Computational Intelligence, Volume 18, Number 4, 2002

eMediator: A NEXT GENERATION ELECTRONIC COMMERCE SERVER*

TUOMAS SANDHOLM

Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania

This paper presents *eMediator,* an electronic commerce server prototype that demonstrates ways in which algorithmic support and game-theoretic incentive engineering can jointly improve the efficiency of e-commerce. *eAuctionHouse,* the configurable auction server, includes a variety of generalized combinatorial auctions and exchanges, pricing schemes, bidding languages, mobile agents, and user support for choosing an auction type. We introduce two new logical bidding languages for combinatorial markets: the XOR bidding language and the OR-of-XORs bidding language. Unlike the traditional OR bidding language, these are fully expressive. They therefore enable the use of the Clarke-Groves pricing mechanism for motivating the bidders to bid truthfully. *eAuctionHouse* also supports supply/demand curve bidding. *eCommitter,* the leveled commitment contract optimizer, determines the optimal contract price and decommitting penalties for a variety of leveled commitment contracting mechanisms, taking into account that rational agents will decommit strategically in Nash equilibrium. It also determines the optimal decommitting strategies for any given leveled commitment contract. *eExchangeHouse,* the safe exchange planner, enables unenforced anonymous exchanges by dividing the exchange into chunks and sequencing those chunks to be delivered safely in alternation between the buyer and the seller.

Key words: combinatorial auction(s), bidding language(s), winner determination, bidding agent(s), proxy bidding, mobile agent(s), contracting, leveled commitment, breach, safe exchange.

1. INTRODUCTION

Electronic commerce is taking off rapidly, but the full power of algorithmic support and game-theoretic tools has not been harnessed to improve its efficiency. This paper presents *eMediator*, a next generation electronic commerce server that demonstrates ways in which these techniques can improve e-commerce both in terms of processes and outcomes. The server is designed to be used by both human and automated agents. The result of our four-year design and implementation effort is now available for use on the web at *http://www.cs.cmu.edu/~amem/eMediator*. The three components of *eMediator* are an auction house, a leveled commitment contract optimizer, and a safe exchange planner. These are discussed in Sections 2, 3, and 4, respectively. Each one exhibits interesting interplay between algorithms (such as search) and game-theoretic incentive engineering.

2. *eAuctionHouse*: A CONFIGURABLE AUCTION

One of the services that *eMediator* provides is a free-to-use Internet auction and exchange prototype called *eAuctionHouse*. Several successful commercial Internet auction sites already exist—such as eBay and Yahoo—and interesting academic auction houses have recently appeared on the Internet (Rodriguez-Aguilar et al. 1997; Wurman et al. 1998). Our motivation in developing an auction server was to prototype novel next generation features, and test their feasibility both computationally and in terms of user comfort. *eAuctionHouse* allows users from across the Internet to buy and sell goods as well as to set up markets. It is a

*Part of this work was supported by the National Science Foundation under CAREER Award IRI-9703122 and Grants IRI-9610122, IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678. Early versions of this paper appeared at the International Conference on Autonomous Agents (AGENTS-00), Barcelona, Spain, pp. 73–96 in the AAAI-99 Workshop on AI in Electronic Commerce, Orlando, FL 1999, pp. 46–55; and as a Washington University, St. Louis, Department of Computer Science technical report WU-CS-99-02, Jan. 1999.

Address correspondence to Tuomas Sandholm, Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213; e-mail: sandholm@cs.cmu.edu.

third-party site, so both sellers and buyers can trust that it executes the market mechanisms as stated. It is implemented in Java, with some of the computationally intensive algorithms in C++. The information about the markets is stored in a relational database to increase reliability. *eAuctionHouse* has been in operation on the Web since December 1998 and, to our knowledge, was the first Internet auction to support combinatorial auctions, bidding via graphically drawn price-quantity graphs, and bidding by mobile agents. *eAuctionHouse* also offers a wide range of market types to be chosen from, and supports the user in choosing the appropriate market type for the setting at hand. These features are now discussed in order.

2.1. Combinatorial Auctions

In a *sequential auction,* items are auctioned one at a time. If a bidder has preferences over bundles, that is, combinations of items (as is often the case, for example, in electricity markets, equities trading, bandwidth auctions (McAfee and McMillan 1996; McMillan 1994), and transportation exchanges (Sandholm 1993)), then bidding in such auctions is difficult. To determine her valuation for an item, the bidder needs to guess what items she will receive in later auctions. This requires speculation on what the others will bid in the future because that affects what items she will receive. Furthermore, what the others bid in the future depends on what they believe others will bid, etc. This counterspeculation introduces computational cost and other wasteful overhead. Moreover, in auctions with a reasonable number of items, such lookahead in the game tree is intractable, and then there is no known way to bid rationally. Bidding rationally would involve optimally trading off the cost of lookahead against the gains it provides, but that would again depend on how others strike that trade-off. Furthermore, even if lookahead were computationally manageable, usually uncertainty remains about the others' bids because agents do not have exact information about each others' preferences. This often leads to inefficient allocations where bidders fail to get the combinations they want and do get ones they do not (Boutilier et al. 1999; Sandholm 2000).

In a *parallel auction* the items are open for auction simultaneously and bidders may place their bids during a certain time period. This has the advantage that the others' bids partially signal to the bidder what the others' bids will end up being for the different items, so the uncertainty and the need for lookahead is not as drastic as in a sequential auction. However, the same problems prevail as in sequential auctions, albeit in a mitigated form.

Combinatorial auctions can be used to overcome the need for lookahead and the inefficiencies that stem from the related uncertainties (McMillan 1994; Rassenti et al. 1982; Rothkopf et al. 1998; Sandholm 1993). In a combinatorial auction bidders may place bids on combinations of items. This allows the bidders to express complementarities between items instead of having to speculate into an item's valuation of the impact of possibly getting other, complementary items. This capability is particularly important in illiquid, highly volatile, or noncommoditized markets where it is unsure whether one can acquire the items of a desired bundle one at a time (for the anticipated price).

Although combinatorial auctions can avoid the need for lookahead by the bidders and tend to therefore lead to more efficient allocations, they impose significant complexity on the auctioneer because the auctioneer needs to determine the winners. This is a nontrivial task. For example, the Federal Communications Commission (FCC) saw the desirability of combinatorial bidding in their bandwidth auctions, but that was not allowed due to perceived intractability of winner determination (among other reasons). (This is about to change given the speed improvement in winner determination algorithms. For one, the FCC is planning to hold its first combinatorial spectrum auction in Spring 2003.)

Our auction server supports a variety of combinatorial auctions. The following subsections discuss some of the main variants.

2.1.1. The OR Bidding Language. In the combinatorial auction setting that has been most commonly discussed (Rothkopf et al. 1998), each bidder can bid on combinations of indivisible items, and her bids are implicitly joined with nonexclusive OR, meaning that any number of her bids can be accepted. Winner determination in this case is the problem of deciding which bids win so as to maximize the sum of the bid prices, under the constraint that every item is allocated to at most one bid. This cannot be done in general in polynomial time in the size of the input (unless $\mathcal{P} = \mathcal{NP}$) because the problem is $\mathcal{NP}$-complete (Rothkopf et al. 1998) (it is analogous to weighted set packing which is $\mathcal{NP}$-complete (Karp 1972)).

Recently, we proved that even approximate winner determination is hard if one is interested in worst case guarantees. Specifically, no polynomial time algorithm can guarantee an allocation within a bound $n^{1-\epsilon}$ from optimum for any $\epsilon > 0$ (unless $\mathcal{NP} = \mathcal{ZPP}$) where n is the number of bids (Sandholm 2002).[1] If the bids exhibit special structure, better approximations can be achieved in polynomial time (Chandra and Halldórsson 1999; Halldórsson 1998; Halldórsson and Lau 1997; Hochbaum 1983; Sandholm and Suri 2000), but even these guarantees are so far from optimum that they are irrelevant for auctions in practice (Sandholm 2002).

Polynomial time winner determination can be achieved by restricting the combinations on which the agents are allowed to bid (Penn and Tennenholtz 2000; Rothkopf et al. 1998; Sandholm and Suri 2000; Tennenholtz 2000). However, because the agents may then not be able to bid on the combinations they want, similar economic inefficiencies prevail as in noncombinatorial auctions.

We recently generated another approach to optimal winner determination (Sandholm 2002). The motivation was to

- allow bidding on all combinations;
- strive for the optimal allocation;
- capitalize heavily on the sparseness of bids. In practice, the space of bids is extremely sparsely populated. For example, if there are 100 items, there are $2^{100} - 1$ combinations, and it would take longer than the life of the universe to bid on all of them even if every person in the world submitted a bid per second. Sparseness of bids implies sparseness of the allocations that need to be checked. Our algorithm constructively checks each allocation that has positive value exactly once, and does not construct the other allocations. Therefore, the algorithm only generates those parts of the search space which are actually populated by bids. The disadvantage then is that the run time depends on the bids received.

