eMediator: A Next Generation Electronic Commerce Server

Tuomas Sandholm

ABSTRACT

This paper presents *eMediator*, a next generation electronic commerce server that demonstrates some ways in which algorithmic support and game theoretic incentive engineering can jointly improve the efficiency of ecommerce. First, its configurable auction house includes a variety of generalized combinatorial auctions, price setting mechanism, novel bid types, mobile agents, and user support for choosing an auction type. Second, its leveled commitment contract optimizer determines the optimal contract price and decommitting penalties for a variety of leveled commitment contracting protocols, taking into account that rational agents will decommit insincerely in Nash equilibrium. Third, its 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 of the three components is based on different types of game theoretic equilibrium analysis, and also required development of new algorithms and GUI designs to make it feasible.

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 some ways in which these techniques can improve ecommerce both in terms of processes and outcomes. The result of our 2-year implementation effort is now available for use on the web at `http://ecommerce.cs.wustl.edu/emediator/`. Three components of *eMediator* are discussed: an auction house, a leveled commitment contract optimizer, and a safe exchange planner. Each one exhibits interesting interplay between algorithms and game theoretic incentive engineering.

2. *eAuctionHouse*

Several successful commercial Internet auction sites exist—such as eBay and Yahoo—and interesting academic auction houses have recently appeared on the Internet [27, 16]. 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. One of the services that *eMediator* provides is a free-to-use Internet auction prototype called *eAuctionHouse*. It allows users from across the Internet to buy and sell goods as well as to set up auctions. It is a third party site, so both sellers and buyers can trust that it executes the auction protocols as stated. It is implemented in Java, with some of the computationally intensive matching algorithms in C++. The information about the auctions is stored in a relational database to increase reliability. To our knowledge, our server is the first—and currently only—Internet auction that supports combinatorial auctions, bidding via graphically drawn price-quantity graphs, and by mobile agents. It also offers a wide range of auction types to be chosen from, and supports the user in that choice. 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, i.e. combinations of items (as is often the case e.g. in electricity markets, equities trading, bandwidth auctions [12], and transportation exchanges [18]), 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 tradeoff. Furthermore, even if lookahead were computationally manageable, usually uncertainty remains about the others' bids because agents do not have exact information about each other, e.g. each others' preferences. This often leads to inefficient allocations where bidders fail to get the combinations they want and get ones they do not.

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 [15, 18, 17, 12]. 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 the impact of possibly getting other, complementary items. This capability is particularly important in illiquid, highly volatile, or non-commoditized markets where it is unsure whether one can acquire the items of a desired bundle

one at a time. Our auction server supports a variety of combinatorial auctions. The following subsections discuss some of them.

2.1.1 OR bids

In the combinatorial auction setting that has been most commonly discussed [17], each bidder can bid on combinations of indivisible items, and her bids are joined with non-exclusive OR, meaning that any number of her bids can be accepted. While combinatorial auctions have the desirable features that they 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 saw the desirability of combinatorial bidding in their bandwidth auctions, but it was not allowed due to perceived intractability of winner determination.

Formally, winner determination with OR bids 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 (it is analogous to weighted set packing which is $\mathcal{NP}$-complete).

Even approximate winner determination is hard if one is interested in worst case guarantees:

PROPOSITION 2.1. *No polytime algorithm can guarantee an allocation within a bound* $\frac{1}{n^{1-\epsilon}}$ *from optimum for any* $\epsilon > 0$ *(unless* $\mathcal{NP}$ *equals probabilistic polytime).*

We proved this via an approximation preserving reduction from maximum clique (which is known to be inapproximable [8]). We present the proof in [20].

If the bids exhibit special structure, better approximations can be achieved in polynomial time [2, 6, 9, 7], but even these guarantees are so far from optimum that they are irrelevant for auctions in practice [20].

Polynomial time winner determination can be achieved by restricting the combinations on which the agents are allowed to bid [17]. However, because the agents may then not be able to bid on the combinations they want, similar economic inefficiencies prevail as in the non-combinatorial auctions.

