packages described in Table 1.¹² The prototype is modular so that researchers may easily

12. Code size is measured in physical source line of code (SLOC)

Component	Purpose	Lines
Agent	Strategic behavior and information revelation decisions	2001
Model	XML support to load goods and true valuations	1353
Bidding Language	Implements <i>TBBL</i>	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

Table 1: Exchange components and code breakdown

replace components for experimentation.¹³ Because ICE’s complexity, it is essential that the code be constructed in a rigid hierarchy that avoids obscuring the high level logic behind the details of generating, running and integrating the results of MIPs. To this end, the system is written in a series of progressively more abstract “mini-languages” each of which defines a clean, understandable API to the next higher level of logic.

The top most level of this hierarchy is the Framework which serves to tie subordinate modules together. Chief among these modules is the “Driver” which defines the exchange flow, and which in turn relies on various “engine” modules to dictate the non-optimizer specific logic for the various functions of the system, such as WD, Pricing, Activity and Closing Rules etc. These engines drive the appropriate MIP/LP “builders” which maintain the state necessary to create MIPs based on our own specification language. This specification is fed into our generalized back-end optimization solver interface¹⁴ (we currently support CPLEX and the LGPL- licensed LPSolve), that handles machine load-balancing and a parallel MIP/LP solving. This concurrent solving capability is essential, as we need to handle tens of thousands of comparatively simple MIPs/LPs.

Because we have to build, run and interpret so many MIPs/LPs, cleanly and efficiently providing for their “code generation” is an imperative. Indeed, MIP “code generation” is so important to the system that much of the code is part of the Engine / Builder / Specification / Solver hierarchy. This hierarchy provides a modular, structured and domain-aware way to assemble MIPs, enabling the reuse of MIP generation code (and therefore Java code). But perhaps just as importantly, it also provides a way to hide the extremely messy steps needed to handle the numerical issues that come out of trying to repeatedly solve coupled optimization problems, where the constraints in one problem may be defined in terms of slightly inaccurate results from an earlier problem. Most of the constraints presented in this paper must be carefully relaxed and monitored in order to handle these numerical precision issues.

13. We plan to release the code in due course.

14. <http://www.eecs.harvard.edu/econcs/jopt>

We are aided in that almost all optimization formulations fall into one of two classes. The first, used by winner determination, the activity rule, the closing rule, and constraint generation in pricing, is a MIP that finds trades that maximize value, holding prices and slacks constant. The second, used by the three pricing stages, is an LP that holds trades constant, seeking to minimize slack, profit, or prices. We take advantage of the commonality of these problems by using common LP/MIP “Builders” that differ only by a few functional hooks to provide the correct variables for optimization.

8. Experimental Analysis

In this section we report the results of a set of experiments that are designed to provide a proof-of-concept for ICE. The results illustrate the scalability of ICE to realistic problem sizes and provide evidence of the effectiveness of the elicitation process and the techniques to bound the efficiency of the provisional trade.

In the experiments, the δ -parameter in MRPAR set to zero and both MRPAR and the DIAR activity rule fire in every round. The rule used to define the ϵ -parameter in DIAR is exactly as described in Section 4.4. We adopt the same straightforward bidding agents that were employed in Section 6 (see the Appendix for details.) The bidding agents adopt a heuristic in meeting the activity rule and seek to maximize the amount of slack maintained in the bid-tree. In simulation, we terminate ICE when the per-agent error in payment relative to the correct payment is within 5% of the average per-agent value for the efficient trade. On typical instances, this incurs an additional 4 rounds beyond those that would be required if we had a last-and-final round in our simulations.

All timing is wall clock time, and does not separately count the large number of parallel threads of execution in the system. The experiments were run on a dual-processor dual-core Pentium IV 3.2GHz with 8GB of memory and CPLEX 10.1. All results are averaged over 10 trials. The instances are available at <http://www.eecs.harvard.edu/~blubin/ice>.

8.1 Problem Instance Generator

Our instance generator begins by generating a set G of good *types*. Next, for each $j \in G$ it creates $d \geq 1$ copies of each good, forming a total potential supply in the market of $d|G|$ goods (exactly how many units are in supply depends on the precise structure of bid trees). Each unit is assigned to one of the bidders uniformly at random. The generator creates a bid tree T_i for each bidder by recursively growing it, starting from the root and adopting two phases. For the tree above *depthLow*, each node receives a number of children drawn uniform between *outDegreeLow* and *outDegreeHigh* (a percentage of which are designated as leaves), resulting in an exponential growth in the number of nodes during this phase. By the *width* at some depth we refer to the number of nodes at that depth. Below this point, we carefully control the expected number of children at each node in order to make the expected width conform to a triangle distribution over depth from *depthLow* to *depthMid* to *depthHigh*: we linearly increase the expected width at each depth between *depthLow* and *depthMid* to a fixed multiple (ξ) of the width at *depthLow*, and then linearly decrease the

expected width back to zero by *depthHigh*.¹⁵ This provides complex and deep trees without inherently introducing an exponential number of nodes.

Each internal node must be assigned the parameters for its interval choose operator. We typically choose y with a high-triangle distribution between 1 and the number of children and x with a low-triangle distribution between 1 and y . This will bias towards the introduction of IC operators that permit a wide choice in the number of children. Each internal node must also be assigned a bonus drawn according to a uniform distribution. Each leaf node is assigned as a “buy” node with a probability $\psi \in [0, 1]$, and then a specific good type for that node is chosen from among those good types for sale in the market. The node is assigned a quantity by drawing from a low-triangle distribution between 1 and the total number in existence.¹⁶ A unit value for the node is then drawn from a specific “buy” distribution, typically uniform, which is multiplied by the quantity and assigned as the node’s bonus. The leaf nodes assigned as “sell” nodes have their goods and bonuses determined similarly, this time with goods selected from among those previously assigned to the bidder.¹⁷

8.2 Empirical Results: Scalability

Figure 18 shows runtime performance of the system as we increase the number of bidders while holding all other parameters constant. In this example, 100 goods in 20 types are being traded by bidders with an average of 104 node trees. The graph shows total CPU time for all parts of the system. While we see super-linear growth in solve time with number of bidders, the constants of this growth are such that markets with large numbers of bidders can be efficiently solved (solving for 20 bidders in around 40 minutes). The error bars in all plots are for the standard error of the statistic.

In Figure 19 we can see the effect of varying the number of types of goods (there are 5 units of each good in the supply) on computation time. For this example we adopt 10 bidders, and the same tree generation parameters, and thus bid trees with an average of 104 nodes. A likely explanation for eventual concavity of the run-time performance is a decrease in the average (item) price quoted at the end of each market (see Figure 20). The average price is a good proxy for the competitiveness of the market. Adding goods to the problem will initially make the winner determination problem more difficult, but only until there is a large over supply at which point competition drops and the outcome is easier to determine. We can see a similar effect in Figure 21, which shows the effect of varying the number of copies of each good though it occurs more slowly.

Figure 22 summarizes the speed-up in computing prices from the improvements offered in Section 5.2. Using provisional locking and lazy constraints can dramatically improve performance in most every round, as illustrated by the improvement in the total time to

15. Note that by setting $depthLow=depthMid=depthHigh$ one can still grow a full tree of a given depth by eliminating phase 2.

16. The total number of goods of a given type in existence may not actually be available for purchase at any price given the structure of seller trees. Thus a bias towards small quantities in “buy” nodes and large quantities in “sell” nodes produces more interesting problem instances.

17. In our experiments, we vary $2 \leq |G| \leq 128$, $1 \leq d \leq 128$, $2 \leq |N| \leq 20$, $2 \leq outDegreeLow \leq 8$, $2 \leq outDegreeHigh \leq 8$, $2 \leq depthLow \leq 6$, $2 \leq depthMid \leq 6$, $2 \leq depthHigh \leq 8$, set a balanced buy probability $\psi = 0.5$, and set width multiplier during the second phase to $\xi = 2$. In these examples, buy node bonuses were drawn uniformly from $[10, 100]$, sell nodes bonuses were drawn uniformly from $[-100, -10]$ and internal nodes bonuses uniformly from $[-25, 25]$.