We achieve these goals by a tree search algorithm that does provably sufficient selective generation of children in the search and by using a method for fast child generation, heuristics that are accurate and optimized for speed, and four methods for preprocessing the search space. Whereas the worst-case complexity is exponential in the number of items for sale, it is polynomial in the number of bids. This is desirable because the auctioneer can control the number of items that are for sale in a given combinatorial auction, but usually cannot (and should not) restrict the number of bids submitted. The algorithm scales up well in practice. For details and experimental results, see Sandholm (2002). The algorithm has been further refined by others (Fujishima et al. 1999), and we then designed a completely different search algorithm that promised to be drastically faster (Sandholm and Suri 2000). Recently, we

[1] We proved this via an approximation-preserving polynomial reduction from a maximum clique (which is known to be inapproximable (Håstad 1999)).

developed a yet faster algorithm, CABOB, for the problem. Experimental results show that it is currently the fastest algorithm for optimal winner determination, scaling to hundreds of items and thousands or tens of thousands of bids (Sandholm et al. 2001).

2.1.2. The XOR Bidding Language. The OR bidding language implicitly assumes that the bids are superadditive: $b_i(S \cup S') \geq b_i(S) + b_i(S')$ where $b_i(S)$ is the bid of agent i on combination S. But what would happen if agent 1 bid $b_1(\{1\}) = \$5$, $b_1(\{2\}) = \$4$, and $b_1(\{1, 2\}) = \$7$, and there were no other bidders? The auctioneer could allocate items 1 and 2 to agent 1 separately, and that agent's bid for the combination would value at $\$5 + \$4 = \$9$ instead of \$7. So, the OR bidding language focuses on settings where combinatorial bids are introduced to capture synergies (complementarities) among items. On the other hand, in many real-world settings local subadditivities (substitutability) can occur as well. For example, when bidding for a landing slot for a plane, the bidder is willing to take any one of a host of slots, but does not want more than one.

To address this, we introduced the *XOR bidding language* which has since become popular (e.g. Parkes and Ungar (2000)). In this language, each bidder can submit one *XOR disjunct,* that is, each bidder can submit multiple bids (each bid including a set of items and a price), but only one bid from any given bidder can be accepted. *eAuctionHouse* supports the XOR bidding language. This allows the bidders to express general preferences with both positive and negative complementarities. In other words, the language is expressive enough to represent any mapping from combinations of items to prices.

Optimal winner determination in the XOR bidding language is at least as hard as in the OR bidding language. The reason is that any problem instance from the OR bidding language can be converted into the XOR bidding language (simply have each bid come from a separate virtual bidder; in this way there would be no XOR constraints). Therefore, the negative results, $\mathcal{NP}$-hardness and inapproximability, apply to the XOR bidding language as well. Our winner determination algorithms extend to the XOR bidding language by inserting the extra constraints that no two bids from the same bidder can be accepted (Sandholm 2002). These extra constraints actually *reduce* the size of the search space. However, this does not imply that the search becomes faster.

To enable an economically efficient allocation to be reached, it would be desirable to extract truthful valuation revelations as bids from the bidders. This is an issue because in traditional "pay-your-winning-bid" auctions, the bidders have incentive to underbid strategically. Bidding truthfully can be made incentive compatible (a weakly dominant strategy) by using the *Clarke-Groves mechanism* (Clarke 1971; Groves 1973) (a.k.a. *generalized Vickrey auction* (MacKie-Mason and Varian 1995)). This means that each bidder is motivated to bid truthfully irrespective of how the others bid. This renders counterspeculation unnecessary. With unrestricted bidder preferences, the Clarke-Groves mechanism requires a fully expressive bidding language. Therefore, our XOR bidding language enables incentive-compatible combinatorial auctions, where the traditional OR bidding language does not.

With our XOR bidding language, the Clarke-Groves mechanism can be applied to the combinatorial auction setting as follows. Winning bids are determined so as to maximize the auctioneer's revenue under the constraint that each item can be allocated to at most one bid (while honoring the XOR constraints). The amount that an agent needs to pay is the sum of the others' winning bids had the agent not submitted any bids, minus the sum of the others' winning bids in the actual optimal allocation. Therefore, the winner determination problem has to be solved once overall, and once per winning agent without any of that agent's bids. This makes fast winner determination even more crucial. Note that, for example, just removing one winning bid at a time would not constitute an incentive-compatible mechanism.

Incentive compatibility can also be lost if either winner determination or price determination is conducted only approximately.[2]

2.1.3. The OR-of-XORs Bidding Language. In addition to full expressiveness, the convenience of using combinatorial auctions is another important issue. Whereas the XOR bidding language is fully expressive, in the worst case expressing one's preferences requires one to submit a bid on each of the $2^m - 1$ bundles of items (where m is the number of items). A shorter representation of preferences without loss of expressive power could be possible by allowing a richer input language.[3] To this end, we introduced the *OR-of-XORs bidding language*. Each bidder can submit multiple XOR disjuncts, where these disjuncts are combined with OR. For example, a bidder could offer

$$[(\{1, 2\}, \$30)\ \textit{XOR}\ (\{1, 3\}, \$25)]$$
$$\textit{OR}\ [(\{4, 5\}, \$75)\ \textit{XOR}\ (\{3, 5\}, \$55)\ \textit{XOR}\ (\{6\}, \$25)]$$
$$\textit{OR}\ [(\{7, 8\}, \$25)]$$

Such OR-of-XORs expressions maintain full expressive power, but lead to shorter (at least no longer) input descriptions than in the XOR bidding language (because the latter is a special case). We also believe that OR-of-XORs expressions are a more natural way to input preferences. The OR connectors represent independence. For example, if a company bids for inputs to its production, and has three plants, it could submit 3 XOR disjuncts (one for each plant, and the different alternative inputs would be XOR'ed) to represent the fact that the plants' production plans are independent. This avoids the combinatorial listing of *combinations* of bundles (as would be the case with XOR bids) in cases where the values of the bundles are independent.

In eAuctionHouse, the bidder can submit such OR-of-XORs expressions in table form. Each row in the table is a combinatorial bid (that lists the set of items and a price), and the rows are combined with XOR. These XOR disjuncts are combined together with a nonexclusive OR. We do this by allowing the user to submit multiple tables—an intuitive way of representing independence across XOR disjuncts (tables). Winners can be determined with the same algorithm as in the case of XOR bids by using exclusivity constraints only between bids that are in the same XOR disjunct (table).

Other enrichments to the language are also possible. Since even the XOR bidding language is fully expressive, expressiveness should be viewed as a necessary, but not sufficient, condition in designing bidding languages. Between fully expressive input languages the appropriate comparison criterion is the convenience of their use in the particular application domain in question.[4]

2.1.4. Other Combinatorial Auction Generalizations. *eAuctionHouse* also supports *multi-unit* combinatorial markets where there can be multiple indistinguishable units of each

[2] A recent paper shows that if each bidder is interested in only one bundle, then incentive compatibility can be guaranteed even under approximate winner determination (Lehmann et al. 1999).

[3] One idea toward this direction was presented early on by Rassenti et al. (1982). They allowed each bidder to place combinatorial bids and to state the maximal number of bids that could be accepted from that bidder. Our XOR bidding language could be viewed as a special case of this where that number is one.

[4] A recent paper proposes a generalization of the OR-of-XORs language called OR*. In that language, a bidder can submit bids as nodes in a graph, and can submit edges into the graph corresponding to arbitrary XOR constraints. Because the OR* language is a generalization of the OR-of-XORs language, it is fully expressive and never less compact. Furthermore, it was shown that there exist preferences for which it is exponentially more compact than the OR-of-XORs language (Nisan 2000). We believe, however, that the OR-of-XORs language is more natural.

item for sale. Furthermore, it supports combinatorial *exchanges,* that is, combinatorial double auctions where there can be multiple buyers and multiple sellers. Some of the buyers may be sellers as well. Even in a single bid, a bidder can buy some items while selling others. Figure 1 illustrates the table-based bidding interface in a multi-unit combinatorial exchange that uses our OR-of-XORs bidding language (the number of units is specified in each cell of the table). Recent research has focused on developing even faster winner determination algorithms for multi-unit combinatorial auctions (Leyton-Brown et al. 2000; Sandholm and Suri 2000), and for single- and multi-unit combinatorial exchanges (Sandholm and Suri 2000; Sandholm et al. 2002).

