

On the Sensitivity of Incremental Algorithms for Combinatorial Auctions

Ryan Kastner

Christina Hsieh

Miodrag Potkonjak

Majid Sarrafzadeh

{kastner,thsieh,miodrag,majid}@cs.ucla.edu

Computer Science Department

University of California, Los Angeles

Los Angeles, CA 90095-1596

Abstract

Despite the large amounts of runtime needed to adequately solve a combinatorial auction (CA), existing iterative CA auction protocols require winner determination during every round of bid submissions. Using existing algorithms for winner determination will cause a timing bottleneck during the winner determination phase. Furthermore, there has recently been work which models the formation of supply chains through auctions. Here, winner determination is used for supply chain formation. As supply chains become more dynamic, there is a need for incremental algorithms that quickly and accurately restructure the supply chain while keeping the initial supplier/producer/consumer constraints satisfied.

In this work, we look into the process of quickly and efficiently handling incremental changes in combinatorial auctions. Given some perturbations, we illustrate the tradeoff between preserving the previous solution while maximizing the valuation of the auction. Our results show that it is possible to use a locally optimal solution while sacrificing little solution quality compared to a globally optimal solution. When we have 5000 bids or more, the local algorithm gives a solution within 2% of optimal on the “matching” benchmark from the Combinatorial Auction Test Suite (CATS). Therefore, a simple, fast algorithm can be used to handle incremental changes in CAs with a large number of bids.

1 Introduction

Auctions provide a relatively efficient way to allocate items among agents. Auctions can be used both when the agents cooperate and when the agents are self-interested. Traditional auctions sell items one at a time. Such a protocol – called a sequential auction – can lead to an inefficient allocation of the items. Consider a bidder who has a preference over bundles (combinations) of items. The valuation of an item will depend on the price of items that will be

sold in future auctions. Therefore, the bidder must speculate what others bid in the future. This necessitates the need for look-ahead which – depending on the number of items – is intractable [19]. Moreover, due to the uncertainty of others, a bidder may obtain combinations of items that he or she does not want - an inefficient solution.

Many auctions involve the selling of a large number of items where bidders have preference over bundles. The FCC spectrum auction [6, 15], auctions for airport time slots [16], railroad segments [14] and carrier-of-last-resort responsibilities for universal services [3] are some of the many examples. Due to complementary effects (likewise substitution effects) between different items, a bidder may have a preference for more than one item. Moreover, the bidder may have different valuations for different sets (or bundles) of items. Because of this, the economic efficiency may be enhanced if bidders are allowed to bid on combinations of different assets [17]. Additionally, there is no longer a need for lookahead [19]. But this doesn't come without a price; the auctioneer must determine the winners. Winner determination – i.e. determining what items each bidder receives – is a non-trivial task.

A *combinatorial auction* (CA) allows bidders to place bids on combinations of items. Winner determination for CA is an NP-Hard problem¹. Recently, CA have been given a lot of attention in the research fields of economics and computer science.

Many researchers have focused on optimal algorithms and heuristics for winner determination. Rothkopf *et al.* propose a dynamic programming solution [12]. Sandholm takes a highly optimized search approach [18]. Layton-Brown develops several optimal algorithms including CASS, VSA and CA-MUS [9, 23]. Yet to our knowledge, there is no research on incremental algorithms for combinatorial auctions.

Given an initial solution and perturbation, an incremental algorithm calculates a new solution. Specifically, an incre-

¹NP-completeness is by transformation from the set packing problem.

mental algorithm for CA would find a new set of winners given an initial instance of the problem, the original set of winning bids and perturbation. The perturbation may be a change in the items of a bid, a bid that drops out, a change in the valuation of a bid, etc. Ideally, an incremental algorithm would execute quickly and produce an optimal solution. An incremental combinatorial auction algorithm would be useful in a variety of applications, more specifically, in iterative combinatorial auctions and supply chain formation.

An *iterative combinatorial auction* is a progressive combinatorial auction where agents bid on a set of items while the auctioneer increases the prices of the items. A *combinatorial auction protocol* is a set of rules which govern the behavior of the iterative CA. The rules include stopping conditions, reporting of prices after every round, conditions for withdrawing bids, etc. The rules are usually formulated such that the auction converges to some kind of optimality or game-theoretic equilibrium. There are several CA protocols including AkBA [13], iBundle [2] and the Generalized Vickrey Auction [4]. For each of these protocols, the winner determination problem must be solved at every stage of the auction. Depending on the time between rounds and the number of bids, it may be impossible to use an optimal algorithm to solve the problem. An incremental algorithm fits nicely in such a situation since every round introduces a perturbation and the previous winning bids are known.