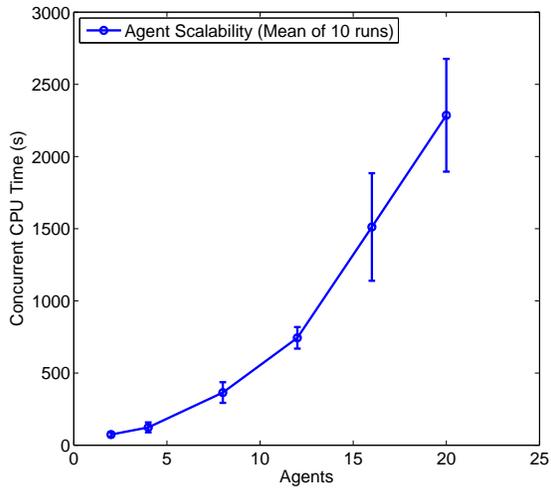


Figure 18: Effect of the number of bidders on the run-time of ICE

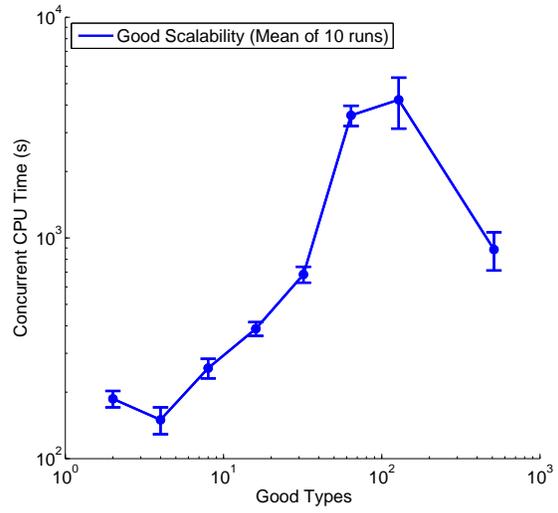


Figure 19: Effect of the number of good types on the run-time of ICE

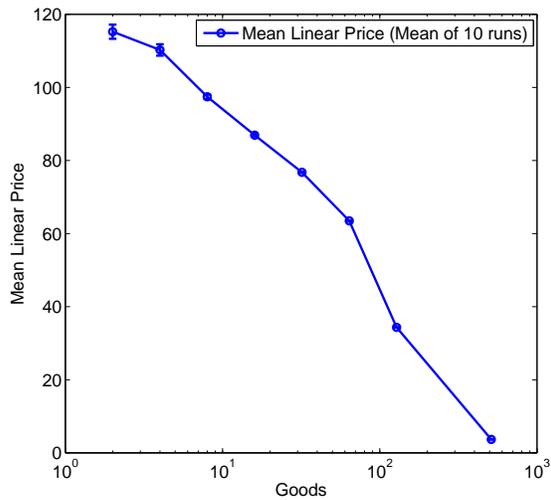


Figure 20: Effect of the number of goods on the average item price upon termination of ICE.

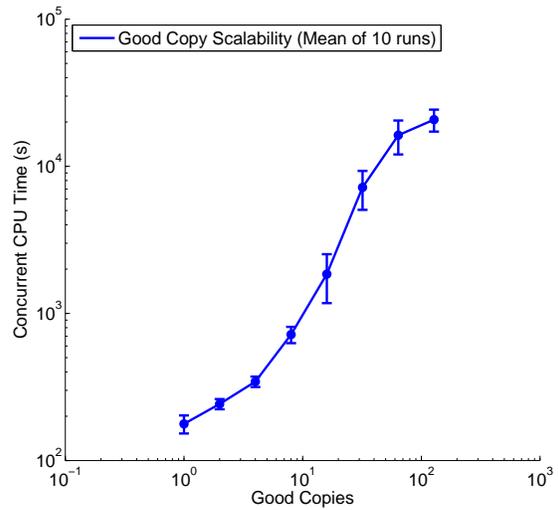


Figure 21: Effect of the number of copies of goods on the run-time of ICE

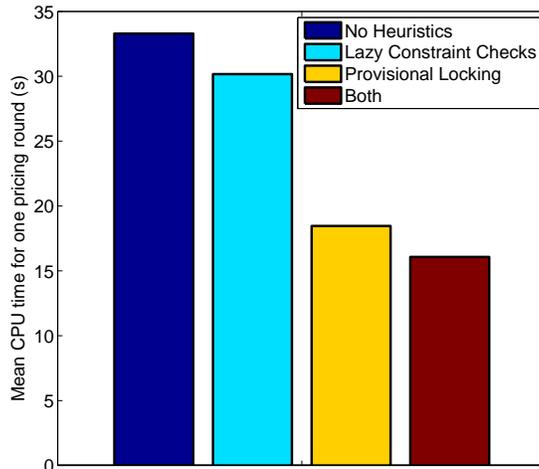


Figure 22: Effect of provisional locking and lazy constraints on pricing CPU time

compute prices across all rounds of 48%. Additionally, we find that heuristically seeding the initial constraint sets (\mathbb{F}) results in around a 26% speedup in run-time for the otherwise expensive first round.

Figures 23 and 24 illustrate the change in run time with the size of agent bid trees. Here we use only the first phase of our tree-generator to avoid confounding the effects of size with structural complexity. In both experiments, 100 goods in 20 types were being traded by 10 bidders. In Figure 23 we vary the number of children of any given node while Figure 24 we vary the depth of the tree. Increasing the branching factor and/or tree depth results in an exponential growth in tree size, which necessarily corresponds to an exponential growth in runtime. However, if we account for this by instead plotting against the number of nodes in the trees, we see that both graphs indicate a *near-polynomial increase in runtime with tree size*. We fit a polynomial function to this data of the form $y = Ax^b$, indicating that this growth is approximately of degree 1.5 in the range of tree sizes considered in these experiments.

8.3 Empirical Results: Economic Properties

The second set of results that we present focus on the economic properties of ICE: the efficiency of trade across rounds, the effectiveness of preference elicitation, and the accuracy and stability of prices. For this set of experiments we average over 10 problem instances, each with 8 bidders, a potential supply of 100 goods in 20 types, and bid trees with an average of 104 nodes.

Figure 25 plots the *true* efficiency of the trades computed at pessimistic (lower bounds \underline{v}), provisional (alpha-valuation) v^α and optimistic (upper bounds \bar{v}) across rounds. In this graph and those that follow, the x -axis indicates the number of rounds completed as a percentage of the total number of rounds until termination which enables results to be aggregated across multiple instances, each of which can have a different number of total

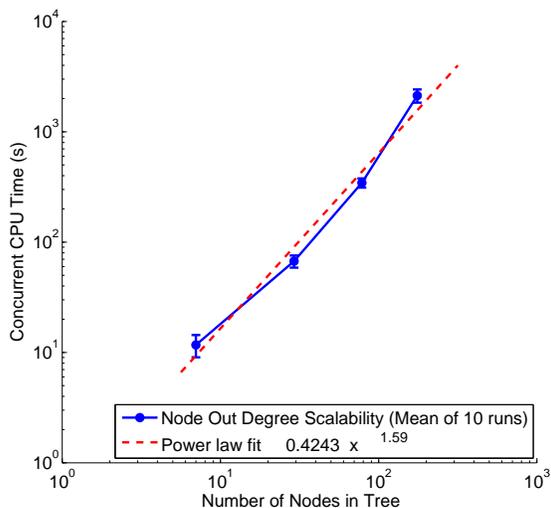


Figure 23: Effect of bid-tree size on run-time of ICE: Varying the node-out degree.

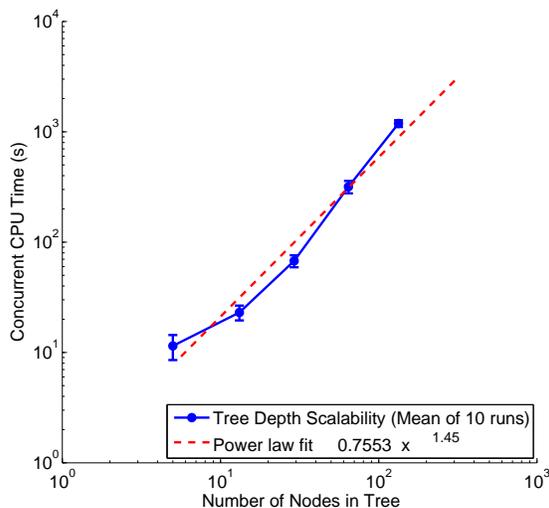


Figure 24: Effect of bid-tree size on run-time of ICE: Varying the tree depth.

rounds.¹⁸ The vertical (dashed) line indicates the average percentage complete when the trade is provably 95% efficient. The exchange remains open past this point while payments converge to the correct Threshold payments (and because we simulate the outcome of the last-and-final round by continuing progress with our straightforward bidding agents). The two lines either side represent one standard error of this statistic.