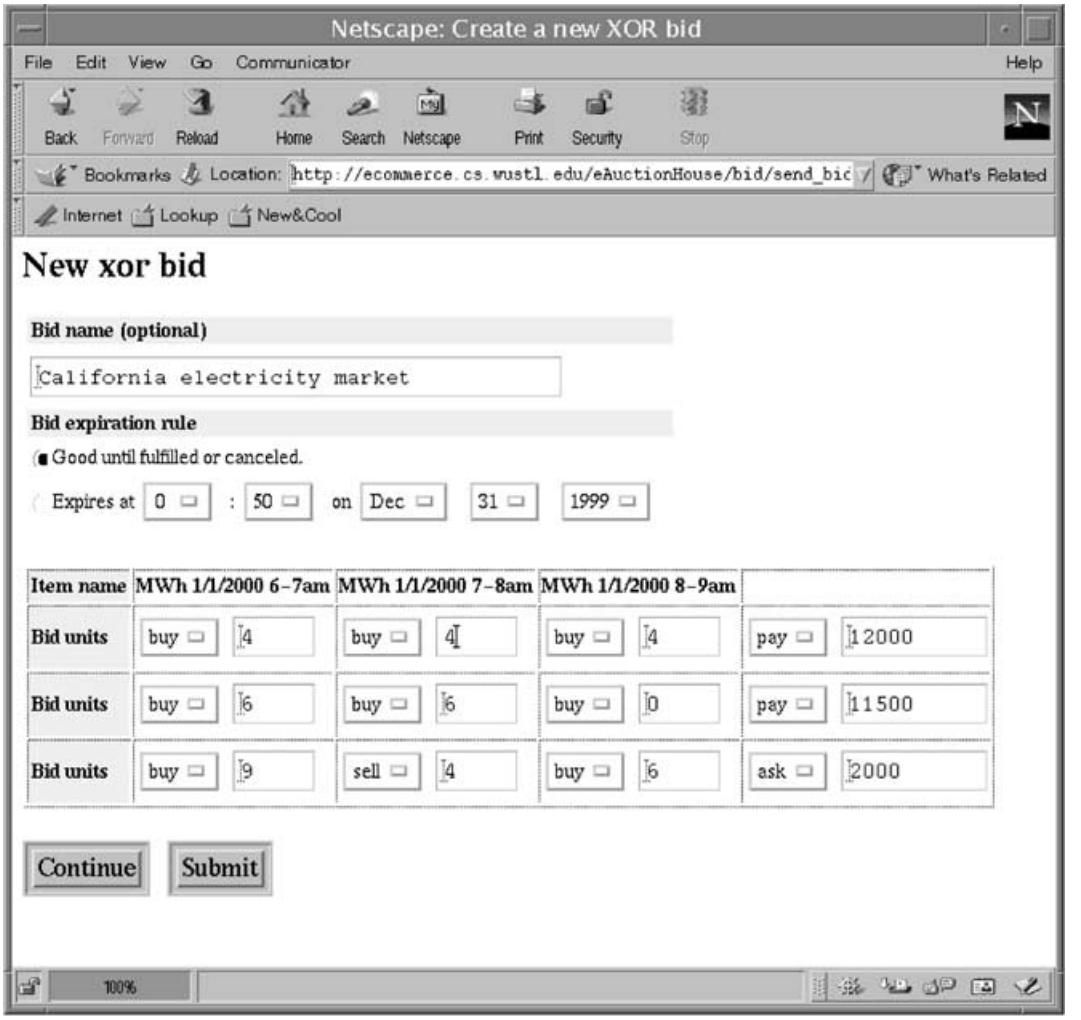

FIGURE 1. An XOR disjunct in a multi-item, multi-unit combinatorial exchange. The example is from an electricity market scenario where the agents can bid for combinations of electricity for different hours of the day, and for multiple megawatt hours for each hour of the day. In this example, a refinery operator wants consecutive hours of electricity for her plant. She prefers to run the plant for three hours at a rate of 4 MWh per hour, but running for two hours at 6 MWh per hour is also feasible. In the last alternative she acts both as a buyer and as a seller of different items in one combination.

2.2. Bidding via Price-Quantity Graphs

For markets where there is one item for sale, but multiple indistinguishable units of it, *eAuctionHouse* supports price-quantity graphs (that is, supply curves and demand curves) so that bidders can express continuous preferences (see Figure 2). For example when a bidder buys a larger quantity, she might only accept a lower unit price. Naturally, a bidder accepts anything below the curve as well (automatically colored region) because she will get the same quantity as on the curve, but at a lower price. Similarly, a seller would accept anything above her curve.[5]

In our implementation, the curves are piecewise linear both for the bidders' drawing convenience and for the convenience of winner determination. In single-sided auctions with price-quantity graph bidding, the winner determination algorithm works as follows. It sums the demand for every unit price (it does not loop though prices but uses the endpoints of each linear piece of the curve to do this). Then, it picks the unit price that maximizes the aggregate price under the constraint that not more is demanded than is available. Each bidder then gets the amount that she bid for at that unit price. In double auctions, both supply and demand curves are separately aggregated, and any one of the points where supply meets demand is chosen. If the curves were noncontinuous, it would be possible that no match exists. This holds both for single and double auctions. To prevent this, we use continuous curves, that is, the slope of each linear segment is finite. We do not assume that the curves are monotonic; for example, the curve in Figure 2 is nonmonotonic.

If the objective is to maximize economic efficiency (welfare among the buyers and sellers), then the best way to clear a fixed set of supply/demand curves is to pick a solution that maximizes the amount of trade under the constraint that supply meets demand. On the other hand, recently Sandholm and Suri studied the complexity of clearing supply/demand curves so as to maximize the profit of the party who runs the market (prices paid by buyers minus prices paid to sellers). They showed that under that objective, it is usually best to reduce the amount of trade. It turns out that with piecewise-linear curves, *nondiscriminatory clearing* (where each buyer is cleared at the same price, and each seller is cleared at the same price—which is generally different from the buyers' price) can be done optimally in polynomial time. However, optimal *discriminatory clearing* (where each party is cleared at a potentially different price) is $\mathcal{NP}$-complete. If the curves are linear, then even discriminatory clearing can be done in polynomial time. These results hold both for auctions (Sandholm and Suri 2001) and for exchanges (Sandholm and Suri 2002).

2.3. Support for Choosing an Auction Type

The auction server, *eAuctionHouse*, supports a wide variety of auction types. The user that sets up a given auction (she may be a buyer, a seller, or a third-party facilitator) decides the auction type. However, since the space of different auction types is enormous, the auction server helps the user in making the choice. First, only choices that are sensible, based on game-theoretic analyses or economics experiments, are provided as alternatives. Furthermore, there is an expert system that restricts the choice of auction types given the auction setting (Figure 3). For any given auction setting, it tells the user what kinds of bids can be accepted, and what price determination scheme should be used. The auction setting differs based on whether it is a single or double auction, whether there are one or multiple items, and whether there are one or multiple units of each item. Furthermore, the units can be divisible

[5] Optimark Technologies, Inc. does a fuzzier match where each bidder specifies how much she prefers different regions of the price-quantity plane (Lupien and Rickard 1997).