Economic activity often involves complex relationships between entities over multiple levels of the production cycle. This concept is often called the *supply chain*. There is a great deal of technology focused on supporting and managing supply chains. Most of the technology assumes that the supply chain has already been formed. Yet, in order to achieve the visions of “virtual corporations” where business dealings are dynamically created and dissolved, we need automated support for *supply chain formation* [21]. Recently, Walsh, Wellman *et al.* have proposed using auctions for supply chain formation [20]. Specifically, they suggest using single item auctions to mediate the negotiation and determine prices and allocations for the supply chains. They show that the use of single item auctions can lead to an inefficient allocation of goods. Later, they extended their work to incorporate CAs for supply chain negotiation [22]. Using CA guarantees that there will be no inefficient allocations of goods.

If we use CA for supply chain formation, the companies involved in the final allocation will have their bids selected by the winner determination algorithm (see Section 2.3 for further details). As supply chains become more dynamic, companies increasingly enter and drop out of the supply chain. Using the CA formulation, a change in the supply chain corresponds to a change in the bid. Therefore, incremental algorithms could be used to reformulate the supply chain solution.

In this work, we study the properties of incremental algorithms for combinatorial auctions. We formulate an integer linear program (ILP) for the incremental algorithm. Using the ILP formulation, we experimentally determine the tradeoff between the quality of solution and reallocation of bids.

In Section 2 we give basic information pertaining to combinatorial auctions, incremental algorithms and supply chain formation. We present an integer linear program (ILP) for incremental CA algorithms in Section 3. In Section 4 we report on experimental results using the ILP incremental algorithm. We conclude and discuss future work in Section 5.

2 Preliminaries

2.1 Combinatorial Auction Formulation

The Combinatorial Auction problem can be formulated as an integer linear program as follows:

Let N be the set of bidders, M the set of m distinct objects and B be the set of b bids. A bid is a tuple which consists of a set of items $S \subseteq M$ and a valuation v for those items. The valuation is the amount that the bidder is willing to pay for the items S . We assume that every bidder has only one bid.

Let $x_j = \{0, 1\}$ be a decision variable that determines whether bid j is selected as a winner.

$$x_j = \begin{cases} 1 & \text{if bid } j \text{ is selected as a winner} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Let $c_{ij} = \{0, 1\}$ be a decision variable relating item i to bid j .

$$c_{ij} = \begin{cases} 1 & \text{if item } i \text{ is in bid } j \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Given this, winner determination for CA can be formulated as:

$$\max \sum_{i=1}^b v_i x_i \quad (3)$$

such that

$$\sum_{j=1}^b c_{ij} x_j \leq 1, i = 1, 2, \dots, m \quad (4)$$

We are trying to maximize the total amount of valuation (price paid) while insuring that every item is sold at most once. It should be noted that this formulation is an instance of the Set Packing Problem (SPP) which is known to be NP-Complete [10]. Therefore, we can formulate any instance of the SPP to an instance of the winner determination problem. In turn, the winner determination problem is NP-complete.

A comprehensive survey of the relationship between SPP, winner determination and NP-completeness can be found in [17].

2.2 Incremental Algorithms

A simple and general model for incremental optimization [1] is:

- An *original instance*, I_0 , is solved by a *full algorithm* to yield a solution S_0 .
- Perturbed instances, I_1, \dots, I_n are generated one by one in sequence.
- Each perturbed instance is solved by an *incremental algorithm* which uses S_{i-1} as the starting point for finding solution $S_i, i = 1, \dots, n$.

A trivial incremental algorithm would simply take the perturbed instance and solve it using the full algorithm. If the full algorithm was optimal, then this would generate an optimal solution. In general, a full algorithm has long runtime and – depending on the amount of perturbation – will produce a result that is similar to the initial unperturbed solution. Therefore, we could take the initial solution, make some modifications based on the perturbation and generate a new, valid solution much faster than rerunning the full algorithm. This is the notion of *iterative optimization* i.e. generate a new solution from a previous solution [7]. Iterative optimization is heart of many optimization algorithms e.g. simulated annealing, genetic algorithms, etc. In general, incremental algorithms trade off quality of solution for fast runtime.

Perturbations for combinatorial auctions can be generalized into four categories:

1. A bidder retracts their bid. This corresponds to removing the bid from consideration.
2. A bidder changes the valuation on their bid.
3. A bidder prefers a different set of items. The set of items in the bid is changed. Additionally, the valuation of the bid may change
4. A new bidder enters the bidding process.

2.3 Combinatorial Auctions and Supply Chains

Walsh *et al.* [22] show that supply chains can be formed using combinatorial auctions. They use a *task dependency network* and formulate bids as the union of the goods an agent (company) wishes to sell and the materials the agent must procure in order to manufacture such goods. The following section briefly summarizes their work.

A task dependency network is a directed, acyclic directed graph (DAG), $G(V, E)$, representing the dependencies among agents and goods. $V = G \cup A$, where G is the set of goods and $A = C \cup \Pi$ is the set of agents. Agents are divided into two sub-sets, the consumers, C , and the producers, Π . The edges, E , connect the agents and the goods which they consume or produce. A directed edge from agent to good, (a, g) corresponds to the agent producing one unit of that good. An edge in the opposite direction from good to agent, (g, a) , means that the agent consumes one unit of the good.