Parenthetically, we had earlier commented that the number of units of goods (100 in these experiments) is the *potential* supply in the market but need not indicate the actual number of goods in the efficient trade. For a data point on this, in these examples an average of 61.1% of the goods are traded in the efficient trade, and all 8 bidders participate in the efficient trade in all 10 instances.

In Figure 25, we see that the exchange quickly converges to highly efficient trades, taking an average of 6.8 rounds to achieve efficiency. In general, the optimistic trade has higher efficiency than the pessimistic one, while the provisional (alpha) trade is typically better than both. This justifies the design decision to adopt the provisional valuations and provisional trade in driving the exchange dynamics. One can also confirm that exchanges designed in the traditional paradigm of *improving* bids would indeed have little useful feedback in early rounds: the efficiency of the pessimistic trade – all that would be available without information about the upper-bounds of bidder valuations – is initially very poor.

Figure 26 shows the average amount of revelation caused by MRPAR and DIAR across the (normalized) rounds of the exchange. Revelation is measured here in terms of the

18. Each data point represents the average across the 10 instances, and is determined by averaging the underlying points in its neighborhood. Error-bars indicate the standard error (SE) of this mean. Thus, the figures are essentially a histogram rendered as a line graph. We generally use 8 bins in these histograms, slightly perturbing this should the choice result in artifacts.

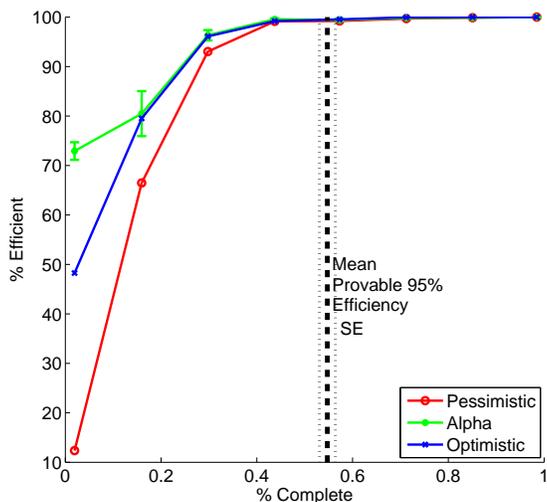


Figure 25: Efficiency of the pessimistic, provisional, and optimistic trades across rounds.

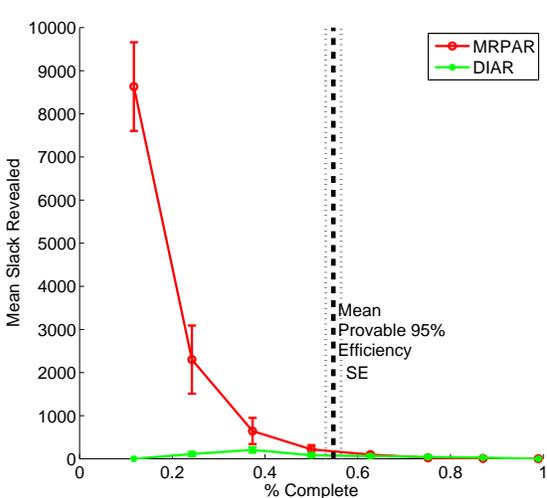


Figure 26: Average reduction in value uncertainty due to each rule.

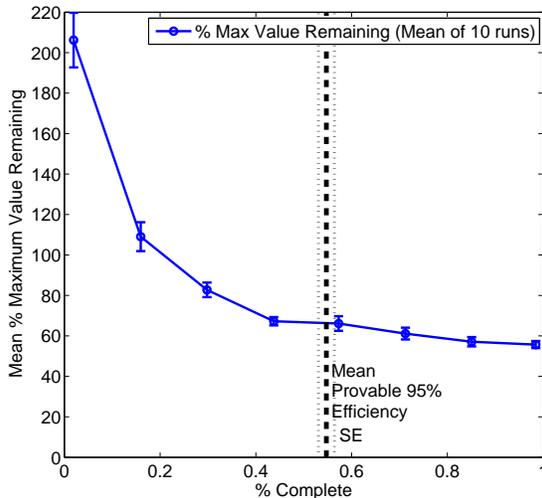


Figure 27: Information revelation: Average value uncertainty in each round

absolute tightening of upper and lower bounds, summed across the bid trees. It is clear that the MRPAR activity rule is the main driving force behind the revelation of information and that the vast majority of revelation (in absolute terms) occurs within the first 25% of rounds. DIAR plays a role in making progress towards identifying the efficient trade but only once MRPAR has substantially reduced the value uncertainty and despite firing in every round. One can think of MRPAR as our rocket’s main engine, and DIAR as a thruster for mid-course correction.

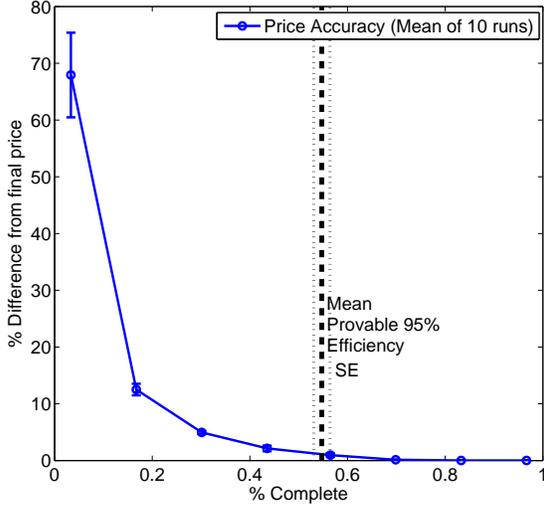


Figure 28: Price trajectory: Closeness of prices in each round to the final prices

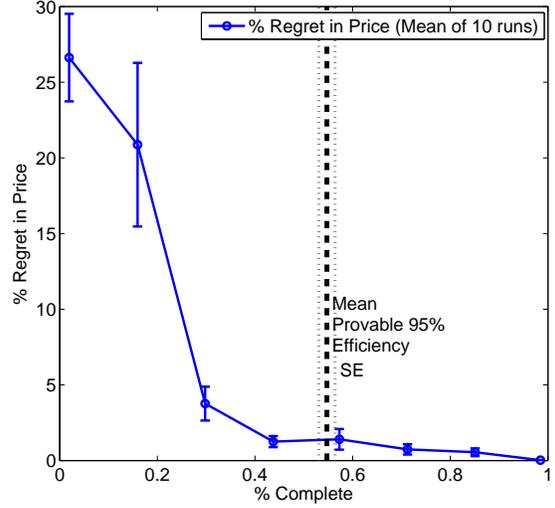


Figure 29: Regret in best-response by bidders due to price inaccuracy relative to final prices.

In Figure 27 we consider how much value uncertainty remains in each round. For this, we define a normalized measure of the amount of slack remaining in agent bid trees T :

$$\frac{1}{n} \sum_{i \in N} \frac{1}{|T_i|} \sum_{\beta \in T_i} \frac{\bar{v}_i(\beta) - \underline{v}_i(\beta)}{\max(|\check{v}_i(\beta)|, |\hat{v}_i(\beta)|)} \times 100\%, \quad (46)$$

where $\check{v}_i(\beta)$ and $\hat{v}_i(\beta)$ are respectively the minimum and maximum values node β could possibly contribute, subject to the interval-choose constraints, defined recursively as:

$$\check{v}_i(\beta) = \begin{cases} v_i(\beta) & , \text{ if } \beta \text{ is a leaf} \\ \min_{s \subseteq \text{children}(\beta): IC_x(\beta) \leq |s| \leq IC_y(\beta)} \sum_{\chi \in s} \check{v}_i(\chi) & , \text{ otherwise} \end{cases}$$

$$\hat{v}_i(\beta) = \begin{cases} v_i(\beta) & , \text{ if } \beta \text{ is a leaf} \\ \max_{s \subseteq \text{children}(\beta): IC_x(\beta) \leq |s| \leq IC_y(\beta)} \sum_{\chi \in s} \hat{v}_i(\chi) & , \text{ otherwise} \end{cases}$$

For a bidder that is mainly buying items below a node β , term $|\hat{v}_i(\beta)|$ will dominate in the denominator; c.f. term $|\check{v}_i(\beta)|$ for a bidder that is mainly selling items below a node. ICE reaches efficiency with the average node retaining an amount of slack of more than 61.6% of the maximum value that the node could contribute to the bidders value. By termination this number has only dropped to 56.3%. This indicates that ICE does indeed allow bidders to focus their revelation on specific parts of the space of possible trades and retain a large amount of uncertainty.