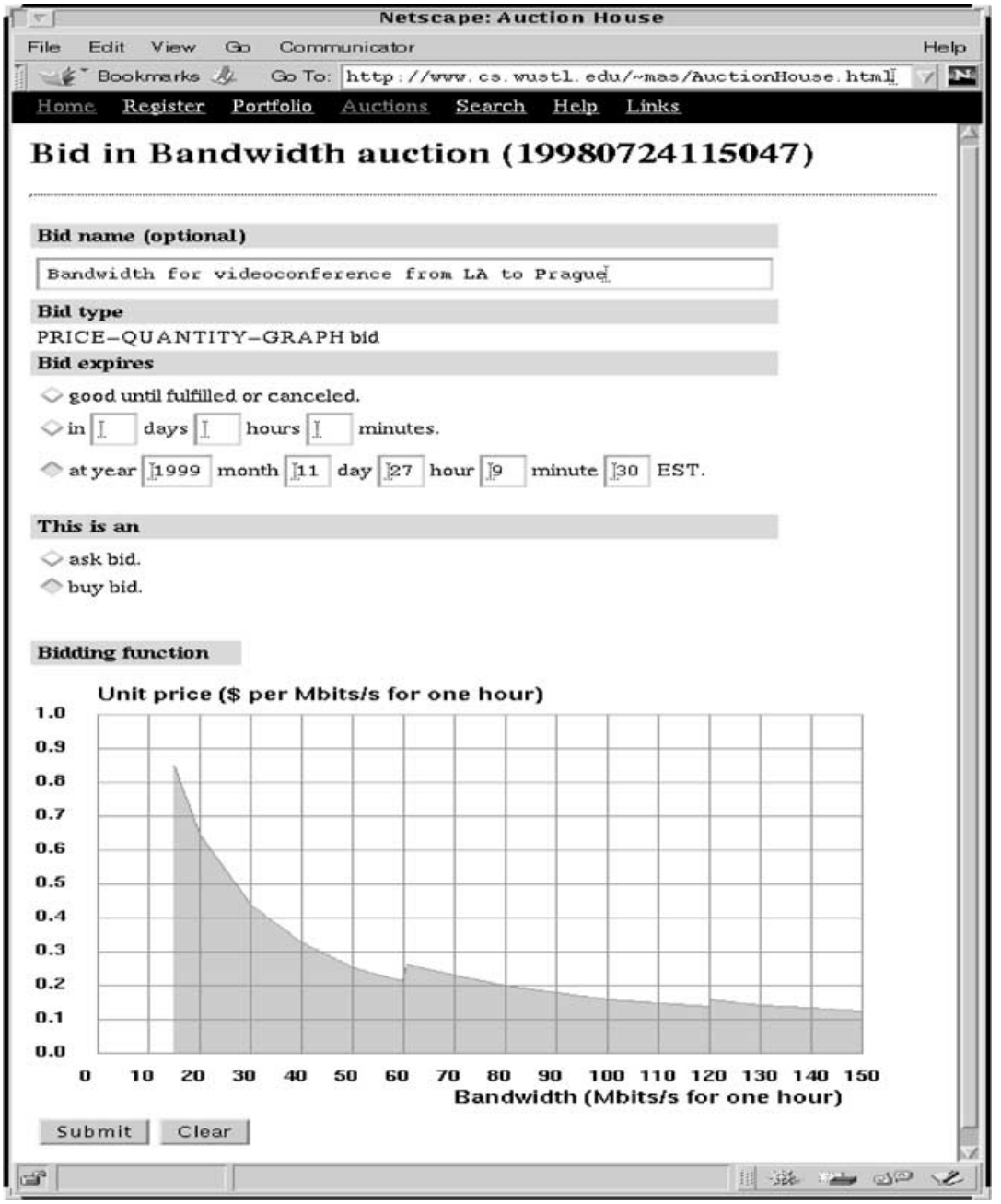

FIGURE 2. A price-quantity graph allows the user to express continuous preferences in the auction server of eMediator. This figure corresponds to the user being able to hold a videoconference at three alternative picture resolutions requiring a bandwidth of 15, 60, or 120 Mbps. The auctioned item could be a prebundled combination of items. For example, the virtual circuit from Los Angeles to Prague can use several network links owned by different backbone providers.

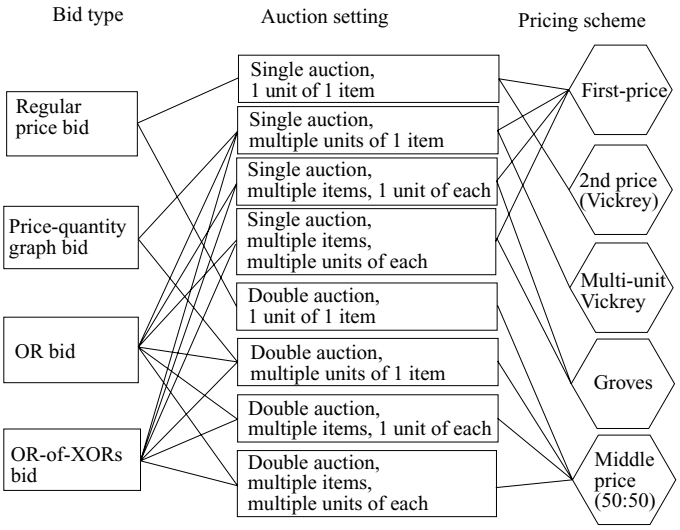

FIGURE 3. Expertise showing valid combinations of choices of some of the parameters of *eAuctionHouse*.

or indivisible. The bid types include a regular price bid where the bidder specifies the price for a good; a price-quantity graph bid (Figure 2); an OR bid; and an OR-of-XORs bid.

The first-price pricing scheme charges the buyer the price of her bid. This scheme leads to strategic underbidding. In single-unit ascending open-cry auctions, each bidder's best strategy is to bid a small increment more than the current price, and stop when her valuation is reached. In sealed-bid auctions with common knowledge assumptions about the priors from which the bidders' private valuations are drawn, a Nash equilibrium analysis can be conducted to determine how much each agent should underbid as a function of her valuation (e.g., Wolfstetter (1994)).

The second-price (Vickrey) auction charges the winning bidder the price of the second-highest bid. Under certain restrictions (Sandholm 2002), it is each bidder's best strategy to bid her true valuation (Vickrey 1961).

The multi-unit Vickrey auction is a generalization of the Vickrey auction to settings with multiple units of an item, or in other words, multiple indistinguishable goods. The bidders can use any of the following fully expressive bidding languages: (1) the XOR bidding language, where each bid includes a price and a quantity, (2) the OR-of-XORs bidding languages, where each bid includes a price and a quantity, or (3) a price-quantity curve as described above. (Even in this restricted setting, the OR bidding language would not be fully expressive. For example, it would not enable an agent to state that the value of 5 units is lower than the value of 2 units plus the value of 3 units.) The units are assigned to the bidders so as to maximize the sum of the accepted bid prices. Incentive-compatible pricing is achieved, again, using the Clarke-Groves mechanism. Specifically, the amount a bidder has to pay is the sum of the prices of the others' accepted bids had the bidder not participated, minus the sum of the prices of the others' accepted bids. Another possible generalization of the Vickrey auction to the multi-unit setting is to charge every bidder the price of the highest bid that just did not win (a.k.a. the *uniform-price auction*). Our auction server uses the former method because the latter is not incentive compatible: it falls prey to *demand reduction lies* by the bidders (Ausubel and Cramton 1996).

For multi-item auctions (with one or more units per item), the Clarke-Groves mechanism, as discussed earlier, is the appropriate generalization of the Vickrey auction. Each bidder's dominant strategy is to bid truthfully.

In double auctions (exchanges), our server splits the gains equally in the standard way. The price is halfway between the bid and the ask. In combinatorial double auctions (exchanges), we maximize the *surplus* (prices of accepted bids minus prices of accepted asks). The surplus can be divided arbitrarily between the buyers, the sellers, and the exchange—as long as no buyer pays more than the prices of her accepted bids (this guarantees that each bidder is motivated to participate, even in hindsight).

The auctions in *eAuctionHouse* also have several other parameters, including

- whether or not matches have to be exact in multi-unit auctions
- tie-breaking rule: random, older bid overrides, or newer bid overrides
- when to clear the auction: when a specific time is reached, every time a bid is received, periodically, or when no bids have been received for a specified time
- when the auction permanently closes: when it is cleared, when no bids have been received for a specified time, or when the auction's owner cancels it
- whether or not bid retraction is allowed (possibly for a penalty) before winners are determined
- whether or not bid retraction is allowed (possibly for a penalty) after winners are determined
- what information is revealed to the bidders during bidding: all bids, highest bids, or none
- what information is revealed to the bidders after clearing: all bids, winning bids, or none

2.4. *NOMAD*: Mobile Proxy Agents

Our auction house supports mobile agents so that a user can have her agent actively participating in the auction while she is disconnected. For example, the user can launch her agent over the phone from an airplane using a laptop, and then disconnect. Mobile agents that execute on the agent dock which is on (or near) the host machine of the auction server also reduce the network latency—an issue of key importance in time-critical bidding. The Michigan Internet AuctionBot (Wurman et al. 1998) provides a TCP/IP-level message protocol via which agents could participate in their auction. Their auction server differs from ours in that they do not provide support for mobile agents. Our auction server uses the commercial Concordia agent dock from Mitsubishi to provide mobile agents a safe execution platform from which they can observe what is transpiring in the auctions, bid, set up auctions, move to other hosts, etc. The user has the full flexibility of Java programming at her disposal when designing her mobile agent. We also provide an easy-to-use HTML interface for nonprogrammers where the user can specify what she wants her agent to do, and our system automatically generates the Java code for the corresponding mobile agent, and launches it. The following parameterizable mobile agent templates are currently available:

1. The *information agent* goes to an auction and sends an e-mail to the user when specified events occur. Using this agent, the user does not have to poll the auction, and gets notified of important events immediately.
2. The *incrementor agent* implements the best strategy on the user's behalf in single-item single-unit ascending open-cry first-price private-value auctions. It bids a small increment more than the current highest price, and stops if the user's reservation price is reached. With this agent the user does not have to follow the auction, and her best strategy in these settings is to report her valuation truthfully to the agent.
3. The *N-agent* underbids optimally on the user's behalf in single-item single-unit sealed-bid first-price auctions where the number of bidders, N, is known, and the bidders' private valuations are independently drawn from a uniform distribution. Specifically,

the symmetric Nash equilibrium strategy is to bid the user's valuation times $(N-1)/N$ (Rasmusen 1989). The user is then motivated to reveal her true valuation to the agent.

