ICE: An Iterative Combinatorial Exchange

David C. Parkes[*] [†] Ruggiero Cavallo[†] Nick Elprin[†] Adam Juda[†] Sébastien Lahaie[†]
Benjamin Lubin[†] Loizos Michael[†] Jeffrey Shneidman[†] Hassan Sultan[†]

ABSTRACT

We present the first design for a fully expressive iterative combinatorial exchange (ICE). The exchange incorporates a tree-based bidding language that is concise and expressive for CEs. Bidders specify lower and upper bounds on their value for different trades. 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 ensures progress across rounds. A VCG-based payment scheme that has been shown to mitigate opportunities for bargaining and strategic behavior is used to determine final payments. The exchange is fully implemented and in a validation phase.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; J.4 [**Computer Applications**]: Social and Behavioral Sciences —*Economics*

General Terms: Algorithms, Economics, Theory.

Keywords: Combinatorial exchange, Threshold payments, VCG, Preference Elicitation.

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 [20]. In a combinatorial auction (CA), a single seller has multiple heterogeneous items up for sale [11]. Buyers may have complementarities or substitutabilities between goods, and are provided with an expressive bidding language. A common goal in both market designs is to determine the efficient allocation, which is the allocation that maximizes total value.

A combinatorial exchange (CE) [24] 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 she is willing to pay $1 million for a trade where she obtains licenses for New York City, Boston, and Philadelphia, and loses her license for Washington DC. Thus, unlike a DA, a CE allows all participants to express complex valuations via expressive bids. Unlike a CA, a CE allows for fragmented ownership, with multiple buyers and sellers and agents that are both buying and selling.

CEs have received recent attention both in the context of wireless spectrum allocation [18] and for airport takeoff and landing slot allocation [3]. In both of these domains there are incumbents with property rights, and it is important 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 [13]. The instantiation of our general purpose design to specific domains is a compelling next step in our research.

This paper presents the first design for a fully expressive *iterative* combinatorial exchange (ICE). The genesis of this project was a class, CS 286r "Topics at the Interface between Economics and Computer Science," taught at Harvard University in Spring 2004.[1] The entire class was dedicated to the design and prototyping of an iterative CE.

The ICE design problem is multi-faceted and quite hard. The main innovation in our design is an expressive yet concise tree-based bidding language (which generalizes known languages such as XOR/OR [23]), and the tight coupling of this language with efficient algorithms for price-feedback to guide bidding, winner-determination to determine trades, and revealed-preference activity rules to ensure progress across rounds. The exchange is iterative: bidders express upper and lower valuations on trades by annotating their bid-tree, and then tighten these bounds in response to price feedback in each round. The *Threshold* payment rule, introduced by Parkes et al. [24], is used to determine final payments.

The exchange has a number of interesting theoretical properties. For instance, when there exist linear prices we establish soundness and completeness: for straightforward bidders that adjust their bounds to meet activity rules while keeping their true value within the bounds, the exchange will terminate with the efficient allocation. In addition, the

[*]Corresponding author. Remaining authors in alphabetical order. parkes@eecs.harvard.edu

[†]Division of Engineering and Applied Sciences, Harvard University, Cambridge MA 02138.

[1]http://www.eecs.harvard.edu/~parkes/cs286r/ice.html

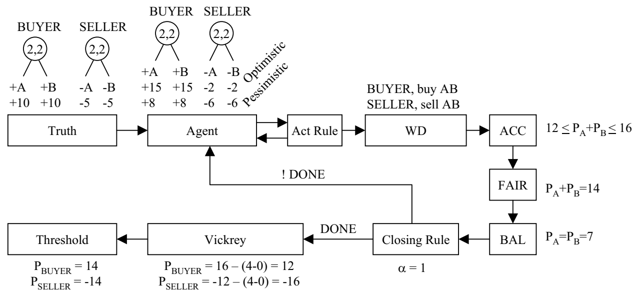

Figure 1: ICE System Flow of Control

efficient allocation can often be determined without bidders revealing, or even knowing, their exact value for all trades. This is essential in complex domains where the valuation problem can itself be very challenging for a participant [28]. While we cannot claim that straightforward bidding is an equilibrium of the exchange (and indeed, should not expect to by the Myerson-Satterthwaite impossibility theorem [22]), the Threshold payment rule minimizes the *ex post* incentive to manipulate across all budget-balanced payment rules.

The exchange is implemented in Java and is currently in validation. In describing the exchange we will first provide an overview of the main components and introduce several working examples. Then, we introduce the basic components for a simple one-shot variation in which bidders state their exact values for trades in a single round. We then describe the full iterative exchange, with upper and lower values, price-feedback, activity rules, and termination conditions. We state some theoretical properties of the exchange, and end with a discussion to motivate our main design decisions, and suggest some next steps.

2. AN OVERVIEW OF THE ICE DESIGN

The design has four main components, which we will introduce in order through the rest of the paper:

• **Expressive and concise tree-based bidding language.** The language describes values for *trades*, such as "my value for selling AB and buying C is \$100," or "my value for selling ABC is -\$50," with negative values indicating that a bidder must receive a payment for the trade to be acceptable. The language allows bidders to express upper and lower bounds on value, which can be tightened across rounds.
• **Winner Determination.** Winner-determination (WD) is formulated as a mixed-integer program (MIP), with the structure of the bid-trees captured explicitly in the formulation. Comparing the solution at upper and lower values allows for a determination to be made about termination, with progress in intermediate rounds driven by an intermediate valuation and the lower values adopted on termination.
• **Payments.** Payments are computed using the Threshold payment rule [24], with the intermediate valuations adopted in early rounds and lower values adopted on termination.
• **Price feedback.** An approximate price is computed for each item in the exchange in each round, in terms of the intermediate valuations and the provisional trade. The prices are optimized to approximate competitive equilibrium prices, and further optimized to best approximate the current Threshold payments with remaining ties broken to favor prices that are balanced across different items. In computing the prices, we adopt the methods of *constraint-generation* to exploit the structure of the bidding language and avoid enumerating all feasible trades. The subproblem to generate new constraints is a variation of the WD problem.
• **Activity rule.** A *revealed-preference* activity rule [1] ensures progress across rounds. In order to remain active, a bidder must tighten bounds so that there is enough information to define a trade that maximizes surplus at the current prices. Another variation on the WD problem is formulated, both to verify that the activity rule is met and also to provide feedback to a bidder to explain how to meet the rule.

An outline of the ICE system flow of control is provided in Figure 1. We will return to this example later in the paper. For now, just observe in this two-agent example that the agents state lower and upper bounds that are checked in the activity rule, and then passed to winner-determination (WD), and then through three stages of pricing (accuracy, fairness, balance). On passing the closing rule (in which parameters α^{eff} and α^{thresh} are checked for convergence of the trade and payments), the exchange goes to a last-and-final round. At the end of this round, the trade and payments are finally determined, based on the lower valuations.

2.1 Related Work

Many ascending-price one-sided CAs are known in the literature [10, 25, 30]. Direct elicitation approaches, in which agents respond to explicit queries about their valuations, have also been proposed for one-sided CAs [8, 14, 19] and for CEs with restricted expressiveness [29]. A number of ascending CAs are designed to work with simple prices on items [12, 17]. The price generation methods that we use in ICE generalize the methods in these earlier papers.