We now provide two different views on the effectiveness of prices. Figure 28 shows the mean percentage absolute difference between the prices computed in some round and the

prices computed in the final round. Prices quickly converge. In our experiments we have driven the exchange beyond the efficient solution in order to converge to the Threshold payments, but we see that most of the price information is already available at the point of efficiency. Figure 29 provides information about the quality of the price feedback. We plot the ‘regret’, averaged across bidders and runs, from the best-response trade as determined from intermediate prices in comparison to the best-response to final prices, where the decision regret is defined in terms of lost payoff at those final prices. Let $\bar{\pi}$ denote prices at true values. Define the average regret, given prices $\hat{\pi}$, as:

$$\frac{1}{n} \sum_{i \in N} \left(1 - \frac{v_i(\lambda^*(\hat{\pi})) - p^{\bar{\pi}}(\lambda^*(\hat{\pi}))}{v_i(\lambda_i^*(\bar{\pi})) - p^{\bar{\pi}}(\lambda_i^*(\bar{\pi}))} \right) \times 100\%, \quad (47)$$

where $\lambda_i^*(\pi) = \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} [v_i(\lambda_i) - p^\pi(\lambda_i)]$. The graph shows that the average price regret is low, with an average of 11.24% when averaged over the rounds needed to find the efficient solution and 6.97% when averaged across all rounds. Because it tends to fall throughout the exchange as more information is revealed, the linear price feedback becomes more and more informative as the rounds proceed, which is exactly to be expected given the additional information at the center’s disposal.

Finally, we present experimental results that relate to the two methods that ICE employs to bound the final efficiency of the pessimistic trade. Figure 30 plots the total error in price accuracy ($\sum_i \delta_i$), i.e. with respect to provisional valuations v_i^α and provisional trade λ^α , across rounds and normalized to the total true value of the efficient trade. From this we can see that linear prices are quite effective in balancing supply and demand despite the combinatorics of the domain. The total (additive) error bound begins at an already low 8.5% of the total value of the efficient trade and falls towards an average of around 3% in the final round.¹⁹ Figure 31 compares the actual efficiency of the pessimistic trade with that estimated by the ω^{direct} bound on efficiency that is available to the exchange. This confirms that the direct bound is reasonably tight, and effective in bounding the true efficiency.

9. Conclusions

In this work we designed and implemented a scalable and highly expressive iterative combinatorial exchange. The design includes many interesting features, including: a new bid-tree language for exchanges, a new method to construct approximate linear prices from expressive languages, a proxied architecture with optimistic and pessimistic valuations coupled with price-based activity rules to drive preference elicitation, and a direct method to estimate the final efficiency of the trade in terms of valuation bounds. Particularly interesting is that by adopting proxy agents with direct, expressive bids the exchange is able to achieve provable efficiency (albeit with straightforward bidders) *despite* using only simple, linear prices. We are not aware of earlier designs for iterative auctions, two-sided or otherwise, that are able to couple linear price feedback with provable guarantees of this kind. Experimental results with automated, simple bidding agents indicate good behavior in terms of both scalability and economic properties. Having bounds on values also enables progress in

19. Note however that this does not immediately imply that this bound is available to the exchange because the actual values are unknown. Rather, one would need to adopt a worst-case, price-based bound (and together with δ -MRPAR) of the form in Eq. 30.

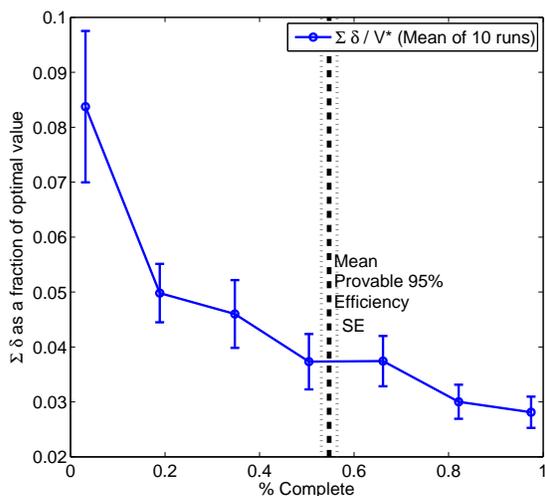


Figure 30: Total error in prices for provisional valuation and trade as a fraction of value of efficient trade

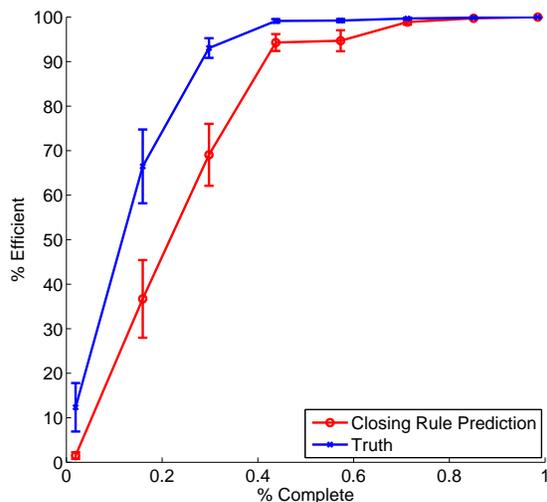


Figure 31: Comparison between the actual efficiency of the pessimistic trade and the ω^{direct} bound.

early rounds and even while there is no efficient trade at pessimistic values. This would not be possible in a more standard design with one-sided information, in which bidders simply *improve* their offers across rounds without providing information about the best possible improvement.

There are many intriguing opportunities for future work. It will be especially interesting to instantiate special-cases of the ICE design to domains for which there exist strategyproof, static (two-sided) combinatorial market designs. This would bring straightforward bidding strategies into an *ex post* Nash equilibrium. For example, it should also be possible to integrate methods such as trade-reduction (McAfee, 1992) and its generalizations (Chu & Shen, 2007; Babaioff & Walsh, 2005; Gonen et al., 2007) in domains with restricted expressiveness. We can also consider ICE as a combinatorial *auction* rather than exchange, where a direct appeal to VCG payments would provide incentive compatibility. The other two major directions for future work are to: (a) modify the design to allow bidders to refine the *structure*, not just the valuation bounds on their *TBBL* tree, across rounds; (b) extend ICE to work in a dynamic environment with a changing bidder population, for instance maintaining linear price feedback and periodically clearing. Recent progress in on-line mechanism design includes truthful, dynamic double auctions for very simple expressiveness (Bredin et al., 2007; Blum et al., 2006), but does not extend to the kind of expressiveness and price sophistication present in ICE; see Parkes (2007) for a recent survey.

Acknowledgments

This work is supported in part by NSF grant IIS-0238147. Nick Elprin, Loizos Michael and Hassan Sultan contributed to earlier versions of this work. Our thanks to the students in Al Roth’s class (Econ 2056) who participated in a trial of this system and to Cynthia Barnhart

for airline domain expertise. We would also like to thank Evan Kwerel and George Donohue for early motivation and encouragement. Some of the computation used in the preparation of this manuscript was performed on the Crimson Grid. Finally, this paper’s genesis is from CS 286r “Topics at the Interface of Computer Science and Economics” taught at Harvard in Spring 2004. Thanks to all the students for their many early, innovative ideas.