4. The *control agent* goes to an auction and submits very low noncompetitive bids. It is a speculator's tool to artificially increase the number, N, of bidders in an auction to mislead others, e.g., the N-agent. For example, it is in the seller's interest to submit control agents so that N-agents would bid higher.
5. The *discover agent* computes the expected gain from bidding a small amount more than the current highest price according to the agent's current distribution of her valuation. This is intended for settings where the user does not know her exact valuation for the item, but only a probability distribution on it. In the future, the probability distribution could be updated based on new information, or in non–private-value auctions, based on what others have bid.

Unlike current electronic negotiation servers which usually only provide an auction house, *eMediator* provides other types of services for facilitating ecommerce in addition, such as a leveled commitment contract optimizer, and a safe exchange planner. These are discussed in Sections 3 and 4, respectively.

3. *eCommitter*: A LEVELED COMMITMENT CONTRACT OPTIMIZER

Normal full commitment contracts are unable to take advantage of the possibilities that unknown future events provide. Once an agent agrees to a contract, she has to follow through no matter how future events unravel. Although a contract may be profitable to an agent when viewed *ex ante*, it need not be profitable when viewed after some future events have occurred. Similarly, a contract may have too low an expected payoff *ex ante*, but in some realizations of the future events, it may be desirable.

Contingency contracts have been suggested for utilizing the potential provided by future events among self-interested agents (Raiffa 1982). The contract obligations are made contingent on future events. In some games this increases the expected payoff to both parties compared to any full commitment contract. However, contingency contracts are often impractical because the space of combinations of future events may be large and unknown. Also, when events are not mutually observable, the observing agent can lie about what transpired.

Leveled commitment contracts are another method for capitalizing on future events (Sandholm and Lesser 2001). Instead of conditioning the contract on future events, a mechanism is built into the contract that allows unilateral decommitting. This is achieved by specifying in the contract the level of commitment by decommitment penalties, one for each agent. If an agent wants to decommit—that is, to be freed from the contract obligations—it can do so simply by paying the decommitment penalty to the other party. The method requires no explicit conditioning on future events: each agent can do her own conditioning dynamically. No event verification mechanism against lying is required either. The decommitment possibility increases each agent's expected payoff under very general assumptions (Sandholm and Lesser 2001).

eMediator includes an optimizer for leveled commitment contracts. We call it *eCommitter*. We analyze contracting situations from the perspective of two risk-neutral agents, each of which attempts to maximize his own expected payoff: the *contractor* who pays to get a task done, and the *contractee* who gets paid for handling the task. The framework can be interpreted as modeling other types of settings than task allocation also—for example, general allocation of rights and obligations where the agents' costs and gains of the rights and obligations may change. In what follows, we word the results in the context of task allocation.

The contractor tries to minimize the contract price ρ that he has to pay to get the task handled. The contractee tries to maximize the payoff ρ that she receives from the contractor for handling the task. We study a setting where the future of the agents involves uncertainty. Specifically, the agents might receive outside offers.[6] The contractor's best outside offer $\breve{a}$ is only probabilistically known *ex ante*, by both agents, and is characterized by a probability density function $f(\breve{a})$. If the contractor does not receive an outside offer, $\breve{a}$ corresponds to its best outstanding outside offer or its fallback payoff, i.e., payoff that it receives if no contract is made. The contractee's best outside offer $\breve{b}$ is also only probabilistically known *ex ante*, and is characterized by a probability density function $g(\breve{b})$. If the contractee does not receive an outside offer, $\breve{b}$ corresponds to its best outstanding outside offer or its fallback payoff.[7] The variables $\breve{a}$ and $\breve{b}$ are assumed to be statistically independent, and f and g are assumed to be common knowledge.

The contractor's options are either to make a contract with the contractee or to wait for $\breve{a}$. Similarly, the contractee's options are either to make a contract with the contractor or to wait for $\breve{b}$. The two agents could make a full commitment contract at some price. Alternatively, they can make a leveled commitment contract which is specified by the contract price, ρ, the contractor's decommitment penalty, a, and the contractee's decommitment penalty, b. We restrict our attention to contracts where $a \geq 0$ and $b \geq 0$, that is, agents do not get paid for decommitting.[8] The contractor has to decide on decommitting when he knows his outside offer $\breve{a}$ but does not know the contractee's outside offer $\breve{b}$. Similarly, the contractee has to decide on decommitting when she knows her outside offer $\breve{b}$ but does not know the contractor's. This seems realistic from a practical contracting perspective.

3.1. Nash Equilibria for a Given Contract

One concern is that a rational self-interested agent is reluctant to decommit because there is a chance that the other party will decommit, in which case the former agent gets freed from the contract, does not have to pay a penalty, and collects a penalty from the breacher. Sandholm and Lesser (2001) showed that despite such insincere decommitting the leveled commitment feature increases each contract party's expected payoff, and enables contracts in settings where no full commitment contract is beneficial to all parties.

The contractor decommits if he gets a low enough outside offer, e.g., he can get his task handled at a low cost. We denote his decommitting threshold by $\breve{a}^*$, so his decommitting probability is

$$p_a = \int_{-\infty}^{\breve{a}^*} f(\breve{a})\, d\breve{a} \tag{1}$$

The contractee decommits if she gets a high enough outside offer, e.g., gets paid for handling a task. We denote her decommitting threshold by $\breve{b}^*$, so her decommitting probability is

$$p_b = \int_{\breve{b}^*}^{\infty} g(\breve{b})\, d\breve{b} \tag{2}$$

[6] The framework can also be interpreted to model situations where the agents' cost structures for handling tasks and for getting tasks handled change, e.g., due to resources going offline or becoming back online.

[7] Games where at least one agent's future is certain are a subset of these games. In such games all of the probability mass of $f(\breve{a})$ and/or $g(\breve{b})$ is on one point.

[8] A recent paper shows that this restriction does not reduce the welfare of the contract parties (Sandholm and Zhou 2002).

3.1.1. Sequential Decommitting Mechanisms. In our sequential decommitting mechanism, one agent has to reveal her decommitting decision before knowing whether the other party decommits. While our implementation analyzes both orders of decommitting, here we only discuss the setting where the contractee has to decide first. The case where the contractor decides first is analogous. There are two alternative leveled commitment contracts that differ on whether or not the agents have to pay the penalties to each other if both decommit.

If the contractee has decommitted, the contractor's best move is not to decommit, because $-\breve{a} - a + b \leq -\breve{a} + b$ (because $a \geq 0$). This also holds for a contract where neither agent has to pay a decommitment penalty if both decommit since $-\breve{a} \leq -\breve{a} + b$. In the subgame where the contractee has not decommitted, the contractor's best move is to decommit if $-\breve{a} - a > -\rho$, i.e.,

$$\breve{a}^* = \rho - a \tag{3a}$$

The contractee gets $\breve{b} - b$ if she decommits, $\breve{b} + a$ if she does not but the contractor does, and ρ if neither decommits. Thus the contractee decommits if $\breve{b} - b > p_a(\breve{b} + a) + (1 - p_a)\rho$. If $p_a = 1$, this is equivalent to $-b > a$ which is false because $a \geq 0$ and $b \geq 0$. In other words, if the contractee surely decommits, the contractor does not. On the other hand, the above is equivalent to

$$\breve{b} > \rho + \frac{b + ap_a}{1 - p_a} \stackrel{\text{def}}{=} \breve{b}^* \quad \text{when } p_a < 1 \tag{4a}$$

3.1.2. Simultaneous Decommitting Mechanisms. In our simultaneous decommitting mechanisms, agents have to reveal their decommitment decisions simultaneously. We first discuss the variant where both have to pay the penalties to each other if both decommit. The contractor decommits if $p_b \cdot (-\breve{a} + b - a) + (1 - p_b)(-\breve{a} - a) > p_b \cdot (-\breve{a} + b) + (1 - p_b)(-\rho)$. If $p_b = 1$, this equates to $a < 0$, but we have already ruled out contracts where an agent gets paid for decommitting. On the other hand, this equates to

$$\breve{a} < \rho - \frac{a}{1 - p_b} \stackrel{\text{def}}{=} \breve{a}^* \quad \text{when } p_b < 1 \tag{3b}$$

The contractee decommits if $(1 - p_a)(\breve{b} - b) + p_a(\breve{b} - b + a) > (1 - p_a)\rho + p_a(\breve{b} + a)$. If $p_a = 1$, this equates to $b < 0$, but we ruled out contracts where an agent gets paid for decommitting. However, this equates to

$$\breve{b} > \rho + \frac{b}{1 - p_a} \stackrel{\text{def}}{=} \breve{b}^* \quad \text{when } p_a < 1 \tag{4b}$$

In another type of simultaneous decommitting mechanism, neither agent has to pay if both decommit. The contractor decommits if $p_b \cdot (-\breve{a}) + (1 - p_b)(-\breve{a} - a) > p_b \cdot (-\breve{a} + b) + (1 - p_b)(-\rho)$. If $p_b = 1$, this equates to $b < 0$, but we have already ruled out contracts where an agent gets paid for decommitting. On the other hand, this equates to

$$\breve{a} < \rho - a - \frac{bp_b}{1 - p_b} \stackrel{\text{def}}{=} \breve{a}^* \quad \text{when } p_b < 1 \tag{3c}$$