Parkes et al. [24] studied sealed-bid combinatorial exchanges and introduced the Threshold payment rule. Subsequently, Krych [16] demonstrated experimentally that the Threshold rule promotes efficient allocations. We are not aware of any previous studies of iterative CEs. Dominant strategy DAs are known for unit demand [20] and also for single-minded agents [2]. No dominant strategy mechanisms are known for the general CE problem.

ICE is a "hybrid" auction design, in that it couples simple item prices to drive bidding in early rounds with combinatorial WD and payments, a feature it shares with the *clock-proxy* design of Ausubel et al. [1] for one-sided CAs. We adopt a variation on the clock-proxy auctions's revealed-preference activity rule.

The bidding language shares some structural elements with the $\mathcal{L}_{GB}$ language of Boutilier and Hoos [7], but has very different semantics. Rothkopf et al. [27] also describe a restricted tree-based bidding language. 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 appear to provide useful "locality," so that the value of one component in a tree can be understood independently from the rest of the tree. The idea of capturing the structure of our bidding language explicitly within a mixed-integer programming formulation follows the developments in Boutilier [6].

Smith et al. [29] have previously studied iterative CEs, but can handle only limited expressiveness and adopt a direct-query based approach with an enumerative internal data structure that scales poorly to many items. A novel feature in their earlier design is *item discovery*, where the items available to trade need not be known in advance.

3. PRELIMINARIES

In our model, we consider a set of goods, indexed $\{1, \ldots, m\}$ and a set of bidders, indexed $\{1, \ldots, n\}$. The initial allocation of goods is denoted $x^0 = (x_1^0, \ldots, x_n^0)$, with $x_i^0 = (x_{i1}^0, \ldots, x_{im}^0)$ and $x_{ij}^0 \geq 0$ for good j indicating the number of units of good j held by bidder i. A trade $\lambda = (\lambda_1, \ldots, \lambda_n)$ denotes the change in allocation, with $\lambda_i = (\lambda_{i1}, \ldots, \lambda_{im})$ where $\lambda_{ij} \in \mathbb{Z}$ is the change in the number of units of item j to bidder i. So, the final allocation is $x^1 = x^0 + \lambda$.

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. Utility is quasi-linear, with $u_i(\lambda_i, p) = v_i(\lambda_i) - p$ for trade λ_i and payment $p \in \mathbb{R}$. Price p can be negative, indicating the bidder receives a payment for the trade. We use the term *payoff* interchangeably with *utility*.

Our goal in the ICE design is to implement the efficient trade. The efficient trade, λ^*, maximizes the total increase in value across bidders.

Definition 1 (Efficient trade). *The efficient trade λ^* solves*

$$\max_{(\lambda_1, \ldots, \lambda_n)} \sum_i v_i(\lambda_i)$$

$$\text{s.t.} \quad \lambda_{ij} + x_{ij}^0 \geq 0, \quad \forall i, \forall j \tag{1}$$

$$\sum_i \lambda_{ij} \leq 0, \quad \forall j \tag{2}$$

$$\lambda_{ij} \in \mathbb{Z} \tag{3}$$

Constraints (1) ensure that no agent sells more items than it has in its initial allocation. Constraints (2) provide free disposal, and allows feasible trades to sell more items than are purchased (but not vice versa).

Later, we adopt $Feas(x^0)$ to denote the set of *feasible trades*, given these constraints and given an initial allocation $x^0 = (x_1^0, \ldots, x_n^0)$.

3.1 Working Examples

In this section, we provide three simple examples of instances that we will use to illustrate various components of the exchange. All three examples have only one seller, but this is purely illustrative.

Example 1. *One seller and one buyer, two goods $\{A, B\}$, with the seller having an initial allocation of AB. Changes in values for trades:*

seller	*buyer*
AND($-A, -B$)	*AND($+A, +B$)*
-10	*+20*

The "AND" indicates that both the buyer and the seller are only interested in trading both goods as a bundle. Here, the efficient (value-maximizing) trade is for the seller to sell AB to the buyer, denoted $\lambda^* = ([-1, -1], [+1, +1])$.

Example 2. *One seller and four buyers, four goods $\{A, B, C, D\}$, with the seller having an initial allocation of $ABCD$. Changes in values for trades:*

seller	*buyer1*	*buyer 2*	*buyer 3*	*buyer 4*
OR($-A, -B, -C, -D$)	*AND($+A, +B$)*	*XOR($+A, +B$)*	*AND($+C, +D$)*	*XOR($+C, +D$)*
0	*+6*	*+4*	*+3*	*+2*

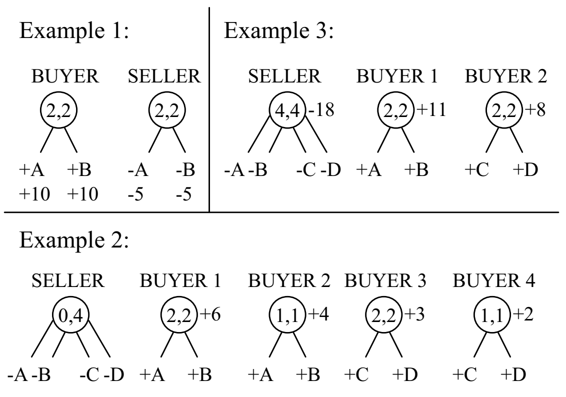

Figure 2: Example Bid Trees.

The "OR" indicates that the seller is willing to sell any number of goods. The "XOR" indicates that buyers 2 and 4 are willing to buy at most one of the two goods in which they are interested. The efficient trade is for bundle AB to go to buyer 1 and bundle CD to buyer 3, denoted $\lambda^* = ([-1, -1, -1, -1], [+1, +1, 0, 0], [0, 0, 0, 0], [0, 0, +1, +1], [0, 0, 0, 0])$.

Example 3. *One seller and two buyers, four goods $\{A, B, C, D\}$, with the seller having an initial allocation of $ABCD$. Changes in values for trades:*

seller	*buyer1*	*buyer 2*
AND($-A, -B, -C, -D$)	*AND($+A, +B$)*	*AND($+C, +D$)*
-18	*+11*	*+8*

The efficient trade is for bundle AB to go to buyer 1 and bundle CD to go to buyer 2, denoted $\lambda^* = ([-1, -1, -1, -1], [+1, +1, 0, 0], [0, 0, +1, +1])$.

4. A ONE-SHOT EXCHANGE DESIGN

The description of ICE is broken down into two sections: one-shot (sealed-bid) and iterative. In this section we abstract away the iterative aspect and introduce a specialization of the tree-based language that supports only exact values on nodes.

4.1 Tree-Based Bidding Language

The bidding language is designed to be expressive and concise, entirely symmetric with respect to buyers and sellers, and to extend to capture bids from mixed buyers and sellers, ranging from simple swaps to highly complex trades. Bids are expressed as annotated *bid trees*, and define a bidder's value for all possible trades.

The language defines changes in values on trades, with leaves annotated with traded items and nodes annotated with changes in values (either positive or negative). The main feature is that it has a general "interval-choose" logical operator on internal nodes, and that it defines careful semantics for propagating values within the tree. We illustrate the language on each of Examples 1–3 in Figure 2.

The language has a tree structure, with trades on items defined on leaves and values annotated on nodes and leaves. The nodes have zero values where no value is indicated. Internal nodes are also labeled with interval-choose (IC) ranges. Given a trade, the semantics of the language define which nodes in the tree can be satisfied, or "switched-on." First, if a child is on then its parent must be on. Second, if a parent node is on, then the number of children that are on must be within the IC range on the parent node. Finally, leaves in which the bidder is buying items can only be on if the items are provided in the trade.