References

- Ausubel, L., Cramton, P., & Milgrom, P. (2006). The clock-proxy auction: A practical combinatorial auction design. In Cramton et al. (Cramton et al., 2006), chap. 5.
- Ausubel, L. M., & Milgrom, P. (2002). Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1, 1–42.
- Babaioff, M., & Walsh, W. E. (2005). Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. *Decision Support Systems*, 39, 123–149.
- Ball, M., Donohue, G., & Hoffman, K. (2006). Auctions for the safe, efficient, and equitable allocation of airspace system resources. In Cramton et al. (Cramton et al., 2006), chap. 20.
- Ball, M. O., Ausubel, L. M., Berardino, F., Cramton, P., Donohue, G., Hansen, M., & Hoffman, K. (2007). Market-based alternatives for managing congestion at new york’s laguardia airport. In *Proceedings of AirNeth Annual Conference*.
- Bererton, C., Gordon, G., & Thrun, S. (2003). Auction mechanism design for multi-robot coordination. In *Proc. 17th Annual Conf. on Neural Information Processing Systems (NIPS’03)*.
- Bertsimas, D., & Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific.
- Blum, A., Sandholm, T., & Zinkevich, M. (2006). Online algorithms for market clearing. *Journal of the ACM*, 53, 845–879.
- Boutilier, C. (2002). Solving concisely expressed combinatorial auction problems. In *In Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 359–366.
- Boutilier, C., & Hoos, H. (2001). Bidding languages for combinatorial auctions. In *Proc. 17th International Joint Conference on Artificial Intelligence*, pp. 1121–1217.
- Bredin, J., Parkes, D. C., & Duong, Q. (2007). Chain: A dynamic double auction framework for matching patient agents. *Journal of Artificial Intelligence Research*. To appear.
- Cavallo, R., Parkes, D. C., Juda, A. I., Kirsch, A., Kulesza, A., Lahaie, S., Lubin, B., Michael, L., & Shneidman, J. (2005). TBBL: A Tree-Based Bidding Language for Iterative Combinatorial Exchanges. In *Multidisciplinary Workshop on Advances in Preference Handling (IJCAI)*.
- Chu, L. Y., & Shen, Z. M. (2007). Truthful double auction mechanisms for e-marketplace. *Operations Research*. To appear.
- Compte, O., & Jehiel, P. (2007). Auctions and information acquisition: Sealed-bid or Dynamic Formats?. *Rand Journal of Economics*. To appear.

- Conen, W., & Sandholm, T. (2001). Preference elicitation in combinatorial auctions. In *Proc. 3rd ACM Conf. on Electronic Commerce (EC-01)*, pp. 256–259. ACM Press, New York.
- Cramton, P. (2003). Electricity Market Design: The Good, the Bad, and the Ugly. In *Proceedings of the Hawaii International Conference on System Sciences*.
- Cramton, P. (2006). Simultaneous ascending auctions. In Cramton et al. (Cramton et al., 2006), chap. 3.
- Cramton, P., Kwerel, E., & Williams, J. Efficient relocation of spectrum incumbents. *Journal of Law and Economics*, 41, 647–675.
- Cramton, P., Shoham, Y., & Steinberg, R. (Eds.). (2006). *Combinatorial Auctions*. MIT Press.
- de Vries, S., Schummer, J., & Vohra, R. V. (2007). On ascending Vickrey auctions for heterogeneous objects. *Journal of Economic Theory*, 132, 95–118.
- de Vries, S., & Vohra, R. V. (2003). Combinatorial auctions: A survey. *Inform Journal on Computing*, 15(3), 284–309.
- Dias, M., Zlot, R., Kalra, N., & Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94, 1257–1270.
- Dunford, M., Hoffman, K., Menon, D., Sultana, R., & Wilson, T. (2003). Testing linear pricing algorithms for use in ascending combinatorial auctions. Tech. rep., SEOR, George Mason University. Submitted to INFORMS J.Computing.
- Fu, Y., Chase, J., Chun, B., Schwab, S., & Vahdat, A. (2003). Sharp: an architecture for secure resource peering. In *Proceedings of the 19th ACM symposium on Operating systems principles*, pp. 133–148. ACM Press.
- Gerkey, B. P., & Mataric, M. J. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*, 18, 758–768.
- Gonen, M., Gonen, R., & Pavlov, E. (2007). Generalized trade reduction mechanisms. In *Proc. 8th ACM Proceedings on Electronic Commerce (EC'07)*.
- Hudson, B., & Sandholm, T. (2004). Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *Proc. 3rd Int. Joint. Conf. on Autonomous Agents and Multi Agent Systems*, pp. 386–393.
- Krishna, V. (2002). *Auction Theory*. Academic Press.
- Krych, D. (2003). Calculation and analysis of Nash equilibria of Vickrey-based payment rules for combinatorial exchanges.. Undergraduate Thesis. Available from <http://www.eecs.harvard.edu/econcs>.
- Kwasnica, A. M., Ledyard, J. O., Porter, D., & DeMartini, C. (2005). A new and improved design for multi-object iterative auctions. *Management Science*, 51, 419–434.
- Kwerel, E., & Williams, J. (2002). A proposal for a rapid transition to market allocation of spectrum. Tech. rep., FCC Office of Plans and Policy.

- Lahaie, S. M., Constantin, F., & Parkes, D. C. (2005). More on the power of demand queries in combinatorial auctions: Learning atomic languages and handling incentives. In *Proc. 19th Int. Joint Conf. on Artificial Intell. (IJCAI'05)*.
- Lahaie, S. M., & Parkes, D. C. (2004). Applying learning algorithms to preference elicitation. In *Proc. ACM Conf. on Electronic Commerce*, pp. 180–188.
- McAfee, R. P. (1992). A dominant strategy double auction. *J. of Economic Theory*, 56, 434–450.
- Milgrom, P. (2004). *Putting Auction Theory to Work*. Cambridge University Press.
- Milgrom, P. (2007). Package auctions and package exchanges (2004 Fisher-Schultz lecture). *Econometrica*, 75, 935–966.
- Mishra, D., & Parkes, D. C. (2007). Ascending price Vickrey auctions for general valuations. *Journal of Economic Theory*, 132, 335–366.
- Myerson, R. B., & Satterthwaite, M. A. (1983). Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28, 265–281.
- Nemhauser, G., & Wolsey, L. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience.
- Nisan, N. (2006). Bidding languages for combinatorial auctions. In Cramton et al. (Cramton et al., 2006), chap. 9.
- Parkes, D. C. (2007). On-line mechanisms. In Nisan, N., Roughgarden, T., Tardos, E., & Vazirani, V. (Eds.), *Algorithmic Game Theory*, chap. 16. Cambridge University Press. To appear.
- Parkes, D. C., Kalagnanam, J. R., & Eso, M. (2001). Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proc 17th International Joint Conference on Artificial Intelligence*, pp. 1161–1168.
- Parkes, D. C., & Ungar, L. H. (2000a). Iterative combinatorial auctions: Theory and practice. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pp. 74–81.
- Parkes, D. C., & Ungar, L. H. (2000b). Preventing strategic manipulation in iterative auctions: Proxy agents and price-adjustment. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pp. 82–89.
- Rassenti, S. J., Smith, V. L., & Bulfin, R. L. (1982). A combinatorial mechanism for airport time slot allocation. *Bell Journal of Economics*, 13, 402–417.
- Rothkopf, M. H., Pekeč, A., & Harstad, R. M. (1998). Computationally manageable combinatorial auctions. *Management Science*, 44(8), 1131–1147.
- Saatcioglu, K., Stallaert, J., & Whinston, A. B. (2001). Design of a financial portal. *Communications of the ACM*, 44, 33–38.
- Sandholm, T. (2006). Optimal winner determination algorithms. In Cramton et al. (Cramton et al., 2006), chap. 14.

- Sandholm, T. (2007). Expressive Commerce and Its Application to Sourcing: How We Conducted \$35 Billion of Generalized Combinatorial Auctions. *AI Magazine*. To appear.
- Sandholm, T., & Boutilier, C. (2006). Preference elicitation in combinatorial auctions. In Cramton et al. (Cramton et al., 2006), chap. 10.
- Smith, T., Sandholm, T., & Simmons, R. (2002). Constructing and clearing combinatorial exchanges using preference elicitation. In *AAAI-02 workshop on Preferences in AI and CP: Symbolic Approaches*.
- Vossen, T. W. M., & Ball, M. O. (2006). Slot trading opportunities in collaborative ground delay programs. *Transportation Science*, 40, 15–28.
- Wurman, P. R., & Wellman, M. P. (2000). AkBA: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, pp. 21–29.