We recently generated another approach to optimal winner determination. 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. While the worst case complexity is exponential (unless $\mathcal{P} = \mathcal{NP}$), the algorithm scales up very well in practice. For details and experimental results, see [20]. This algorithm has been further refined by others [4], and we recently developed a totally different search algorithm that promises to be drastically faster [25].

2.1.2 XOR bids

The above methods for conquering the intractability of winner determination are based on the common assumption 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 current techniques focus on situations where combinational bids are introduced to capture synergies (positive 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 a new bid type, *XOR bid*, i.e. a bid on multiple combinations such that only one of the combinations can get accepted. *eAuctionHouse* supports XOR bids. This allows the bidders to express general preferences with both positive and negative complementarities.

Optimal winner determination with XOR bids is at least as hard as the basic winner determination problem because the latter is a special case of the former. Therefore, the negative results, $\mathcal{NP}$-hardness and inapproximability, apply to this setting as well. Our winner determination algorithm for OR bids extends to this setting by inserting the extra constraints that no two combinations from the same bid can be accepted [20]. These extra constraints will actually make the algorithm faster because the constraints prune some allocations. Therefore, for a given number of bids, winner determination tends to be faster for XOR bids than for OR bids.

To allow an efficient allocation to be reached, it would be desirable to extract truthful valuation revelations as the bids to the auctioneer. Bidding truthfully can be made incentive compatible (a dominant strategy) by using the *Groves-Clarke mechanism* [5, 3]. This means that each bidder is motivated to bid truthfully irrespective of what the others bid. This renders counterspeculation unnecessary. The Groves-Clarke 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. 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 done only approximately.[1]

2.1.3 OR-of-XORs bids

In addition to the tractability of winner determination, the convenience of using combinatorial auctions is another important issue. While XOR bids allow the bidder to express general preferences, in the worst case this would involve placing a bid for each of the $2^m - 1$ possible combinations, 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. One idea toward this direction was presented early on by Rassenti et al [15]. They allowed the bidder to place combinational bids and to state the maximal number of combinations that could be accepted. An XOR bid can be viewed as a special case of this where that number is one. In our auction server, we allow the user to submit multiple XOR bids. We do this in table form, where each row is a combinational bid, and the rows are combined with XOR (Figure 1). These multiple bids are combined together with a non-exclusive OR. We do this by allowing the user to submit multiple tables. Winners can be determined with the same algorithm as in the case of XOR bids by using exclusivity constraints only between combinations that are in the same table. This method maintains full expressive power, but leads to shorter (at least no longer) input descriptions than XOR bids only. We also believe that it is a more natural way to input preferences. Other enrichments to the language are also possible. Since even basic XOR bids have full expressive power, expressiveness should be viewed as a necessary but not a sufficient condition in designing combinatorial auctions. Between fully expressive input languages the appropriate comparison criterion is the convenience of their use in the particular application domain in question.

2.1.4 Other combinatorial auction generalizations

Our input representation (Figure 1) also allows combinatorial double auctions instead of just single-sided auctions. In other words, there can be multiple buyers and multiple sellers. In addition to double auction extensions, it allows combinatorial auctions where the agents can bid for multiple units of each item in a combination (the number of units is specified in each cell of the table). The latter splits into two cases depending on whether the matches of units have to be exact or whether partial matches (e.g. I am willing to pay a certain amount for this vector of items, *or for more*) are allowed. We have also developed optimal winner

[1] A recent paper shows that under certain conditions, incentive compatibility can be guaranteed even under approximate winner determination [10].