The contractee decommits if $(1 - p_a)(\breve{b} - b) + p_a\breve{b} > (1 - p_a)\rho + p_a(\breve{b} + a)$. If $p_a = 1$, this equates to $a < 0$, but we have ruled out contracts where an agent gets paid for

decommitting. However, this equates to

$$\check{b} > \rho + b - \frac{ap_a}{1 - p_a} \stackrel{\text{def}}{=} \check{b}^* \quad \text{when } p_a < 1 \tag{4c}$$

3.1.3. Using eCommitter *to Find the Nash Equilibria for a Given Contract.* To use the *eCommitter* optimizer, the user inputs the probability distributions f and g graphically or textually. The user also inputs the contract price ρ and the decommitment penalties (a and b). For each one of the decommitting mechanisms (which vary based on who has to reveal her decommitting decision first—the simultaneous mechanisms are also considered—and whether or not the agents have to pay the penalties to each other if both decommit), *eCommitter* finds all the Nash equilibria. In other words, *eCommitter* determines decommitting threshold pairs. *eCommitter* does this by solving the simultaneous Equations (3) and (4) which use (1) and (2).[9] Given an equilibrium, it is easy to compute the agents' expected payoffs under the contract. Finally, *eCommitter* presents all the equilibria and the associated expected payoffs to the user.

3.2. Using *eCommitter* to Find Optimal Contracts

In addition to determining how agents should play under a given leveled commitment contract, *eCommitter* also determines the optimal leveled commitment contracts. Again, the user inputs the distributions f and g graphically or textually. However, the user does not input the contract price or decommitment penalties. Rather, *eCommitter* determines the values of these (ρ, a, b) in a way that maximizes the sum of the contract parties' expected payoffs (Sandholm et al. 1999). (Again, that optimization algorithm takes into account that the agents decommit insincerely in Nash equilibrium according to Equations (1)–(4).) The expected benefit from the contract is the sum of the agents' expected payoffs under the contract, minus the sum of the agents' (expected) payoffs under the status quo where they do not make a contract.

The contract is optimized separately for each one of the decommitting mechanisms, which vary based on who has to reveal her decommitting decision first—the simultaneous mechanisms are also considered—and whether or not the agents have to pay the penalties to each other if both decommit. The optimal contracts for each mechanism are then presented to the user. For each mechanism, there is a continuum of optimal contracts that vary based on how the expected benefit is divided between the agents. *eCommitter* visualizes how each agent's payoff changes as a function of the contract price ρ, and shows how the decommitment penalties (a and b) should be tailored to the contract price so as to maximize the expected benefit.[10] *eCommitter* also points out, for each of the mechanisms, the fair optimal contract, that is, the contract that maximizes the expected benefit and divides it equally between the agents.

[9] Although solving this equation group is complex in general, we have the user input piecewise-linear f and g. This allows the problem to be decomposed into rectangles (corresponding to one piece of f and one of g), where f and g are linear within each rectangle. In that case, the equation group can be easily solved in constant time. Our optimizer solves the problem in this way for each rectangle separately. For details, see Sandholm et al. (1999).

[10] The expected benefit can be maximized while dividing the benefit in any way—as long as each party gets a nonnegative portion of the benefit (Sandholm et al. 1999; Sandholm and Zhou 2002). In other words, here integrative bargaining (increasing the expected benefit) does not hinder distributive bargaining (dividing the expected benefit). However, this independence only holds if the decommitment penalties are tailored to the chosen contract price.

4. *eExchangeHouse*: A SAFE EXCHANGE PLANNER

Contract execution is more difficult in electronic commerce than physical commerce because the parties may be anonymous and can disappear easily. The problem will be exacerbated in the future by software agents for automated negotiation. Such an agent can vanish by simply killing its process, and litigation is infeasible unless the other contract party knows which real-world entity the agent represented. Another problem is the lack of uniform laws on electronic commerce—and particularly agent-mediated commerce—in different countries.

An important aspect of contract execution is making sure that the seller gets paid, and that the buyer gets the goods. The risk is that once one party has received the item, he may be motivated to vanish without delivering his part of the contract. In fact, 6% of Americans with online buying experience have reported nondelivery (National Consumers League 1999).

Nondelivery could be avoided by having a trusted third party that takes the payment and goods, and carries out the transaction only after all parties have delivered their part to the intermediary. Today's electronic commerce implements a coarse one-sided variant of this where the third party takes the payment into escrow, and releases it to the seller only after the buyer has verified receipt of the goods. A disadvantage of these third-party escrow companies like Tradenable and Tradesafe is the cost of running such an intermediary, which is recovered as fees—currently about 3 to 5% of the contract price—from the contract parties.

A method for tackling this problem without third parties was developed in Sandholm and Lesser (1995). The exchange is divided into chunks where each party delivers a small amount at a time, and the exchange proceeds with such alternation. The method targets settings where dividing the goods into chunks is relatively inexpensive, such as is often the case, for example, with information goods and computational services. A sequence is called safe if each party is motivated to follow the exchange at every step in anticipation of the profit from the rest of the exchange instead of vanishing with what the other party has delivered so far. (Specifically, a safe sequence can be executed in subgame-perfect Nash equilibrium.) Some chunkings allow a safe sequence while others do not (Sandholm 1996; Sandholm and Lesser 1995). Similarly, some sequences of delivering given chunks are safe whereas others are not.

As part of *eMediator*, we built a safe exchange planner called *eExchangeHouse*. As input, the planner takes the following information:

- the contract price,
- how the seller's cost accrues as a function of how much he has delivered so far in the transaction,
- how the buyer's valuation accrues as a function of how much he has received so far in the transaction,
- how much gain the seller is willing to forego to avoid possible reputation costs from defecting in the exchange (or costs associated with the risk of getting caught after defecting), and
- how much gain the buyer is willing to forego for the same reasons.

The form of the cost and valuation accrual information depends on the setting. The setting can have one item, multiple independent items, or multiple dependent items to be exchanged. Also, there can be one unit of each item, multiple independent units of each item, or multiple dependent units of each item. Finally, the units can be divisible or indivisible.

For example, in the case where a single divisible item is exchanged, the user inputs a graph of how the buyer's valuation accrues as a function of how much has been delivered, and another graph that shows how the seller's cost accrues. As another example, in the case

of multiple distinguishable items, the user lists the items and how many units of each item are to be exchanged. For each possible *state* of the exchange (units of item 1 delivered × units of item 2 delivered × ...) the user inputs the buyer's valuation and the seller's cost. Combining indistinguishable goods into units of a single item significantly reduces the state space because within each good, only the number of units delivered matters, not which specific units have been delivered.

Based on this input, the planner finds a safe chunking that minimizes the number of chunks and a safe chunk sequence if they exist. If they do not exist, the user is alerted of this. The planner can determine the optimal safe exchange plan under any of the following four alternative objectives: (1) minimizing the number of deliveries, (2) minimizing the number of payments, (3) minimizing the number of payments plus the number of deliveries, and (4) minimizing the number of steps, where each step may include both a payment and a delivery. The chunking algorithms and the chunk sequencing algorithms are highly nontrivial; they are described and analyzed in detail in other papers (Sandholm 1996; Sandholm and Ferrandon 2000; Sandholm and Lesser 1995).

Generally, a user (buyer or seller) only knows the desired information about himself. He can use bounds on the other party's values and the exchange plan will still be safe. An upper bound can be used on how the buyer's value accrues and a lower bound on how the seller's cost accrues. Similarly, an upper bound can be used on the seller's cost of the completed contract and a lower bound can be used on the buyer's valuation of the completed contract. Finally, lower bounds can be used on the amounts that the agents are willing to forego in order to avoid defection-related adverse reputation costs or the chance of getting caught after nondelivery. Although the planned exchange will be safe when such bounds are used as the input, the planning problem may be infeasible if the bounds are too far off. In that case, *eExchangeHouse* reports that no safe exchange plan exists given the inputs.

Each trade party could have a software agent execute its sequence of delivery chunks. This would provide a level of isolation between the user and the exchange so that the user need not even know that the trade is occurring in chunks instead of all at once.

4.1. Example of Exchanging Independent Units of Independent Items

In this example the agents exchange multiple indivisible independent units of multiple independent items. Specifically, they exchange five stocks (multiple units of each): Microsoft (MSFT), Hewlett-Packard (HWP), Dell (DELL), eBay (EBAY) and Procter & Gamble (PG). The stocks have different values for the supplier and the demander (for example, because the supplier and the demander have different portfolios, and the value of a stock in a portfolio depends on how correlated the stock is with the rest of the portfolio statistically).