For instance, in Example 2 we can consider the efficient trade, and observe that in this trade all nodes in the trees of buyers 1 and 3 (and also the seller), but none of the nodes in the trees of buyers 2 and 4, can be on. On the other hand, in the trade in which A goes to buyer 2 and D to buyer 4, then the root and appropriate leaf nodes can be on for buyers 2 and 4, but no nodes can be on for buyers 1 and 3. Given a trade there is often a number of ways to choose the set of satisfied nodes. The semantics of the language require that the nodes that maximize the summed value across satisfied nodes be activated.

Consider bid tree T_i from bidder i. This defines nodes $\beta \in T_i$, of which some are leaves, $\mathit{Leaf}(i) \subseteq T_i$. Let $\mathit{Child}(\beta) \subseteq T_i$ denote the children of a node β (that is not itself a leaf). All nodes except leaves are labeled with the *interval-choose* operator $[IC_i^x(\beta), IC_i^y(\beta)]$. Every node is also labeled with a *value*, $v_{i\beta} \in \mathbb{R}$. Each leaf β is labeled with a *trade*, $q_{i\beta} \in \mathbb{Z}^m$ (i.e., leaves can define a bundled trade on more than one type of item.)

Given a trade λ_i to bidder i, the interval-choose operators and trades on leaves define which nodes can be *satisfied*. There will often be a choice. Ties are broken to maximize value. Let $sat_{i\beta} \in \{0,1\}$ denote whether node β is satisfied. Solution sat_i is valid given tree T_i and trade λ_i, written $sat_i \in \mathit{valid}(T_i, \lambda_i)$, if and only if:

$$\sum_{\beta \in \mathit{Leaf}(i)} q_{i\beta j} \cdot sat_{i\beta} \le \lambda_{ij}, \quad \forall i,\ \forall j \tag{4}$$

$$IC_i^x(\beta) sat_{i\beta} \le \sum_{\beta' \in \mathit{Child}(\beta)} sat_{i\beta'} \le IC_i^y(\beta) sat_{i\beta}, \forall \beta \notin \mathit{Leaf}(i) \tag{5}$$

In words, a set of leaves can only be considered satisfied given trade λ_i if the total increase in quantity summed across all such leaves is covered by the trade, for all goods (Eq. 4). 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. We also need "upwards-propagation": any time a node other than the root is satisfied then its parent must be satisfied (by $\sum_{\beta' \in \mathit{Child}(\beta)} sat_{i\beta'} \le IC_i^y(\beta) sat_{i\beta}$ in Eq. 5). Finally, we need "downwards-propagation": any time an internal node is satisfied then the appropriate number of children must also be satisfied (Eq. 5). The total value of trade λ_i, given bid-tree T_i, is defined as:

$$v_i(T_i, \lambda_i) = \max_{sat \in \mathit{valid}(T_i, \lambda_i)} \sum_{\beta \in T} v_\beta \cdot sat_\beta \tag{6}$$

The tree-based language generalizes existing languages. For instance: $IC(2,2)$ on a node with 2 children is equivalent to an AND operator; $IC(1,3)$ on a node with 3 children is equivalent to an OR operator; and $IC(1,1)$ on a node with 2 children is equivalent to an XOR operator. Similarly, the XOR/OR bidding languages can be directly expressed as a bid tree in our language.[2]

[2]The OR* language is the OR language with dummy items to provide additional structure. OR* is known to be expressive and concise. However, it is not known whether OR* dominates XOR/OR in terms of conciseness [23].

4.2 Winner Determination

This section defines the winner determination problem, which is formulated as a MIP and solved in our implementation with a commercial solver.[3] The solver uses branch-and-bound search with dynamic cut generation and branching heuristics to solve large MIPs in economically feasible run times.

In defining the MIP representation we are careful to avoid an XOR-based enumeration of all bundles. A variation on the WD problem is reused many times within the exchange, e.g. for column generation in pricing and for checking revealed preference.

Given bid trees $T = (T_1, \ldots, T_n)$ and initial allocation x^0, the mixed-integer formulation for WD is:

$$\begin{aligned} \mathrm{WD}(T, x^0): \ & \max_{\lambda, sat} \sum_i \sum_{\beta \in T_i} v_{i\beta} \cdot sat_{i\beta} \\ \text{s.t.} \quad & (1), (2), sat_{i\beta} \in \{0,1\}, \lambda_{ij} \in \mathbb{Z} \\ & sat_i \in \mathit{valid}(T_i, \lambda_i), \quad \forall i \end{aligned}$$

Some goods may go unassigned because free disposal is allowed within the clearing rules of winner determination. These items can be allocated back to agents that sold the items, i.e. for which $\lambda_{ij} < 0$.

4.3 Computing Threshold Payments

The Threshold payment rule is based on the payments in the Vickrey-Clarke-Groves (VCG) mechanism [15], which itself is truthful and efficient but does not satisfy budget balance. Budget-balance requires that the total payments to the exchange are equal to the total payments made by the exchange. In VCG, the payment paid by agent i is

$$p_{\mathrm{vcg},i} = \hat{v}(\lambda_i^*) - (V^* - V_{-i}) \tag{7}$$

where λ^* is the efficient trade, V^* is the reported value of this trade, and V_{-i} is the reported value of the efficient trade that would be implemented without bidder i. We call $\Delta_{\mathrm{vcg},i} = V^* - V_{-i}$ the *VCG discount.* For instance, in Example 1 $p_{\mathrm{vcg,seller}} = -10 - (+10 - 0) = -20$ and $p_{\mathrm{vcg,buyer}} = +20 - (+10 - 0) = 10$, and the exchange would run at a budget deficit of $-20 + 10 = -10$.

The Threshold payment rule [24] determines budget-balanced payments to minimize the maximal error across all agents to the VCG outcome.

DEFINITION 2. *The Threshold payment scheme implements the efficient trade λ^* given bids, and sets payments $p_{\mathrm{thresh},i} = \hat{v}_i(\lambda_i^*) - \Delta_i$, where $\Delta = (\Delta_1, \ldots, \Delta_n)$ is set to minimize $\max_i(\Delta_{\mathrm{vcg},i} - \Delta_i)$ subject to $\Delta_i \le \Delta_{\mathrm{vcg},i}$ and $\sum_i \Delta_i \le V^*$ (this gives budget-balance).*

EXAMPLE 4. *In Example 2, the VCG discounts are $(9, 2, 0, 1, 0)$ to the seller and four buyers respectively, VCG payments are $(-9, 4, 0, 2, 0)$ and the exchange runs at a deficit of -3. In Threshold, the discounts are $(8, 1, 0, 0, 0)$ and the payments are $(-8, 5, 0, 3, 0)$. This minimizes the worst-case error to VCG discounts across all budget-balanced payment schemes.*

Threshold payments are designed to minimize the maximal *ex post* incentive to manipulate. Krych [16] confirmed that Threshold promotes allocative efficiency in restricted and approximate Bayes-Nash equilibrium.

[3]CPLEX, www.ilog.com

5. THE ICE DESIGN

We are now ready to introduce the iterative combinatorial exchange (ICE) design. Several new components are introduced, relative to the design for the one-shot exchange. Rather than provide precise valuations, bidders can provide lower and upper valuations and revise this bid information across rounds. The exchange provides price-based feedback to guide bidders in this process, and terminates with an efficient (or approximately-efficient) trade with respect to reported valuations.