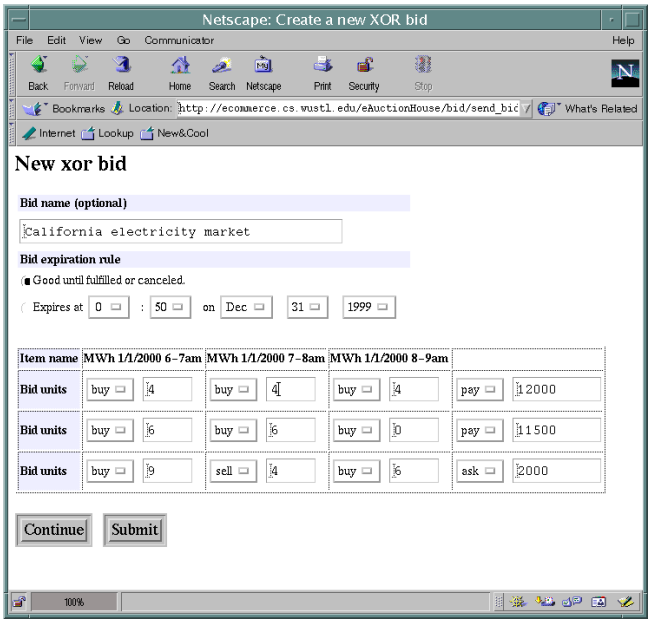

Figure 1: An XOR bid. The example is from an electricity market scenario where the agents can bid for combinations of electricity for different hours of the day, and for multiple Mega Watt hours for each hour of the day. In this example, a refinery operator wants consecutive hours of electricity for 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.

determination algorithms for these settings [25].

2.2 Bidding via price-quantity graphs

Price-quantity graphs are supported 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.[2]

In our implementation, the curves are piecewise linear both for 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 aggregate solution that maximizes the unit price under the constraint that not more is demanded than is available. Each bidder then gets the amount that she bid at that unit price. In double auctions, both supply and demand curves are separately aggregated, and any one of the points were supply meets demand is chosen. If the curves were noncontinuous, it would be possible that no match exists. This holds both

[2] Optimark Technologies, Inc. does a fuzzier match where each bidder specifies how much she prefers different regions of the price-quantity plane [11].

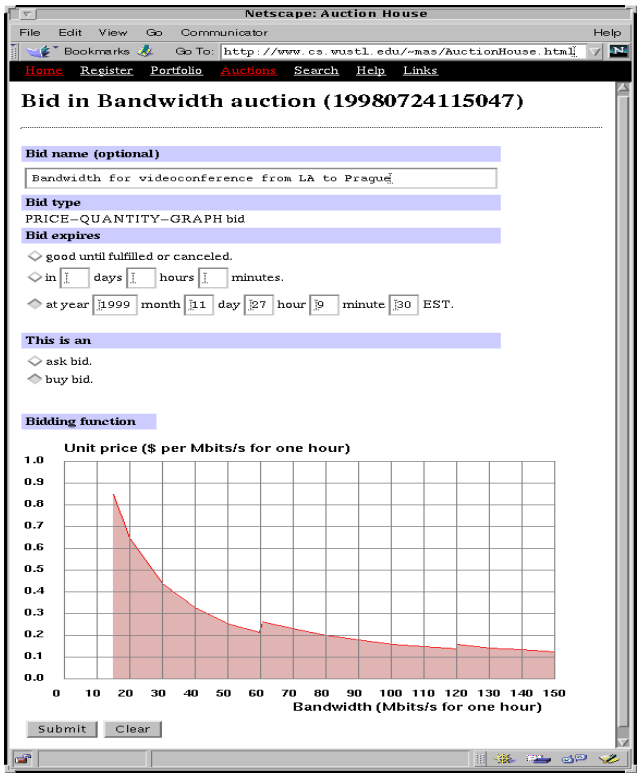

Figure 2: A price-quantity graph allows the user to express continuous preferences in the auction server of eMediator. This figure corresponds to the user being able to hold a video-conference at three alternative picture resolutions requiring a bandwidth of 15, 60, or 120 Mbps. The auctioned item could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

for single and double auctions. To prevent this, we use continuous curves, i.e. the slope of each linear segment is finite. We do not assume that the curves are monotonic (Figure 2).

2.3 Support for choosing an auction type

The auction server 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 schemes should be used. The auction setting differs based on whether it is a single or double auction, whether there is one or multiple items, and whether there is one or multiple units of each item. Furthermore, the units can be divisible or indivisible. The bid types include a regular price bid where the user 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