Appendix

A1. Computation for MRPAR

In this section we show that MRPAR can be computed by solving a sequence of 3 MIPs. We begin by considering the special case of $\delta = 0$. The general case follows almost immediately. Define a *candidate passing trade*, λ_i^L , as:

$$\lambda_i^L \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} \underline{v}_i(\lambda_i) - p^\pi(\lambda_i) \quad (48)$$

breaking ties

- (i) to maximize $\bar{v}_i(\lambda_i) - \underline{v}_i(\lambda_i)$
- (ii) in favor of λ_i^α

This can be computed by solving one MIP to maximize $\underline{v}_i(\lambda_i) - p^\pi(\lambda_i)$, followed by a second MIP in which this objective is incorporated as a constraint and $\bar{v}_i(\lambda_i) - \underline{v}_i(\lambda_i)$ becomes the objective. Given perturbed valuation \tilde{v}_i , defined with respect to trade λ_i^L (as in Section 4.2), we can define a *witness trade*, λ_i^U , as:

$$\lambda_i^U \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} \tilde{v}_i(\lambda_i) - p^\pi(\lambda_i). \quad (49)$$

This can be found by solving a third MIP. Given prices π , provisional trade λ_i^α and bid tree T_i , the *computational MRPAR* rule (C-MRPAR) for the case of $\delta = 0$ can now be defined as:

- (1) $\underline{v}_i(\lambda_i^L) - p^\pi(\lambda_i^L) \geq \tilde{v}_i(\lambda_i^U) - p^\pi(\lambda_i^U)$,and
- (2) $\lambda_i^L = \lambda_i^\alpha$, or $\underline{v}_i(\lambda_i^L) - p^\pi(\lambda_i^L) > \tilde{v}_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha)$

We now establish that C-MRPAR is equivalent to MRPAR, as defined by Eqs. (22, 23, 24).

Lemma 2 *Given trades λ_i and λ_i' , prices π , and tree T_i , then $\theta^\pi(\lambda_i, \lambda_i', v_i') \geq 0$, $\forall v_i' \in T_i$ if and only if $\underline{v}_i(\lambda_i) - p^\pi(\lambda_i) \geq \tilde{v}_i(\lambda_i') - p^\pi(\lambda_i')$, where \tilde{v}_i is defined with respect to trade λ_i .*

Proof:Direction (\Rightarrow) is immediate since $\tilde{v}_i \in T_i$. Consider direction (\Leftarrow) and suppose, for contradiction, that $\underline{v}_i(\lambda_i) - p^\pi(\lambda_i) \geq \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i)$ but there exists some $v'_i \in T_i$ such that $v'_i(\lambda_i) - p^\pi(\lambda_i) < v'_i(\lambda'_i) - p^\pi(\lambda'_i)$. Subtract $\sum_{\beta \in \lambda_i \cap \lambda'_i} [v'_i(\beta) - \underline{v}_i(\beta)]$ from both sides, where $\beta \in \lambda_i$ indicates that node β is satisfied by trade λ_i , to get

$$\begin{aligned} & \sum_{\beta \in \lambda_i \setminus \lambda'_i} v'_i(\beta) + \sum_{\beta \in \lambda_i \cap \lambda'_i} v'_i(\beta) - \sum_{\beta \in \lambda_i \cap \lambda'_i} v'_i(\beta) + \sum_{\beta \in \lambda_i \cap \lambda'_i} \underline{v}_i(\beta) - p^\pi(\lambda_i) < \\ & \sum_{\beta \in \lambda'_i \setminus \lambda_i} v'_i(\beta) + \sum_{\beta \in \lambda_i \cap \lambda'_i} v'_i(\beta) - \sum_{\beta \in \lambda_i \cap \lambda'_i} v'_i(\beta) + \sum_{\beta \in \lambda_i \cap \lambda'_i} \underline{v}_i(\beta) - p^\pi(\lambda'_i) \end{aligned} \quad (50)$$

$$\Rightarrow \sum_{\beta \in \lambda_i \setminus \lambda'_i} \underline{v}_i(\beta) + \sum_{\beta \in \lambda_i \cap \lambda'_i} \underline{v}_i(\beta) - p^\pi(\lambda_i) < \sum_{\beta \in \lambda'_i \setminus \lambda_i} \bar{v}_i(\beta) + \sum_{\beta \in \lambda_i \cap \lambda'_i} \underline{v}_i(\beta) - p^\pi(\lambda'_i) \quad (51)$$

$$\Rightarrow \underline{v}_i(\lambda_i) - p^\pi(\lambda_i) < \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i), \quad (52)$$

which is a contradiction. \square

Lemma 3 *Given trade λ_i , prices π , and tree T_i then $\theta^\pi(\lambda_i, \lambda'_i, v'_i) \geq 0$, $\forall v'_i \in T_i$, $\forall \lambda'_i \in \mathcal{F}_i(x^0)$, if and only if $\underline{v}_i(\lambda_i) - p^\pi(\lambda_i) \geq \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i)$, where \tilde{v}_i is defined with respect to trade λ_i and λ'_i is the witness trade.*

Proof:Direction (\Rightarrow) is immediate since $\tilde{v}_i \in T_i$ and $\lambda'_i \in \mathcal{F}_i(x^0)$. Consider direction (\Leftarrow) and suppose, for contradiction, that $\underline{v}_i(\lambda_i) - p^\pi(\lambda_i) \geq \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i)$ but there exists some $\lambda'_i \in \mathcal{F}_i(x^0)$ and $v'_i \in T_i$ such that $\theta^\pi(\lambda_i, \lambda'_i, v'_i) < 0$. By Lemma 2, this means $\underline{v}_i(\lambda_i) - p^\pi(\lambda_i) < \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i)$. But, we have a contradiction because

$$\underline{v}_i(\lambda_i) - p^\pi(\lambda_i) \geq \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i) \quad (53)$$

$$= \max_{\lambda''_i \in \mathcal{F}_i(x^0)} \tilde{v}_i(\lambda''_i) - p^\pi(\lambda''_i) \geq \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i) \quad (54)$$

\square

Theorem 7 *C-MRPAR is equivalent to δ -MRPAR for $\delta = 0$.*

Proof:Comparing Eqs. (25) and (26) with C-MRPAR, and given Lemmas 2 and 3, all that is left to show is that it is sufficient to check λ_i^L , as the only candidate to pass MRPAR. That is, we need to show that if there is some $\lambda_i \in \mathcal{F}_i(x^0)$ that satisfies MRPAR then λ_i^L satisfies MRPAR. We argue as follows:

1. Trade $\check{\lambda}_i$ must solve $\max_{\lambda_i \in \mathcal{F}_i(x^0)} [\underline{v}_i(\lambda_i) - p^\pi(\lambda_i)]$. Otherwise, there is some λ'_i with $\underline{v}_i(\lambda'_i) - p^\pi(\lambda'_i) > \underline{v}_i(\check{\lambda}_i) - p^\pi(\check{\lambda}_i)$. A contradiction with Eq. (25).
2. Trade $\check{\lambda}_i$ must also break ties in favor of maximizing $\bar{v}_i(\lambda_i) - \underline{v}_i(\lambda_i)$. Otherwise, there is some λ'_i with the same profit as $\check{\lambda}_i$ at \underline{v}_i , with $\bar{v}_i(\lambda'_i) - \underline{v}_i(\lambda'_i) > \bar{v}_i(\check{\lambda}_i) - \underline{v}_i(\check{\lambda}_i)$. This implies $\bar{v}_i(\lambda'_i) - \bar{v}_i(\check{\lambda}_i) > \underline{v}_i(\lambda'_i) - \underline{v}_i(\check{\lambda}_i)$, and $\theta^\pi(\lambda'_i, \lambda_i, \bar{v}_i) > \theta^\pi(\lambda'_i, \check{\lambda}_i, \underline{v}_i)$. But, since λ'_i has the same profit as λ_i at \underline{v}_i we have $\theta^\pi(\lambda'_i, \lambda_i, \underline{v}_i) = 0$ and so $\theta^\pi(\lambda'_i, \lambda_i, \bar{v}_i) > 0$. This is a contradiction with Eq. (25).