In each round $t \in \{0, 1, \ldots\}$ the current lower and upper bounds, $\underline{v}^t$ and $\overline{v}^t$, are used to define a *provisional valuation* profile v^α (the *α-valuation*), together with a provisional trade λ^t and provisional prices $p^t = (p_1^t, \ldots, p_m^t)$ on items. The α-valuation is a linear combination of the current upper and lower valuations, with $\alpha^{\mathrm{EFF}} \in [0,1]$ chosen endogenously based on the "closeness" of the optimistic trade (at $\overline{v}$) and the pessimistic trade (at $\underline{v}$). Prices p^t are used to inform an activity rule, and drive progress towards an efficient trade.

5.1 Upper and Lower Valuations

The bidding language is extended to allow a bidder i to report a lower and upper value $(\underline{v}_{i\beta}, \overline{v}_{i\beta})$ on each node. These take the place of the exact value $v_{i\beta}$ defined in Section 4.1. Based on these labels, we can define the valuation functions $\underline{v}_i(T_i, \lambda_i)$ and $\overline{v}_i(T_i, \lambda_i)$, using the exact same semantics as in Eq. (6). We say that such a bid-tree is *well-formed* if $\underline{v}_{i\beta} \leq \overline{v}_{i\beta}$ for all nodes. The following lemma is useful:

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

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

5.2 Price Feedback

In each round, approximate competitive-equilibrium (CE) prices, $p^t = (p_1^t, \ldots, p_m^t)$, are determined. Given these provisional prices, the price on trade λ_i for bidder i is $p^t(\lambda_i) = \sum_{j \leq m} p_j^t \cdot \lambda_{ij}$.

DEFINITION 3 (CE PRICES). *Prices p^* are competitive equilibrium prices if the efficient trade λ^* is supported at prices p^*, so that for each bidder:*

$$\lambda_i^* \in \arg \max_{\lambda \in Feas(x^0)} \{v_i(\lambda_i) - p^*(\lambda_i)\} \tag{8}$$

CE prices will not always exist and we will often need to compute approximate prices [5]. We extend ideas due to Rassenti et al. [26], Kwasnica et al. [17] and Dunford et al. [12], and select approximate prices as follows:

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 the prices in driving progress across rounds.

In computing prices, we explain how to compute approximate (or otherwise) prices for structured bidding languages, and without enumerating all possible trades. For this, we adopt constraint generation to efficient handle an exponential number of constraints. Each step is described in detail below.

I: Accuracy. We adopt a definition of price accuracy that generalizes the notions adopted in previous papers for unstructured bidding languages. Let λ^t denote the current provisional trade and suppose the provisional valuation is v^α. To compute accurate CE prices, we consider:

$$\min_{p,\delta} \ \delta \tag{9}$$

$$\text{s.t.} \quad v_i^\alpha(\lambda) - p(\lambda) \leq v_i^\alpha(\lambda_i^t) - p(\lambda_i^t) + \delta, \quad \forall i, \forall \lambda \tag{10}$$

$$\delta \geq 0, p_j \geq 0, \quad \forall j.$$

This linear program (LP) is designed to find prices that minimize the worst-case error across all agents.

From the definition of CE prices, it follows that CE prices would have $\delta = 0$ as a solution to (9), at which point trade λ_i^t would be in the best-response set of every agent (with $\lambda_i^t = \emptyset$, i.e. no trade, for all agents with no surplus for trade at the prices.)

EXAMPLE 5. *We can illustrate the formulation (9) on Example 2, assuming for simplicity that $v^\alpha = v$ (i.e. truth). The efficient trade allocates AB to buyer 1 and CD to buyer 3. Accuracy will seek prices $p(A), p(B), p(C)$ and $p(D)$ to minimize the $\delta \geq 0$ required to satisfy constraints:*

$$\begin{aligned} p(A)+p(B)+p(C)+p(D) &\geq 0 && \text{(seller)}\\ p(A)+p(B) &\leq 6+\delta && \text{(buyer 1)}\\ p(A)+\delta \geq 4, \quad p(B)+\delta &\geq 4 && \text{(buyer 2)}\\ p(C)+p(D) &\leq 3 && \text{(buyer 3)}\\ p(C)+\delta \geq 2, \quad p(D)+\delta &\geq 2 && \text{(buyer 4)} \end{aligned}$$

An optimal solution requires $p(A) = p(B) = 10/3$, with $\delta = 2/3$, with $p(C)$ and $p(D)$ taking values such as $p(C) = p(D) = 3/2$.

But, (9) has an exponential number of constraints (Eq. 10). Rather than solve it explicitly we use constraint generation [4] and dynamically generate a sufficient subset of constraints. Let $\mathbb{F}_i$ denote a manageable subset of all possible feasible trades to bidder i. Then, a relaxed version of (9) (written ACC) is formulated by substituting (10) with

$$v_i^\alpha(\lambda) - p(\lambda) \leq v_i^\alpha(\lambda_i^t) - p(\lambda_i^t) + \delta, \ \forall i, \forall \lambda \in \mathbb{F}_i, \tag{11}$$

where $\mathbb{F}_i$ is a set of trades that are feasible for bidder i given the other bids. Fixing the prices p^*, we then solve n subproblems (one for each bidder),

$$\max_{\lambda} \quad v_i^\alpha(\lambda_i) - p^*(\lambda_i) \qquad \text{[R-WD}(i)\text{]}$$

$$\text{s.t.} \quad \lambda \in Feas(x^0), \tag{12}$$

to check whether solution (p^*, δ^*) to ACC is feasible in problem (9). In R-WD(i) the objective is to determine a most preferred trade for each bidder at these prices. Let $\hat{\lambda}_i$ denote the solution to R-WD(i). Check condition:

$$v_i^\alpha(\hat{\lambda}_i) - p^*(\hat{\lambda}) \leq v_i^\alpha(\lambda_i^t) - p^*(\lambda_i^t) + \delta^*, \qquad (13)$$

and if this condition holds for all bidders i, then solution (p^*, δ^*) is optimal for problem (9). Otherwise, trade $\hat{\lambda}_i$ is added to $\mathbb{F}_i$ for all bidders i for which this constraint is violated and we re-solve the LP with the new set of constraints.[4]

II: Fairness. Second, we break remaining ties to prefer fair prices: choosing prices that minimize the worst-case error with respect to Threshold payoffs (i.e. utility to bidders with Threshold payments), but without choosing prices that are less accurate.[5]

EXAMPLE 6. *For example, accuracy in Example 1 (depicted in Figure 1) requires $12 \leq p_A + p_B \leq 16$ (for $v^\alpha = \underline{v}$). At these valuations the Threshold payoffs would be 2 to both the seller and the buyer. This can be exactly achieved in pricing with $p_A + p_B = 14$.*

The fairness tie-breaking method is formulated as the following LP:

$$\min_{p,\pi} \quad \pi \qquad \text{[FAIR]}$$

$$\text{s.t.} \quad v_i^\alpha(\lambda) - p(\lambda) \leq v_i^\alpha(\lambda_i^t) - p(\lambda_i^t) + \delta_i^*, \forall i, \forall \lambda \in \mathbb{F}_i \quad (14)$$

$$\pi \geq \pi_{\text{vcg},i} - (v_i^\alpha(\lambda_i^t) - p(\lambda_i^t)), \quad \forall i \quad (15)$$

$$\pi \geq 0, p_j \geq 0, \quad \forall j,$$

where δ^* represents the error in the optimal solution, from ACC. The objective here is the same as in the Threshold payment rule (see Section 4.3): minimize the maximal error between bidder payoff (at v^α) for the provisional trade and the VCG payoff (at v^α). Problem FAIR is also solved through constraint generation, using R-WD(i) to add additional violated constraints as necessary.