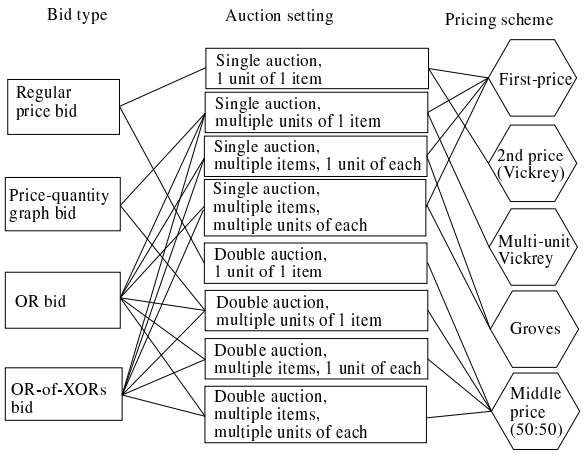

Figure 3: Expertise showing valid combinations of choices of some of the parameters of *eAuctionHouse*.

of her bid. This scheme leads to underbidding. In single unit ascending open-cry auctions, each bidder's dominant 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.

The second-price (Vickrey) auction charges the winning bidder the price of the second highest bid. Under certain restrictions [19], it is each bidder's dominant strategy to bid her true valuation [26].

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. Each bidder can submit multiple bids. The units are assigned from highest bid downward until they run out. Each winning bid is charged the price of the bid that it displaces from the set of winning bids. This achieves incentive compatibility, i.e. each bidder is motivated to bid truthfully, independent of what others bid. Another possible generalization of the Vickrey auction for this setting is to charge every bidder the price of the highest bid that just did not win. Our auction server uses the former method because the latter is not incentive compatible: it falls prey to demand reduction lies by the bidders [1].

For multi-item auctions (with one or more units per item), the Groves-Clarke mechanism, 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 half way 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.

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 [27] 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 where 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 non-programmers 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 email 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 dominant 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 dominant 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 [14]. 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 by new events, or in non-private value auctions, by what others have bid.

Unlike current electronic commerce 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.

3. *eCommitter*: A leveled commitment contract optimizer

eMediator includes an optimizer for leveled commitment contracts. Normal full commitment contracts are unable to take advantage of the possibilities that such 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, i.e. *ex post*. Similarly, a contract may have too low 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 [13]. 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 [23]. 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—i.e. to be freed from the obligations of

the contract—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 [23].

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.[3] 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 fall-back 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 fall-back payoff.[4] The variables $\breve{a}$ and $\breve{b}$ are assumed 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 $b \geq 0$, i.e. agents do not get paid for decommitting. 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

[3]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 off-line or becoming back on-line.

[4]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.

One concern is that a rational self-interested agent is reluctant in decommitting 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. [23] 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

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

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

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

3.2 Sequential decommitting games

In our sequential decommitting game, 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 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

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

3.3 Simultaneous decommitting games

In our simultaneous decommitting games, agents have to reveal their decommitment decisions simultaneously. We first discuss the variant where both have to pay the penalties 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 already ruled out contracts where an agent gets paid for decommitting. On the other

hand, this equates to

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

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

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

In another type of simultaneous decommitting game, 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 already ruled out contracts where an agent gets paid for decommitting. On the other hand, this equates to

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

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 ruled out contracts where an agent gets paid for decommitting. However, this equates to

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

3.4 Contract optimizer implementation

For each game, calculating the Nash equilibria amounts to solving the simultaneous equations (3) and (4) which use (1) and (2). Given an equilibrium, it is easy to compute the agents' expected payoffs under the contract. Furthermore, we have developed algorithms for choosing the contract price and the decommitting penalties in a way that maximizes the sum of the agent's expected payoffs, and divides the gains fairly, or in any other way as long as both parties benefit [24]. That optimization algorithm takes into account that the agents decommit insincerely in Nash equilibrium. To begin, the user inputs f and g graphically or textually. The contract is optimized separately for each one of the protocols, which vary based on who has to reveal her decommitting decision first—the simultaneous protocols are also considered—and whether or not the agents have to pay the penalties if both decommit. The optimal contracts for each protocol are then presented to the user, see `http://ecommerce.cs.wustl.edu/eCommitter`.

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. For example, a shopping 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. This could be avoided by 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 i-Escrow Inc. and Trade-Direct is the cost of running such an intermediary, which is recovered as fees - currently about 5% of the contract price - from the contract parties.

A method for tackling this problem without third parties was developed by [22]. 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. Similarly, some sequences of delivering given chunks are safe while others are not.