3. Proceed now by case analysis. Either $\check{\lambda}_i = \lambda_i^\alpha$, in which case we are done because this will be explicitly selected as candidate passing trade λ_i^L . For the other case, let Λ_i^L denote all feasible solutions to Eq. (48) and consider the difficult case when $|\Lambda_i^L| > 1$. We argue that if $\check{\lambda}_i \in \Lambda_i^L$ satisfies MRPAR, then so does any other trade $\lambda'_i \in \Lambda_i^L$, with $\lambda'_i \neq \check{\lambda}_i$. By MRPAR, we have $\theta^\pi(\check{\lambda}_i, \lambda'_i, v'_i) \geq 0, \forall v'_i \in T_i$. In particular, $\tilde{v}_i(\check{\lambda}_i) - p^\pi(\check{\lambda}_i) \geq \tilde{v}_i(\lambda'_i) - p^\pi(\lambda'_i)$, where \tilde{v}_i is defined with respect to λ_i , and equivalently,

$$\underline{v}_i(\check{\lambda}_i) - p^\pi(\check{\lambda}_i) \geq \underline{v}_i(\lambda'_i) - p^\pi(\lambda'_i). \quad (55)$$

On the other hand,

$$\underline{v}_i(\check{\lambda}_i) - p^\pi(\check{\lambda}_i) = \underline{v}_i(\lambda'_i) - p^\pi(\lambda'_i), \quad (56)$$

since both in Λ_i^L . Taking Eq. (55) together with Eq. (56), we must have that λ'_i satisfies no uncertain value nodes in T_i not also satisfied in $\check{\lambda}_i$. Moreover, since $\bar{v}_i(\check{\lambda}_i) - \underline{v}_i(\check{\lambda}_i) = \bar{v}_i(\lambda'_i) - \underline{v}_i(\lambda'_i)$, then both trades must satisfy *exactly* the same uncertain value nodes. Finally, by Eq. (56) the profit from all fixed value nodes in T_i must be the same in both trades. We conclude that the profit is the same for *all* $v'_i \in T_i$ for $\check{\lambda}_i$ and λ'_i at the current prices and MRPAR is satisfied by either trade. \square

To understand the importance of the tie-breaking rule (i) in selecting the candidate passing trade, λ_i^L , in C-MRPAR, consider the following example for MRPAR with $\delta = 0$:

Example 8 *A bidder has XOR(+A, +B) and a value of +5 on the leaf +A and a value range of [5, 10] on leaf +B. Suppose prices are currently 3 for each of A and B and $\lambda_i^\alpha = +B$. The MRPAR rule is satisfied because the market knows that however the remaining value uncertainty on +B is resolved the bidder will always (weakly) prefer +B to +A and +B is λ_i^α . Notice that both +A and +B have the same pessimistic utility, but only +B can satisfy MRPAR. But +B has maximal value uncertainty, and therefore this is selected over +A by C-MRPAR.*

To understand the importance of selecting, and evaluating, λ_i^U , with respect to \tilde{v}_i rather than \bar{v}_i consider the following example (again for $\delta = 0$). It illustrates the role of “shared uncertainty” in the tree, which occurs when multiple trades share a node with uncertain value and the value, although uncertain, will be resolved in the same way for both trades.

Example 9 *A bidder has XOR(+A, +B) and value bounds [5, 10] on the root node and a value of 1 on leaf +A. Suppose prices are currently 3 for each of A and B and $\lambda_i^\alpha = +B$. The MRPAR rule is satisfied because the bidder strictly prefers +A to +B, whichever way the uncertain value on the root node is ultimately resolved. C-MRPAR selects λ_i^L as “buy A”, with payoff $\underline{v}_i(\lambda_i^L) - p^\pi(\lambda_i^L) = 5 + 1 - 3 = 3$. At valuation \bar{v}_i , the witness trade “buy B” would be selected and have payoff $10 - 3 = 7$ and seem to violate MRPAR. But, which ever way the uncertain value at the root is resolved it will affect +A and +B in the same way. This is addressed by setting $\tilde{v}_i(\beta) = \underline{v}_i(\beta) = 5$ on the root node, the same value adopted in determining the payoff from λ_i^L . Evaluated at \tilde{v}_i , the witness is “buy A” and (1) of C-MRPAR is trivially satisfied while (2) is satisfied since $3 > 5 - 3 = 2$.*

For δ -MRPAR with $\delta > 0$, we adopt a slight variation, with a δ -C-MRPAR procedure defined as:

- (1) Check $\theta^\pi(\lambda_i^\alpha, \lambda'_i, v'_i) \geq -\delta$ for all $v'_i \in T_i$, all $\lambda'_i \in \mathcal{F}_i(x^0)$ directly, by application of Lemma 3 with valuation \tilde{v}_i defined with respect to trade λ_i^α , and test

$$\underline{v}_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) \geq \tilde{v}_i(\lambda_i^U) - p^\pi(\lambda_i^U) - \delta \quad (57)$$

- (2) If this is not satisfied then fall back on C-MRPAR to verify Eqs. (23) and (24), with candidate passing trade λ_i^L modified from Eq. (48) to drop tie-breaking in favor of λ_i^α and with the second step of C-MRPAR modified to require $\underline{v}_i(\lambda_i^L) - p^\pi(\lambda_i^L) > \tilde{v}_i(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha) + \delta$, again with \tilde{v}_i defined with respect to λ_i^L .

The argument adopted in the proof of Theorem 7 remains valid in establishing that it is sufficient to consider, λ_i^L , as defined in δ -C-MRPAR, in the case that λ_i^α does not pass the activity rule.

A2: Computation for DIAR

The ϵ -DIAR rule can be verified by solving two MIPs. The first optimization problem identifies the trade with maximal DIAR error for which the current bounds refinement has improved this error by at least ϵ :

$$\Delta_i^P = \max_{\lambda_i \in \mathcal{F}_i(x^0)} [\tilde{v}_i^0(\lambda_i) - p^\pi(\lambda_i) - (\underline{v}_i^0(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))] \quad (58)$$

$$\begin{aligned} \text{s.t. } & (\tilde{v}_i^0(\lambda_i) - p^\pi(\lambda_i) - (\underline{v}_i^0(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))) \\ & - (\tilde{v}_i^1(\lambda_i) - p^\pi(\lambda_i) - (\underline{v}_i^1(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))) \geq \epsilon \end{aligned} \quad (59)$$

$$= -C + \max_{\lambda_i \in \mathcal{F}_i(x^0)} \tilde{v}_i^0(\lambda_i) - p^\pi(\lambda_i) \quad (60)$$

$$\text{s.t. } \tilde{v}_i^0(\lambda_i) - \underline{v}_i^0(\lambda_i^\alpha) - \tilde{v}_i^1(\lambda_i) + \underline{v}_i^1(\lambda_i^\alpha) \geq \epsilon, \quad (61)$$

where \tilde{v}_i^0 and \tilde{v}_i^1 are defined with respect to λ_i^α , v^0 and v^1 represent valuations defined before and after the bidder's refinement respectively, and $C = \underline{v}_i^0(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha)$. Note that the problem could be infeasible, in which case we define $\Delta_i^P := -\infty$.

The second optimization problem identifies the trade with maximal DIAR error for which v^1 still allows for the possibility of valuation bounds that provide an ϵ error reduction over v^0 :

$$\Delta_i^F = \max_{\lambda_i \in \mathcal{F}_i(x^0)} [\tilde{v}_i^0(\lambda_i) - p^\pi(\lambda_i) - (\underline{v}_i^0(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))] \quad (62)$$

$$\begin{aligned} \text{s.t. } & (\tilde{v}_i^0(\lambda_i) - p^\pi(\lambda_i) - (\underline{v}_i^0(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))) \\ & - (\underline{v}_i^1(\lambda_i) - p^\pi(\lambda_i) - (\check{v}_i^1(\lambda_i^\alpha) - p^\pi(\lambda_i^\alpha))) \geq \epsilon \end{aligned} \quad (63)$$

$$= -C + \max_{\lambda_i \in \mathcal{F}_i(x^0)} \tilde{v}_i^0(\lambda_i) - p^\pi(\lambda_i) \quad (64)$$

$$\text{s.t. } \tilde{v}_i^0(\lambda_i) - \underline{v}_i^0(\lambda_i^\alpha) - \underline{v}_i^1(\lambda_i) + \check{v}_i^1(\lambda_i^\alpha) \geq \epsilon, \quad (65)$$

where \tilde{v}_i is defined with respect to λ_i^α , and \check{v}_i is similarly defined with respect to λ_i . The second term in Eq. (63) recognizes that it remains possible to decrease the value on λ_i to the new lower-bound $\underline{v}_i^1(\lambda_i)$, while increasing the value on λ_i^α to the new upper-bound $\bar{v}_i^1(\lambda_i^\alpha)$ except on those nodes that are shared with λ_i , giving $\check{v}_i^1(\lambda_i^\alpha)$. We see that Eq. (65) is equivalent to:

$$\sum_{\beta \in \lambda_i \setminus \lambda_i^\alpha} [\bar{v}_i^0(\beta) - \underline{v}_i^1(\beta)] + \sum_{\beta \in \lambda_i^\alpha \setminus \lambda_i} [\bar{v}_i^1(\beta) - \underline{v}_i^0(\beta)] \geq \epsilon,$$

which calculates the amount of refinement that is still possible in service of reducing the DIAR error. Note the problem could be infeasible, in which case we define $\Delta_i^F := -\infty$. We ultimately compare the two solutions, and the bidder passes DIAR if and only if $\Delta_i^P \geq \Delta_i^F$.