III: Balance. Third, we break remaining ties to prefer balanced prices: choosing prices that minimize the maximal price across all items. Returning again to Example 1, depicted in Figure 1, we see that accuracy and fairness require $p(A) + p(B) = 14$. Finally, balance sets $p(A) = p(B) = 7$. Balance is justified when, all else being equal, items are more likely to have similar than dissimilar values.[6] The LP for balance is formulated as follows:

$$\min_{p,Y} \quad Y \qquad \text{[BAL]}$$

$$\text{s.t.} \quad v_i^\alpha(\lambda) - p(\lambda) \leq v_i^\alpha(\lambda_i^t) - p(\lambda_i^t) + \delta_i^*, \forall i, \forall \lambda \in \mathbb{F}_i \quad (16)$$

$$\pi_i^* \geq \pi_{\text{vcg},i} - (v_i^\alpha(\lambda_i^t) - p(\lambda_i^t)), \forall i, \quad (17)$$

$$Y \geq p_j, \quad \forall j \quad (18)$$

$$Y \geq 0, p_j \geq 0, \quad \forall j,$$

where δ^* represents the error in the optimal solution from ACC and π^* represents the error in the optimal solution from FAIR. Constraint generation is also used to solve BAL, generating new trades for $\mathbb{F}_i$ as necessary.

Comment 1: Lexicographical Refinement. For all three sub-problems we also perform lexicographical refinement (with respect to bidders in ACC and FAIR, and with respect to goods in BAL). 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 a constraint (11) is binding. For such a bidder i, the constraint is replaced with

$$v_i^\alpha(\lambda) - p(\lambda) \leq v_i^\alpha(\lambda_i^t) - p(\lambda_i^t) + \delta_i^*, \forall \lambda \in \mathbb{F}_i, \qquad (19)$$

and the error to bidder i no longer appears explicitly in the objective. ACC is then re-solved, and makes progress by further minimizing the maximal error across all bidders yet to be pinned down. This continues, pinning down any new bidders for whom one of constraints (11) is binding, until the error is lexicographically optimized for all bidders.[7] The exact same process is repeated for FAIR and BAL, with bidders pinned down and constraints (15) replaced with $\pi_i^* \geq \pi_{\text{vcg},i} - (v_i^\alpha(\lambda_i^t) - p(\lambda_i^t)), \forall \lambda \in \mathbb{F}_i$, (where π_i^* is the current objective) in FAIR, and items pinned down and constraints (18) replaced with $p_j^* \geq p_j$ (where p_j^* represents the target for the maximal price on that item) in BAL.

Comment 2: Computation. All constraints in $\mathbb{F}_i$ are retained, and this set grows across all stages and across all rounds of the exchange. Thus, the computational effort in constraint generation is re-used. In implementation we are careful to address a number of "ϵ-issues" that arise due to floating-point issues. We prefer to err on the side of being conservative in determining whether or not to add another constraint in performing check (13). This avoids later infeasibility issues. In addition, when pinning-down bidders for the purpose of lexicographical refinement we relax the associated bidder-constraints with a small $\epsilon > 0$ on the right-hand side.

5.3 Revealed-Preference Activity Rules

The role of activity rules in the auction is to ensure both consistency and progress across rounds [21]. Consistency in our exchange requires that bidders tighten bounds as the exchange progresses. Activity rules ensure that bidders are active during early rounds, and promote useful elicitation throughout the exchange.

We adopt a simple *revealed-preference* (RP) activity rule. The idea is loosely based around the RP-rule in Ausubel et al. [1], where it is used for one-sided CAs. The motivation is to require more than simply consistency: we need bidders to provide enough information for the system to be able to to *prove* that an allocation is (approximately) efficient.

It is helpful to think about the bidders interacting with "proxy agents" that will act on their behalf in responding to provisional prices p^{t-1} determined at the end of round $t-1$. The only knowledge that such a proxy has of the

[4]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 agent i. Thus, the structure of the bid-tree language is exploited in generating new constraints, because this is solved as a concise MIP. The other bidders are kept around in the MIP (but do not appear in the objective), and are used to define the space of feasible trades.

[5]The methods of Dunford et al. [12], that use a nucleolus approach, are also closely related.

[6]The use of *balance* was advocated by Kwasnica et al. [17]. Dunford et al. [12] prefer to smooth prices across rounds.

[7]For example, applying this to accuracy on Example 2 we solve once and find bidders 1 and 2 are binding, for error $\delta^* = 2/3$. We pin these down and then minimize the error to bidders 3 and 4. Finally, this gives $p(A) = p(B) = 10/3$ and $p(C) = p(D) = 5/3$, with accuracy 2/3 to bidders 1 and 2 and 1/3 to bidders 3 and 4.

valuation of a bidder is through the bid-tree. Suppose a proxy was queried by the exchange and asked which trade the bidder was most interested in at the provisional prices. The RP rule says the following: *the proxy must have enough information to be able to determine this surplus-maximizing trade at current prices.* Consider the following examples:

EXAMPLE 7. *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. The RP rule is satisfied because the proxy knows that however the remaining value uncertainty on $+B$ is resolved the bidder will always (weakly) prefer $+B$ to $+A$.*

EXAMPLE 8. *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. The RP rule is satisfied because the bidder will always prefer $+A$ to $+B$ at equal prices, whichever way the uncertain value on the root node is ultimately resolved.*

Overloading notation, let $v_i \in T_i$ denote a valuation that is consistent with lower and upper valuations in bid tree T_i.

DEFINITION 4. *Bid tree T_i satisfies RP at prices p^{t-1} if and only if there exists some feasible trade L^* for which,*

$$v_i(L_i^*) - p^{t-1}(L_i^*) \geq \max_{\lambda \in Feas(x^0)} v_i(\lambda_i) - p^{t-1}(\lambda_i), \quad \forall v_i \in T_i. \tag{20}$$

To make this determination for bidder i we solve a sequence of problems, each of which is a variation on the WD problem. First, we construct a candidate lower-bound trade, which is a feasible trade that solves:

$$\max_{\lambda} \quad \underline{v}_i(\lambda_i) - p^{t-1}(\lambda_i) \qquad [\text{RP1}(i)]$$

$$\text{s.t.} \quad \lambda \in Feas(x^0), \tag{21}$$

The solution π_l^* to RP1(i) represents the maximal payoff that bidder i can achieve across all feasible trades, given its pessimistic valuation.

Second, we break ties to find a trade with maximal value uncertainty across all possible solutions to RP1(i):

$$\max_{\lambda} \quad \overline{v}_i(\lambda_i) - \underline{v}_i(\lambda_i) \qquad [\text{RP2}(i)]$$

$$\text{s.t.} \quad \lambda \in Feas(x^0) \tag{22}$$

$$\underline{v}_i(\lambda_i) - p^{t-1}(\lambda_i) \geq \pi_l^* \tag{23}$$

We adopt solution L_i^* as our candidate for the trade that may satisfy RP. To understand the importance of this tie-breaking rule consider Example 7. The proxy can prove $+B$ but not $+A$ is a best-response for all $v_i \in T_i$, and should choose $+B$ as its candidate. Notice that $+B$ is a counterexample to $+A$, but not the other way round.