The input to the planner includes the contract price ($25,864.60) and the amounts the agents are willing to forego to avoid reputation-related problems (the seller is ready to forego $1000, and the buyer is ready to forego $1500). The input also includes the information in Table 1. The user actually submits this information by filling in such a table in *eExchangeHouse*. Note that the seller's cost per unit ($\Delta v_s / \Delta x_i$) can be greater than the demander's valuation per unit ($\Delta v_d / \Delta x_i$) for some items i—but not all of them—to allow safe exchange.

Based on this input, *eExchangeHouse* determines an optimal safe exchange plan which is shown in Figure 4.

4.2. Example of Exchanging Dependent Units of Dependent Items

Consider a software system consisting of two parts, a main package and a plug-in. The former takes four floppy disks (units) whereas the latter takes only one. They depend on each

TABLE 1. Input form for Exchanging Independent Units of Independent Items

Item Name	Number of Units to Exchange	$\frac{\Delta v_s}{\Delta x_i}$	$\frac{\Delta v_d}{\Delta x_i}$
MSFT	60	$90.81	$98
HWP	30	$102.63	$109.90
DELL	10	$41.81	$42.50
EBAY	100	$120.25	$127.25
PG	40	$85.10	$88.44

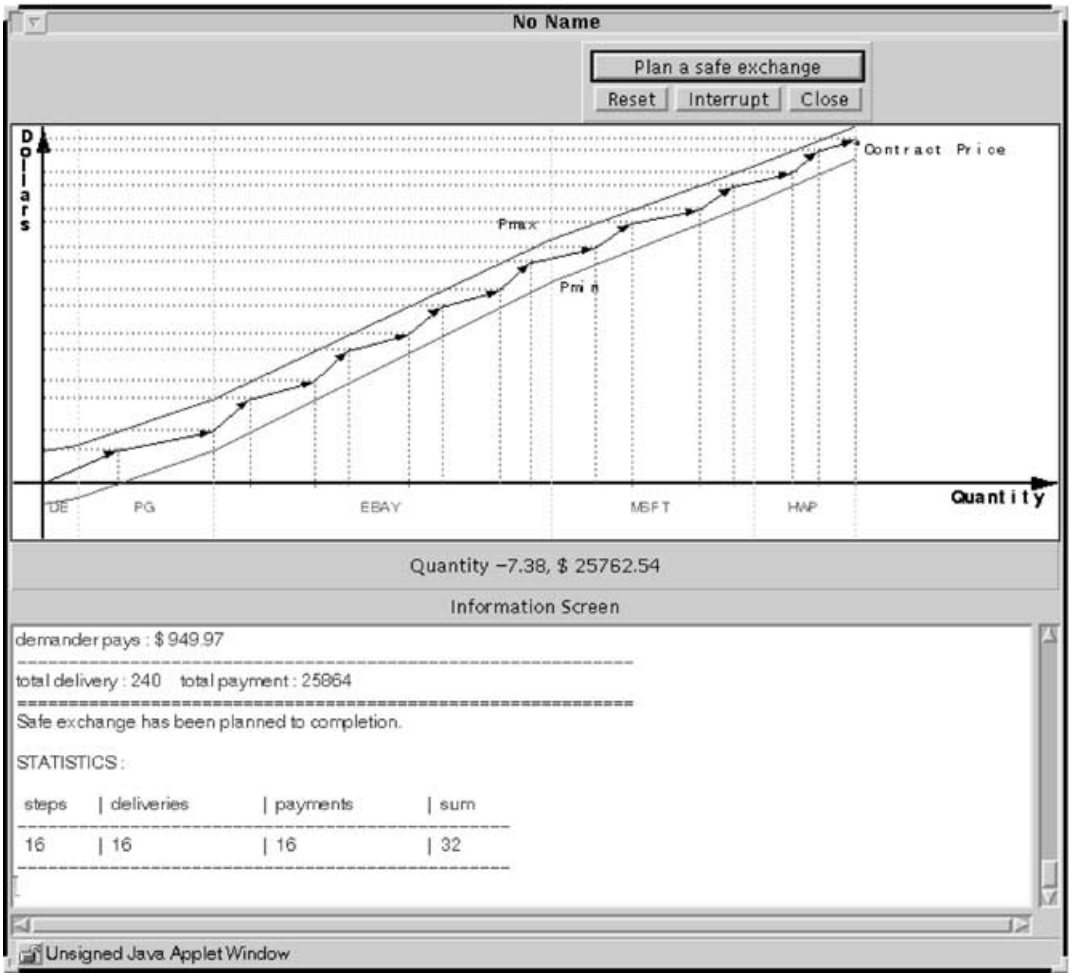

FIGURE 4. Output from the planner. This optimal plan is 16 steps long. Each step of the exchange is described in detail in the bottom window (only the end of the safe exchange plan, with summary statistics, is visible).

TABLE 2. Input From for Exchanging Dependent Units of Dependent Items

Plug-in	Package	v_s	v_d
0	0	0	0
0	1	5	3
0	2	9	6
0	3	13	10
0	4	17	14
1	0	10	6
1	1	13	10
1	2	16	15
1	3	18	20
1	4	19	26

other in the sense of the cost function of the seller (v_s) and the value function of the demander (v_d), as the user has expressed in Table 2.[11]

We see in this example that—without a plug-in—the cost of one disk of the package decreases (5, 4, 4, 4) depending on how many disks have been delivered so far. Also, the value of one disk increases (3, 3, 4, 4) depending on how many disks have been delivered so far. Therefore, the units of the item "package" are dependent. Moreover, once the plug-in is delivered, the cost and value of one disk of the package change (respectively (4, 4, 3, 2) and (4, 4, 5, 5)). Therefore, the item "package" depends on the item "plug-in." (Similarly, the plug-in depends on the package.)

Let the contract price be \$22, and let each agent be willing to forego \$2 to avoid negative reputation effects associated with defecting.

Given this input, and the objective of minimizing the number of exchange steps (where each step could include both a delivery and a payment), *eExchangeHouse* returns the following optimal safe exchange plan. In the first step, the supplier delivers 2 units of item "package" and the demander pays \$5. In the second step, the supplier delivers 1 unit of item "package" and the demander pays \$9. In the third step, the supplier delivers 1 unit of item "package" and the demander pays \$4. In the fourth step, the supplier does not deliver, but the demander pays \$4. In the fifth step, the supplier delivers 1 unit of item "plug-in" and the demander does not pay anything. So, the exchange is completed safely in five steps, containing four deliveries and four payments.

In a recent theoretical paper, we develop a unified model of safe exchange which captures the known safe exchange mechanisms such as cryptographic signatures, cryptographic coin ripping, and our chunking technique (as well as new safe-exchange mechanisms). In that general framework we uncover the conditions under which safe exchange is inherently possible or impossible—not just using our chunking mechanism, but using *any* mechanism (Sandholm and Wang 2002).

5. CONCLUSIONS

The *eMediator* electronic commerce server prototype exemplifies several new features that can facilitate more efficient ecommerce in the future. Each one of the three components

[11] In settings where the items are dependent, but the units within each item are independent, the amount of input required from the user can be reduced (Sandholm and Ferrandon 2000).

exhibits an interesting interplay between algorithms and game-theoretic incentive engineering.

eAuctionHouse, the configurable auction server, includes a variety of generalized combinatorial auctions and exchanges, pricing schemes, bidding languages, mobile agents, and user support for choosing an auction type. We introduced two new logical bidding languages for combinatorial markets: the XOR bidding language and the OR-of-XORs bidding language. Unlike the traditional OR bidding language, they are fully expressive. They therefore enable the use of the Clarke-Groves pricing mechanism for motivating the bidders to bid truthfully. *eAuctionHouse* also supports supply/demand curve bidding for single-item multi-unit markets.

eCommitter, the leveled commitment contract optimizer, determines the optimal contract price and decommitting penalties for a variety of leveled commitment contracting mechanisms, taking into account that rational agents will decommit strategically. It also determines the optimal decommitting strategies for any given leveled commitment contract.

eExchangeHouse, the safe exchange planner, enables unenforced anonymous exchanges by dividing the exchange into chunks and sequencing those chunks to be delivered safely in alternation between the buyer and the seller.

Each component can be used separately, but they can also be used together. For example, if a deal is reached using *eAuctionHouse* or *eCommitter*, then *eExchangeHouse* could be used to carry out the deal. As another example, the contract determined using *eCommitter* could be auctioned using *eAuctionHouse*.

In the future we are planning to add to *eMediator* nonmanipulable reputation maintenance algorithms, product evaluation methods, a nonmanipulable voting server, and coalition formation support.

ACKNOWLEDGMENTS

I thank Qianbo Huai for programming most of the auction house component. I thank Qianbo Huai, Alan Huffman, and Pradeep Gore for implementing the mobile agent component. I thank Sandeep Sikka, Samphel Norden, and Yunhong Zhou for their contributions to analyzing and programming the leveled commitment contract optimizer. I thank Vincent Ferrandon for programming the safe exchange planner.