A3: The Automated Bidding Agents and Bidder Feedback

The bidding agent that is used for the simulation experiments is carefully constructed (but rather complex), and attempts to minimize the amount of information it must reveal in order to pass the activity rules and without violating its truthful valuation. In summarizing the behavior of our bidding agents, there are three things to explain: (a) the method that we adopt in place of the last-and-final round; (b) the feedback that is provided by ICE to bidders in meeting MRPAR and DIAR; and (c) the logic that is followed by the bidding agents.

Rather than define a method for bidding agents to adjust their bounds in a last-and-final round we keep ICE open past the point in which it would ordinarily go to last-and-final. Past this point, bidders continue to refine their bounds using a simple heuristic and we terminate each instance when the Threshold payments are within some desired accuracy. This “payment discovery” phase replaces the last-and-final round. Each bidder reduces its uncertainty by some multiplicative factor on all nodes that are active in the current provisional trade or in any of the provisional trades for the economies with bidder i removed. *Please note that this is used for simulation only and is not intended to be employed in a real application of ICE which would contain a last-and-final round.*

Our bidding agents operate in a loop, heuristically modifying their valuation bounds in trying to meet MRPAR and DIAR and querying the proxy for advice. The proxy provides guidance to help the bidding agent further refine its valuation so it can meet the activity rule. For both MRPAR and DIAR, the optimization problems that are solved in checking whether a bidder has satisfied the activity rule also provide information that can guide the bidder. First consider MRPAR and recall that λ_i^L is the candidate passing trade and λ_i^U is the witness trade. The following lemma is easy, and stated without proof:

Lemma 4 *When MRPAR is not satisfied for the current valuation bounds, a bidder must increase a lower bound on at least one node in $\{\lambda_i^L \setminus \lambda_i^U\}$, or decrease an upper bound on at least one node in $\{\lambda_i^U \setminus \lambda_i^L\}$, in order to meet the activity rule.*

Once a simple bidder makes some changes on some subset of these nodes, the bidder can inquire if he has passed the activity rule. The proxy can then respond “yes,” or can revise the set of nodes on which the bidding agent should refine its valuation bounds.

A similar functionality is provided for DIAR. This time the trade that solves the second MIP (with DIAR error Δ_i^F) is provided as feedback, together with information about how

much the bidder must either further reduce the error, or further constrain the possibilities on this trade, to satisfy DIAR. The bidding agent can determine from this information which nodes it must modify, and by how much in total, and is free to decide how much to modify each node to satisfy the rule.

The key to our agent design is the following lemma:

Lemma 5 *The trade with which a straightforward bidder passes MRPAR (for $\delta = 0$) must be a trade that is weakly preferred by the bidder to all other trades for his true valuation.*

Proof:By contradiction. Suppose true valuation $v_i \in T_i$ and trade $\check{\lambda}_i$ meets MRPAR but is not a weakly preferred trade at the true valuation and prices π . Then, there exists a trade $\lambda_i^* \in \mathcal{F}_i(x^0)$ such that $\theta^\pi(\lambda_i^*, \check{\lambda}_i, v_i) > 0$. But, this is a contradiction with MRPAR since $\theta^\pi(\check{\lambda}_i, \lambda'_i, v'_i) \geq 0$ for all $v'_i \in T_i$ and all $\lambda'_i \in \mathcal{F}_i(x^0)$, including $v'_i = v_i$ and $\lambda'_i = \lambda_i^*$. \square

We use this observation to define a procedure UPDATEMRPAR by which a bidder can intelligently refine its valuation bounds to meet MRPAR. Let $\check{\lambda}_i$ be the trade with which we hope to pass MRPAR, and define $u_i(\lambda_i, \pi) = v_i(\lambda_i) - p^\pi(\lambda_i)$, $\underline{u}_i(\lambda_i, \pi) = \underline{v}_i(\lambda_i) - p^\pi(\lambda_i)$, $\tilde{u}_i(\lambda_i, \pi) = \tilde{v}_i(\lambda_i) - p^\pi(\lambda_i)$, where \tilde{v}_i is defined with respect to candidate passing trade $\check{\lambda}_i$. The high-level approach is as follows:

function UPDATEMRPAR

$\check{\lambda}_i \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} u_i(\lambda_i, \pi)$

if $u_i(\check{\lambda}_i, \pi) < 0$ **then**

 reduce slack on $\check{\lambda}_i$ by $u_i(\check{\lambda}_i, \pi)$

end if

$\lambda_i^U \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} \tilde{u}_i(\lambda_i, \pi)$

while $\underline{u}_i(\check{\lambda}_i, \pi) < \tilde{u}_i(\lambda_i^U, \pi)$ **do**

 Heuristically reduce upper bounds on $\lambda_i^U \setminus \check{\lambda}_i$ by $\tilde{u}_i(\lambda_i^U, \pi) - \underline{u}_i(\check{\lambda}_i, \pi)$

 If remaining slack heuristically reduce lower bounds on $\lambda_i \setminus \lambda_i^U$

$\lambda_i^U \in \arg \max_{\lambda_i \in \mathcal{F}_i(x^0)} \tilde{u}_i(\lambda_i, \pi)$

end while

if $\check{\lambda}_i \neq \lambda_i^\alpha$ **then**

while $\underline{u}_i(\check{\lambda}_i, \pi) \leq \tilde{u}_i(\lambda_i^\alpha, \pi)$ **do**

 Heuristically reduce upper bounds on $\lambda_i^\alpha \setminus \check{\lambda}_i$ by $\tilde{u}_i(\lambda_i^\alpha, \pi) - \underline{u}_i(\check{\lambda}_i, \pi)$

 If remaining slack heuristically reduce lower bounds on $\lambda_i \setminus \lambda_i^\alpha$

end while

end if

return $\check{\lambda}_i$

end function

The bidding agent makes use of a couple of optimization modalities that are exposed by the proxy to the bidder. The procedure first chooses the most preferred trade at truth as the trade to pass MRPAR with $\check{\lambda}_i$; the bidding agent requests that the proxy finds this trade by solving a MIP. If the trade has negative profit, then the bidding agent attempts demonstrate positive profit for this trade. Next, the bidding agent enters a loop, wherein it repeatedly requests the proxy to run a MIP that calculates a witness trade λ_i^U w.r.t. $\check{\lambda}_i$.

As long as this witness has more profit than that of what should be the most preferred trade, the bidding agent adjust bounds so as to reverse this mis-ordering. Lastly, because the bidding agent must pass MRPAR, not merely RPAR, the bidding agent attempts to show a strict preference for λ_i over λ_i^α when they are not identical.

In meeting DIAR, the bidding agent responds to the $\Delta^F \geq 0$, and the $\epsilon \geq 0$ parameter provided by the proxy as follows. Let λ^F be the trade chosen in the maximization that calculates Δ^F . The high-level approach is as follows:

```

function UPDATEDIAR
  while Proxy says we still have not passed DIAR do
    if  $\lambda^F$  or  $\lambda^\alpha$  can be modified to reduce DIAR error by  $\epsilon$  over last round then
      Heuristically reduce the upper-bound slack in  $\lambda^F \setminus \lambda^\alpha$ 
      Heuristically reduce the lower-bound slack in  $\lambda^\alpha \setminus \lambda^F$ 
    else
      Heuristically reduce the upper-bound slack in  $\lambda^\alpha \setminus \lambda^F$ 
      Heuristically reduce the lower-bound slack in  $\lambda^F \setminus \lambda^\alpha$ 
    end if
  end while
end function

```

The bidding agent attempts to make the current failing trade pass DIAR if possible by reducing the error with respect to that trade. Otherwise, it reduces bounds to prove that DIAR could not be made to pass on that trade and loops on to the next trade.