Now, we construct a modified valuation $\tilde{v}_i$, by setting

$$\tilde{v}_{i\beta} = \begin{cases} \underline{v}_{i\beta} & \text{, if } \beta \in sat(L_i^*) \\ \overline{v}_{i\beta} & \text{, otherwise.} \end{cases} \tag{24}$$

where $sat(L_i^*)$ is the set of nodes that are satisfied in the lower-bound tree for trade L_i^*. Given this modified valuation, we find U^* to solve:

$$\max_{\lambda} \quad \tilde{v}_i(\lambda_i) - p^{t-1}(\lambda_i) \qquad [\text{RP3}(i)]$$

$$\text{s.t.} \quad \lambda \in Feas(x^0) \tag{25}$$

Let π_u^* denote the payoff from this optimal trade at modified values $\tilde{v}$. We call trade U_i^* the *witness trade.* We show in Proposition 1 that the RP rule is satisfied if and only if $\pi_l^* \geq \pi_u^*$.

Constructing the modified valuation as $\tilde{v}_i$ recognizes that there is "shared uncertainty" across trades that satisfy the same nodes in a bid tree. Example 8 helps to illustrate this. Just using $\overline{v}_i$ in RP3(i), we would find L_i^* is "buy A" with payoff $\pi_l^* = 3$ but then find U_i^* is "buy B" with $\pi_u^* = 7$ and fail RP. We must recognize that *however the uncertainty on the root node is resolved it will affect $+A$ and $+B$ in exactly the same way.* For this reason, we set $\tilde{v}_{i\beta} = \underline{v}_{i\beta} = 5$ on the root node, which is exactly the same value that was adopted in determining π_l^*. Then, RP3(i) applied to U_i^* gives "buy A" and the RP test is judged to be passed.

PROPOSITION 1. *Bid tree T_i satisfies RP given prices p^{t-1} if and only if any lower-bound trade L_i^* that solves RP1(i) and RP2(i) satisfies:*

$$\underline{v}_i(T_i, L_i^*) - p^{t-1}(L_i^*) \geq \tilde{v}_i(T_i, U_i^*) - p^{t-1}(U_i^*), \tag{26}$$

where $\tilde{v}_i$ is the modified valuation in Eq. (24).

PROOF. For sufficiency, notice that the difference in payoff between trade L_i^* and another trade λ_i is unaffected by the way uncertainty is resolved on any node that is satisfied in both L_i^* and λ_i. Fixing the values in $\tilde{v}_i$ on nodes satisfied in L_i^* has the effect of removing this consideration when a trade U_i^* is selected that satisfies one of these nodes. On the other hand, fixing the values on these nodes has no effect on trades considered in RP3(i) that do not share a node with L_i^*. For the necessary direction, we first show that any trade that satisfies RP must solve RP1(i). Suppose otherwise, that some λ_i with payoff greater than π_l^* satisfies RP. But, valuation $\underline{v}_i \in T_i$ together with L_i^* presents a counterexample to RP (Eq. 20). Now, suppose (for contradiction) that some λ_i with maximal payoff π_l^* but uncertainty less than L_i^* satisfies RP. Proceed by case analysis. Case a): only one solution to RP1(i) has uncertain value and so λ_i has certain value. But, this cannot satisfy RP because L_i^* with uncertain value would be a counterexample to RP (Eq. 20). Case b): two or more solutions to RP1(i) have uncertain value. Here, we first argue that one of these trades must satisfy a (weak) superset of all the nodes with uncertain value that are satisfied by all other trades in this set. This is by RP. Without this, then for any choice of trade that solves RP1(i), there is another trade with a disjoint set of uncertain but satisfied nodes that provides a counterexample to RP (Eq. 20). Now, consider the case that some trade contains a superset of all the uncertain satisfied nodes of the other trades. Clearly RP2(i) will choose this trade, L_i^*, and λ_i must satisfy a subset of these nodes (by assumption). But, we now see that λ_i cannot satisfy RP because L_i^* would be a counterexample to RP. $\square$

Failure to meet the activity rule must have some consequence. In the current rules, the default action we choose is to set the upper bounds in valuations down to the maximal value of the provisional price on a node[8] and the lower-

[8]The provisional price on a node is defined as the minimal total price across all feasible trades for which the subtree rooted at the tree is satisfied.

bound value on that node.[9] Such a bidder can remain active within the exchange, but only with valuations that are consistent with these new bounds.

5.4 Bidder Feedback

In each round, our default design provides every bidder with the provisional trade and also with the current provisional prices. See 7 for an additional discussion. We also provide guidance to help a bidder meet the RP rule. Let $sat(L_i^*)$ and $sat(U_i^*)$ denote the nodes that are satisfied in trades L_i^* and U_i^*, as computed in RP1–RP3.

LEMMA 2. *When RP fails, a bidder must increase a lower bound on at least one node in $sat(L_i^*) \setminus sat(U_i^*)$ or decrease an upper bound on at least one node in $sat(U_i^*) \setminus sat(L_i^*)$ in order to meet the activity rule.*

PROOF. Changing the upper- or lower- values on nodes that are not satisfied by either trade does not change L_i^* or U_i^*, and does not change the payoff from these trades. Thus, the RP condition will continue to fail. Similarly, changing the bounds on nodes that are satisfied in both trades has no effect on revealed preference. A change to a lower bound on a shared node affects both L_i^* and U_i^* identically because of the use of the modified valuation to determine U_i^*. A change to an upper bound on a shared node has no effect in determining either L_i^* or U_i^*. $\square$

Note that when $sat(U_i^*) = sat(L_i^*)$ then condition (26) is always trivially satisfied, and so the guidance in the lemma is always well-defined when RP fails. This is an elegant feedback mechanism because it is adaptive. Once a bidder makes some changes on some subset of these nodes, the bidder can query the exchange. The exchange can then respond "yes," or can revise the set of nodes $sat(\lambda_l^*)$ and $sat(\lambda_u^*)$ as necessary.

5.5 Termination Conditions

Once each bidder has committed its new bids (and either met the RP rule or suffered the penalty) then round t closes. At this point, the task is to determine the new α-valuation, and in turn the provisional allocation λ^t and provisional prices p^t. A termination condition is also checked, to determine whether to move the exchange to a last-and-final round. To define the α-valuation we compute the following two quantities:

Pessimistic at Pessimistic (PP) Determine an efficient trade, λ_l^*, at pessimistic values, i.e. to solve $\max_\lambda \sum_i \underline{v}_i(\lambda_i)$, and set PP=$\sum_i \underline{v}_i(\lambda_{li}^*)$.

Pessimistic at Optimistic (PO) Determine an efficient trade, λ_u^*, at optimistic values, i.e. to solve $\max_\lambda \sum_i \overline{v}_i(\lambda_i)$, and set PO=$\sum_i \underline{v}_i(\lambda_{ui}^*)$.