REFERENCES

AUSUBEL, L. M., and P. C. CRAMTON. 1996. Demand reduction and inefficiency in multi-unit auctions. Technical Report 96-07, University of Maryland, Department of Economics.

BOUTILIER, C., M. GOLDSZMIDT, and B. SABATA. 1999. Sequential auctions for the allocation of resources with complementarities. *In* Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden. pp. 527–534.

CHANDRA, B., and M. M. HALLDÓRSSON. 1999. Greedy local search and weighted set packing approximation. *In* 10th Annual SIAM-ACM Symposium on Discrete Algorithms (SODA), pp. 169–176.

CLARKE, E. H. 1971. Multipart pricing of public goods. Public Choice, **11**:17–33.

FUJISHIMA, Y., K. LEYTON-BROWN, and Y. SHOHAM. 1999. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *In* Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, pp. 548–553.

GROVES, T. 1973. Incentives in teams. Econometrica, **41**:617–631.

HALLDÓRSSON, M. M. 1998. Approximations of independent sets in graphs. *In* The First International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), Aalborg, Denmark; LNCS 1444. *Edited by* K. Jansen and J. Rolim. Springer-Verlag, Berlin, pp. 1–14.

HALLDÓRSSON, M. M., and H. C. LAU, 1997. Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and 3-coloring. Journal of Graph Algorithms and Applications, **1**(3): 1–13.

HÅSTAD, J. 1999. Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica, **182**:105–142.

HOCHBAUM, D. S. 1983. Efficient bounds for the stable set, vertex cover, and set packing problems. Discrete Applied Mathematics, **6**:243–254.

KARP, R. M. 1972. Reducibility among combinatorial problems. *In* Complexity of Computer Computations. *Edited by* R. E. Miller and J. W. Thatcher. Plenum Press, New York, pp. 85–103.

LEHMANN, D., L. I. O'CALLAGHAN, and Y. SHOHAM. 1999. Truth revelation in rapid, approximately efficient combinatorial auctions. *In* Proceedings of the ACM Conference on Electronic Commerce (ACM-EC), Denver, CO, pp. 96–102.

LEYTON-BROWN, K., M. TENNENHOLTZ, and Y. SHOHAM. 2000. An algorithm for multi-unit combinatorial auctions. *In* Proceedings of the National Conference on Artificial Intelligence (AAAI), Austin, TX.

LUPIEN, W. A., and J. T. RICKARD. 1997. Crossing network utilizing optimal mutual satisfaction density profile. US Patent 5,689,652, granted Nov. 18 to Optimark Technologies.

MACKIE-MASON, J. K., and H. R. VARIAN. 1995. Generalized Vickrey auctions. Technical Report, University of Michigan.

MCAFEE, R. P., and J. MCMILLAN. 1996. Analyzing the airwaves auction. Journal of Economic Perspectives, **10**(1):159–175.

MCMILLAN, J. 1994. Selling spectrum rights. Journal of Economic Perspectives, **8**(3):145–162.

NATIONAL CONSUMERS LEAGUE. 1999. New NCL survey shows consumers are both excited and confused about shopping online. Survey conducted by Opinion Research Corporation *http://www.natlconsumersleague.org/BeEWisepr.html*.

NISAN, N. 2000. Bidding and allocation in combinatorial auctions. *In* Proceedings of the ACM Conference on Electronic Commerce (ACM-EC), Minneapolis, MN, pp. 1–12.

PARKES, D. C., and L. UNGAR. 2000. Iterative combinatorial auctions: Theory and practice. *In* Proceedings of the National Conference on Artificial Intelligence (AAAI), Austin, TX, pp. 74–81.

PENN, M., and M. TENNENHOLTZ. 2000. Constrained multi-object auctions and *b*-matching. Information Processing Letters, **75**(1–2):29–34.

RAIFFA, H. 1982. The Art and Science of Negotiation. Harvard University Press, Cambridge, MA.

RASMUSEN, E. 1989. Games and Information. Basil Blackwell, Oxford.

RASSENTI, S. J., V. L. SMITH, and R. L. BULFIN. 1982. A combinatorial auction mechanism for airport time slot allocation. Bell Journal of Economics, **13**:402–417.

RODRIGUEZ-AGUILAR, J. A., P. NORIEGA, C. SIERRA, and J. PADGET. 1997. FM96.5: A Java-based electronic auction house. *In* Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97).

ROTHKOPF, M. H., A. PEKEČ, and R. M. HARSTAD. 1998. Computationally manageable combinatorial auctions. Management Science, **44**(8):1131–1147.

SANDHOLM, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. *In* Proceedings of the National Conference on Artificial Intelligence (AAAI), Washington, DC, pp. 256–262.

SANDHOLM, T. 1996. Negotiation among Self-Interested Computationally Limited Agents. Ph.D. thesis, University of Massachusetts, Amherst. Available at *http://www.cs.cmu.edu/~sandholm/dissertation.ps*.

SANDHOLM, T. 2000. Issues in computational Vickrey auctions. International Journal of Electronic Commerce, **4**(3):107–129. Special Issue on Applying Intelligent Agents for Electronic Commerce. A short, early version appeared at the Second International Conference on Multi-Agent Systems (ICMAS), pp. 299–306.

SANDHOLM, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence, **135**:1–54. First appeared as an invited talk at the First International Conference on Information and Computation Economies, Charleston, SC, 1998. Extended version appeared as Washington University Department of Computer Science Technical Report WUCS-99-01. Conference version appeared at the International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 1999, pp. 542–547.

SANDHOLM, T., and V. FERRANDON. 2000. Safe exchange planner. *In* Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS), Boston, MA, pp. 255–262.

SANDHOLM, T., and V. R. LESSER. 1995. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. *In* Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, pp. 694–701.

SANDHOLM, T., and V. R. LESSER. 2001. Leveled commitment contracts and strategic breach. Games and Economic Behavior, special issue on AI and Economics, **35**:212–270. Early versions appeared as "Advantages of a leveled commitment contracting protocol" in Proceedings of the National Conference on Artificial Intelligence (AAAI), Portland, OR, pp. 126–133, and as Technical Report 95–72, Computer Science Department, University of Massachusetts at Amherst.

SANDHOLM, T., S. SIKKA, and S. NORDEN. 1999. Algorithms for optimizing leveled commitment contracts. *In* Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), pp. 535–540, Stockholm, Sweden. Extended version: Washington University Department of Computer Science Technical Report WUCS-99-04.

SANDHOLM, T., and S. SURI. 2000. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. *In* Proceedings of the National Conference on Artificial Intelligence (AAAI), Austin, TX. pp. 90–97.

SANDHOLM, T., and S. SURI. 2001. Market clearability. *In* Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), Seattle, WA, pp. 1145–1151.

SANDHOLM, T., and S. SURI. 2002. Optimal clearing of supply/demand curves. *In* 13th Annual International Symposium on Algorithms and Computation (ISAAC) Vancouver, Canada. AI, appeared in AAAI-02 Workshop on Agent-Based Technologies for B2B Electronic Commerce (AAAI Technical Report WS-02-01), Edmonton, Canada, pp. 15–22.

SANDHOLM, T., S. SURI, A. GILPIN, and D. LEVINE. 2001. CABOB: A fast optimal algorithm for combinatorial auctions. *In* Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), Seattle, WA, pp. 1102–1108.

SANDHOLM, T., S. SURI, A. GILPIN, and D. LEVINE. 2002. Winner determination in combinatorial auction generalizations. *In* International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Bologna, Italy. Early version appeared at the AGENTS-01 Workshop on Agent-Based Approaches to B2B, Montreal, Canada 2001, pp. 35–41.

SANDHOLM, T., and X. WANG. 2002. (Im)possibility of safe exchange mechanism design. *In* Proceedings of the National Conference on Artificial Intelligence (AAAI), Edmonton, Canada, pp. 338–344.

SANDHOLM, T., and Y. ZHOU. 2002. Surplus equivalence of leveled commitment contracts. Artificial Intelligence (in press). Early versions appeared in ICMAS-00 and in the AAAI-99 Workshop on Negotiation: Settling Conflicts and Identifying Opportunities.

TENNENHOLTZ, M. 2000. Some tractable combinatorial auctions. *In* Proceedings of the National Conference on Artificial Intelligence (AAAI), Austin, TX.

VICKREY, W. 1961. Counterspeculation, auctions, and competitive sealed tenders. Journal of Finance, **16**:8–37.

WOLFSTETTER, E. 1994. Auctions: An introduction. Technical Report, Humboldt University of Berlin.

WURMAN, P. R., M. P. WELLMAN, and W. E. WALSH. 1998. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. *In* Proceedings of the Second International Conference on Autonomous Agents (AGENTS), Minneapolis/St. Paul, MN, pp. 301–308.