As part of *eMediator*, we built a safe exchange planner called *eExchangeHouse*. In the case where a single divisible good 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. In the case of multiple distinguishable goods, the user lists how many goods are to be exchanged, and how many units of each good. For each possible state of the exchange (units of good 1 delivered $\times$ units of good 2 delivered $\times$ ...) the user inputs the the buyer's valuation and the seller's cost. Combining indistinguishable goods into units of a single good significantly reduces the state space since within each good, only the number of units delivered matters, not which ones. Finally, the user inputs how much gain the seller is willing to forego to avoid possible reputation costs from defecting in the exchange, and similarly, how much gain the buyer is willing to forego. 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 chunking algorithms and the chunk sequencing algorithms are highly nontrivial, and are omitted here due to limited space [22, 21]. *eExchangeHouse* is available for use at `http://ecommerce.cs.wustl.edu/eExchangeHouse`.

5. CONCLUSIONS

The *eMediator* prototype exemplifies several new features that can facilitate more efficient ecommerce in the future. Its configurable auction house includes a variety of generalized combinatorial auctions, price setting mechanism, bidding methods, mobile agents, and user support for choosing an auction type. The leveled commitment contract optimizer determines the optimal contract price and decommitting penalties for a variety of leveled commitment protocols, taking into account that rational agents will decommit insin-

cerely. Finally, 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 one of the three mechanisms exhibits an interesting interplay between algorithms and game theoretic incentive engineering. 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.

6. 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.

7. REFERENCES

[1] L. M. Ausubel and P. C. Cramton. Demand reduction and inefficiency in multi-unit auctions. Technical Report 96-07, University of Maryland, Department of Economics, 1996.

[2] B. Chandra and M. M. Halldórsson. Greedy local search and weighted set packing approximation. In *10th Annual SIAM-ACM Symposium on Discrete Algorithms (SODA)*, pages 169–176, Jan. 1999.

[3] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[4] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, Aug. 1999.

[5] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[6] M. M. Halldórsson. Approximations of independent sets in graphs. In K. Jansen and J. Rolim, editors, *The First International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–14, Aalborg, Denmark, July 1998. Springer LNCS 1444.

[7] M. M. Halldórsson and H. C. Lau. Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and 3-coloring. *Journal of Graph Algo. Applic.*, 1(3):1–13, 1997.

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

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

[10] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 96–102, Denver, CO, Nov. 1999.

[11] W. A. Lupien and J. T. Rickard. Crossing network utilizing optimal mutual satisfaction density profile. US Patent 5,689,652, granted Nov. 18, 1997.

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

[13] H. Raiffa. *The Art and Science of Negotiation.* Harvard Univ. Press, Cambridge, Mass., 1982.

[14] E. Rasmusen. *Games and Information.* Basil Blackwell, 1989.

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

[16] J. A. Rodriguez-Aguilar, P. Noriega, C. Sierra, and J. Padget. FM96.5: A Java-based electronic auction house. In *In Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'97)*, 1997.

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

[18] T. W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262, Washington, D.C., July 1993.

[19] T. W. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS)*, pages 299–306, Keihanna Plaza, Kyoto, Japan, Dec. 1996.

[20] T. W. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 542–547, Stockholm, Sweden, 1999. Extended version first appeared as Washington Univ., Dept. of Computer Science, tech report WUCS-99-01, January 28th.

[21] T. W. Sandholm and V. Ferrandon. Safe exchange planner. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, pages 255–262, Boston, MA, July 2000.

[22] T. W. Sandholm and V. R. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 694–701, Montreal, Canada, Aug. 1995.

[23] T. W. Sandholm and V. R. Lesser. Advantages of a leveled commitment contracting protocol. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 126–133, Portland, OR, Aug. 1996.

[24] T. W. Sandholm, S. Sikka, and S. Norden. Algorithms for optimizing leveled commitment contracts. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 535–540, Stockholm, Sweden, 1999. Extended version: Washington University, Department of Computer Science technical report WUCS-99-04.

[25] T. W. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 90–97, Austin, TX, 2000.

[26] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[27] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Proceedings of the Second International Conference on Autonomous Agents (AGENTS)*, pages 301–308, Minneapolis/St. Paul, MN, May 1998.