First, note that $PP \geq PO$ and $PP \geq 0$ by definition, for all bid-trees, although PO can be negative (because the "right" trade at $\overline{v}$ is not currently a useful trade at $\underline{v}$). Recognizing this, define

$$\gamma^{\text{eff}}(PP, PO) = 1 + \frac{PP - PO}{PP}, \tag{27}$$

when $PP > 0$, and observe that $\gamma^{\text{eff}}(PP, PO) \geq 1$ when this is defined, and that $\gamma^{\text{eff}}(PP, PO)$ will start large and then trend towards 1 as the optimistic allocation converges towards the pessimistic allocation. In each round, we define $\alpha^{\text{eff}} \in [0, 1]$ as:

$$\alpha^{\text{eff}} = \begin{cases} 0 & \text{when PP is 0} \\ 1/\gamma^{\text{eff}} & \text{otherwise} \end{cases} \tag{28}$$

which is 0 while PP is 0 and then trends towards 1 once PP> 0 in some round. This is used to define α-valuation

$$v_i^\alpha = \alpha^{\text{eff}} \underline{v}_i + (1 - \alpha^{\text{eff}})\overline{v}_i, \forall i, \tag{29}$$

which is used to define the provisional allocation and provisional prices. The effect is to endogenously define a schedule for moving from optimistic to pessimistic values across rounds, based on how "close" the trades are to one another.
Termination Condition. In moving to the last-and-final round, and finally closing, we also care about the convergence of payments, in addition to the convergence towards an efficient trade. For this we introduce another parameter, $\alpha^{\text{thresh}} \in [0, 1]$, that trends from 0 to 1 as the Threshold payments at lower and upper valuations converge. Consider the following parameter:

$$\gamma^{\text{thresh}} = 1 + \frac{||p_{\text{thresh}}(\overline{v}) - p_{\text{thresh}}(\underline{v})||_2}{(PP/N_{\text{active}})}, \tag{30}$$

which is defined for $PP > 0$, where $p_{\text{thresh}}(v)$ denotes the Threshold payments at valuation profile v, N_{active} is the number of bidders that are actively engaged in trade in the PP trade, and $|| \cdot ||_2$ is the L_2-norm. Note that γ^{thresh} is defined for *payments* and not *payoffs*. This is appropriate because it is the accuracy of the outcome of the exchange that matters: i.e. the trade and the payments. Given this, we define

$$\alpha^{\text{thresh}} = \begin{cases} 0 & \text{when PP is 0} \\ 1/\gamma^{\text{thresh}} & \text{otherwise} \end{cases} \tag{31}$$

which is 0 while PP is 0 and then trends towards 1 as progress is made.

DEFINITION 5 (TERMINATION). *ICE transitions to a last-and-final round when one of the following holds:*

1. *$\alpha^{\text{eff}} \geq CUTOFF_{\text{eff}}$ and $\alpha^{\text{thresh}} \geq CUTOFF_{\text{thresh}}$,*
2. *there is no trade at the optimistic values,*

where $CUTOFF_{\text{eff}}, CUTOFF_{\text{thresh}} \in (0, 1]$ determine the accuracy required for termination.

At the end of the last-and-final round $v^\alpha = \underline{v}$ is used to define the final trade and the final Threshold payments.

EXAMPLE 9. *Consider again Example 1, and consider the upper and lower bounds as depicted in Figure 1. First, if the seller's bounds were $[-20, -4]$ then there is an optimistic trade but no pessimistic trade, and $PO = -4$ and $PP = 0$, and $\alpha^{\text{eff}} = 0$. At the bounds depicted, both the optimistic and the pessimistic trades occur and $PO = PP = 4$ and $\alpha^{\text{eff}} = 1$. However, we can see the Threshold payments are $(17, -17)$ at $\overline{v}$ but $(14, -14)$ at $\underline{v}$. Evaluating γ^{thresh}, we have $\gamma^{\text{thresh}} = 1 + \frac{\sqrt{1/2(3^2+3^2)}}{(4/2)} = 5/2$, and $\alpha^{\text{thresh}} = 2/5$. For $CUTOFF_{\text{thresh}} < 2/5$ the exchange would remain open. On the other hand, if the buyer's value for $+AB$ was between $[18, 24]$ and the seller's value for $-AB$ was between $[-12, -6]$, the Threshold payments are $(15, -15)$ at both upper and lower bounds, and $\alpha^{\text{thresh}} = 1$.*

[9]This is entirely analogous to when a bidder in an ascending clock auction stops bidding at a price: she is not permitted to bid at a higher price again in future rounds.

Component	Purpose	Lines
Agent.	Captures strategic behavior and information revelation decisions	762
Model Support	Provides XML support to load goods and valuations into world	200
World	Keeps track of all agent, good, and valuation details	998
Exchange Driver & Communication	Controls exchange, and coordinates remote agent behavior	585
Bidding Language	Implements the tree-based bidding language	1119
Activity Rule Engine	Implements the revealed preference rule with range support	203
Closing Rule Engine	Checks if auction termination condition reached	137
WD Engine	Provides WD-related logic	377
Pricing Engine	Provides Pricing-related logic	460
MIP Builders	Translates logic used by engines into our general optimizer formulation	346
Pricing Builders	Used by three pricing stages	256
Winner Determination Builders	Used by WD, activity rule, closing rule, and pricing constraint generation	365
Framework	Support code; eases modular replacement of above components	510

Table 1: Exchange Component and Code Breakdown.

6. SYSTEMS INFRASTRUCTURE

ICE is approximately 6502 lines of Java code, broken up into the functional packages described in Table 1.[10]

The prototype is modular so that researchers may easily replace components for experimentation. In addition to the core exchange discussed in this paper, we have developed an agent component that allows a user to simulate the behavior and knowledge of other players in the system, better allowing a user to formulate their strategy in advance of actual play. A user specifies a valuation model in an XML-interpretation of our bidding language, which is revealed to the exchange via the agent's strategy.

Major exchange tasks are handled by "engines" that dictate the non-optimizer specific logic. These engines drive the appropriate MIP/LP "builders". We realized that all of our optimization formulations boil down to two classes of optimization problem. The first, used by winner determination, activity rule, 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.

We have generalized our back-end optimization solver interface[11] (we currently support CPLEX and the LGPL- licensed LPSolve), and can take advantage of the load-balancing and parallel MIP/LP solving capability that this library provides.

7. DISCUSSION

The bidding language was defined to allow for perfect symmetry between buyers and sellers and provide expressiveness in an exchange domain, for instance for mixed bidders interested in executing trades such as swaps. This proved especially challenging. The breakthrough came when we focused on *changes in value for trades* rather than providing *absolute values for allocations*. For simplicity, we require the same tree structure for both the upper and lower valuations. This allows the language itself to ensure consistency (with the upper value at least the lower value on all trades) and enforce monotonic tightening of these bounds for all trades across rounds. It also provides for an efficient method to check the RP activity rule, because it makes it simple to reason about "shared uncertainty" between trades.

The decision to adopt a "direct and proxied" approach in which bidders express their upper and lower values to a trusted proxy agent that interacts with the exchange was made early in the design process. In many ways this is the clearest and most immediate way to generalize the design in Parkes et al. [24] and make it iterative. In addition, this removes much opportunity for strategic manipulation: bidders are restricted to making (incremental) statements about their valuations. Another advantage is that it makes the activity rule easy to explain: bidders can always meet the activity rule by tightening bounds such that their true value remains in the support.[12] Perhaps most importantly, having explicit information on upper and lower values permits progress in early rounds, even while there is no efficient trade at pessimistic values.

Upper and lower bound information also provides guidance about when to terminate. Note that taken by itself, $PP = PO$ does not imply that the current provisional trade is efficient with respect to all values consistent with current value information. The difference in values between different trades, aggregated across all bidders, could be similar at lower and upper bounds but quite different at intermediate values (including truth). Nevertheless, we conjecture that $PP = PO$ will prove an excellent indicator of efficiency in practical settings where the "shape" of the upper and lower valuations does convey useful information. This is worthy of experimental investigation. Moreover, the use of price and RP activity provides additional guarantees.

We adopted linear prices (prices on individual items) rather than non-linear prices (with prices on a trade not equal to the sum of the prices on the component items) early in the design process. The conciseness of this price representation is very important for computational tractability within the exchange and also to promote simplicity and transparency for bidders. The RP activity rule was adopted later, and is a good choice because of its excellent theoretical properties when coupled with CE prices. The following can be easily established: *given exact CE prices* p^{t-1} *for provisional trade*

[10] Code size is measured in physical source line of code (SLOC), as generated using David A. Wheeler's SLOC Count. The total of 6502 includes 184 for instrumentation (not shown in the table). The JOpt solver interface is another 1964 lines, and Castor automatically generates around 5200 lines of code for XML file manipulation.

[11] http://econcs.eecs.harvard.edu/jopt

[12] This is in contrast to indirect price-based approaches, such as clock-proxy [1], in which bidders must be able to reason about the RP-constraints implied by bids in each round.

λ^{t-1} at valuations v^{α}, then if the upper and lower values at the start of round t already satisfy the RP rule (and without the need for any tie-breaking), the provisional trade is efficient for ***all*** *valuations consistent with the current bid trees.* When linear CE prices exist, this provides for a soundness and completeness statement: if $PP = PO$, linear CE prices exist, and the RP rule is satisfied, the provisional trade is efficient (soundness); if prices are exact CE prices for the provisional trade at v^{α}, but the trade is inefficient with respect to some valuation profile consistent with the current bid trees, then at least one bidder must fail RP with her current bid tree and progress will be made (completeness). Future work must study convergence experimentally, and extend this theory to allow for approximate prices.

Some strategic aspects of our ICE design deserve comment, and further study. First, we do not claim that truthfully responding to the RP rule is an ex post equilibrium.[13] However, the exchange is designed to mimic the Threshold rule in its payment scheme, which is known to have useful incentive properties [16]. We must be careful, though. For instance we do not suggest to provide α^{eff} to bidders, because as α^{eff} approaches 1 it would inform bidders that bid values are becoming irrelevant to determining the trade but merely used to determine payments (and bidders would become increasingly reluctant to increase their lower valuations). Also, no consideration has been given in this work to collusion by bidders. This is an issue that deserves some attention in future work.

8. CONCLUSIONS

In this work we designed and prototyped 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, and a proxied elicitation method with optimistic and pessimistic valuations with a new method to evaluate a revealed- preference activity rule. The exchange is fully implemented in Java and is in a validation phase.

The next steps for our work are to allow bidders to refine the structure of the bid tree in addition to values on the tree. We intend to study the elicitation properties of the exchange and we have put together a test suite of exchange problem instances. In addition, we are beginning to engage in collaborations to apply the design to airline takeoff and landing slot scheduling and to resource allocation in wide-area network distributed computational systems.

Acknowledgments

We would like to dedicate this paper to all of the participants in CS 286r at Harvard University in Spring 2004. This work is supported in part by NSF grant IIS-0238147.

9. REFERENCES

[1] L. Ausubel, P. Cramton, and P. Milgrom. The clock-proxy auction: A practical combinatorial auction design. In Cramton et al. [9], chapter 5.

[2] M. Babaioff, N. Nisan, and E. Pavlov. Mechanisms for a spatially distributed market. In *Proc. 5th ACM Conf. on Electronic Commerce*, pages 9–20, 2001.

[3] M. Ball, G. Donohue, and K. Hoffman. Auctions for the safe, efficient, and equitable allocation of airspace system resources. In S. Cramton, Shoham, editor, *Combinatorial Auctions.* 2004. Forthcoming.

[4] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[5] S. Bikhchandani and J. M. Ostroy. The package assignment model. *Journal of Economic Theory*, 107(2):377–406, 2002.

[6] C. Boutilier. Solving concisely expressed combinatorial auction problems. In *Proc. 18th National Conference on Artificial Intelligence (AAAI-02)*, July 2002.

[7] C. Boutilier and H. Hoos. Bidding languages for combinatorial auctions. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[8] W. Conen and T. Sandholm. Preference elicitation in combinatorial auctions. In *Proc. 3rd ACM Conf. on Electronic Commerce (EC-01)*, pages 256–259, 2001.

[9] P. Cramton, Y. Shoham, and R. Steinberg, editors. MIT Press, 2004.

[10] S. de Vries, J. Schummer, and R. V. Vohra. On ascending Vickrey auctions for heterogeneous objects. Technical report, MEDS, Kellogg School, Northwestern University, 2003.

[11] S. de Vries and R. V. Vohra. Combinatorial auctions: A survey. *Informs Journal on Computing*, 15(3):284–309, 2003.

[12] M. Dunford, K. Hoffman, D. Menon, R. Sultana, and T. Wilson. Testing linear pricing algorithms for use in ascending combinatorial auctions. Technical report, SEOR, George Mason University, 2003.

[13] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: an architecture for secure resource peering. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 133–148, 2003.

[14] B. Hudson and T. Sandholm. 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*, pages 386–393, 2004.

[15] V. Krishna. *Auction Theory.* Academic Press, 2002.

[16] D. Krych. Calculation and analysis of Nash equilibria of Vickrey-based payment rules for combinatorial exchanges, April 2003.

[17] A. M. Kwasnica, J. O. Ledyard, D. Porter, and C. DeMartini. A new and improved design for multi-object iterative auctions. *Management Science*, 2004. To appear.

[18] E. Kwerel and J. Williams. A proposal for a rapid transition to market allocation of spectrum. Technical report, FCC Office of Plans and Policy, Nov 2002.

[19] S. M. Lahaie and D. C. Parkes. Applying learning algorithms to preference elicitation. In *Proc. ACM Conf. on Electronic Commerce*, pages 180–188, 2004.

[20] R. P. McAfee. A dominant strategy double auction. *J. of Economic Theory*, 56:434–450, 1992.

[21] P. Milgrom. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy*, 108:245–272, 2000.

[22] R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

[23] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proc. 2nd ACM Conf. on Electronic Commerce (EC-00)*, pages 1–12, 2000.

[24] D. C. Parkes, J. R. Kalagnanam, and M. Eso. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1161–1168, 2001.

[25] D. C. Parkes and L. H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Proc. 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 74–81, July 2000.

[26] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[27] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[28] T. Sandholm and C. Boutilier. Preference elicitation in combinatorial auctions. In Cramton et al. [9], chapter 10.

[29] T. Smith, T. Sandholm, and R. Simmons. Constructing and clearing combinatorial exchanges using preference elicitation. In *AAAI-02 workshop on Preferences in AI and CP: Symbolic Approaches*, 2002.

[30] P. R. Wurman and M. P. Wellman. A*k*BA: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, pages 21–29, 2000.

[13] Given the Myerson-Satterthwaite impossibility theorem [22] and the method by which we determine the trade we should not expect this.