A *consumer* c tries to obtain one unit of a good and obtains a value v_c if the transaction completes. A *producer* π may produce a unit of an output good if the good is consumed and the producer acquires all the necessary input goods. The input goods are a fixed number (zero or more) specified by the in-edges to the producer node. The producer incurs a production cost κ_π to provide an output. An example of a task allocation graph is shown in Figure 1.

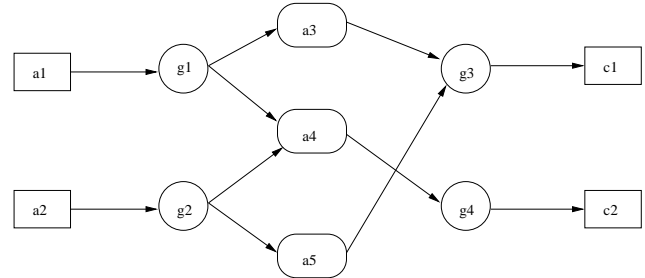


Figure 1. Task Dependency Network. The Circles represent goods, rectangles are agents and arrows denote the use or production of a good.

An *allocation* is a subgraph $G'(V', E') \subseteq G(V, E)$. For every node $g \in G'$, an edge (a, g) signifies that agent a acquires good g . An agent is in the allocation if and only if it acquires or provides a good. A good is in an allocation graph if and only if it is acquired or provided. The value of allocation G' is:

$$value(G'(V', E')) = \sum_{c \in C} v_c(E') - \sum_{\pi \in \Pi} \kappa_\pi(E') \quad (5)$$

An optimal allocation is the allocation with maximum value.

An agent formulates a bid for the combinatorial auction according to the goods it must acquire, wants to produce and the cost associated with each of these goods. Agent a places bid b in the following form:

$$b = \{r_a, \{g_1, q_a^1\}, \dots, \{g_n, q_a^n\}\}$$

where q_a^i is the integer quantity agent a demands (positive for input demands and negative for output demands) of good

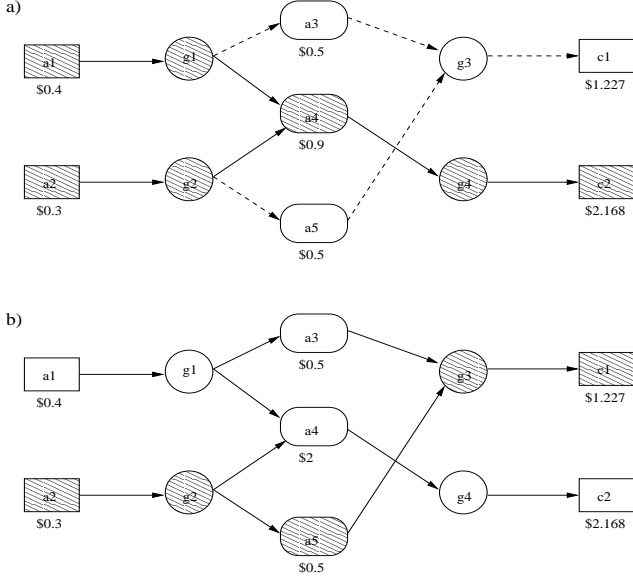


Figure 2. An efficient allocation of a task dependency network. The shaded goods and solid edges are part of the solution. Part a) shows the allocation when producer a4 wishes to sell item g4 for \$0.9. Part b shows the new optimal allocation when a4 increases its price to \$2.

g_i and r_a is its reported willingness to pay (or be paid if the number is negative) for the demanded bundle of goods. For example, producer a_4 in Figure 1 requires inputs g_1 and g_2 to produce output g_4 . The producer wants a payment of 5. It would place the bid $\{-5, \{g_1, 1\}, \{g_2, 1\}, \{g_4, 1\}\}$.

Given the set of bids, a CA would determine the winners. The CA ILP must be formulated a little differently to handle the notion of negative goods. The CA for supply chain formation is formulated as:

$$\max \sum_{b_a \in B} r_a x_a \quad (6)$$

such that

$$\sum_{b_a \in B} q_a^i x_a = 0, i = 1, \dots, n \quad (7)$$

where $x_a = 1$ if agent a wins its bid and $x_a = 0$ otherwise. The winners would form the supply chain. Note that a winning producer is guaranteed to receive every input required to produce its output good. This is a property of an optimal allocation which may not be satisfied if we construct the supply chain via a sequential (single item) auction (see Figure 2).

It may occur that an agent wants to change the price of the good that they sell. This would result in a change in the valuation of that agent's bid. Therefore, the winning bids in an optimal allocation could change. Looking at Figure

2, we can see that if producer a_4 increases their price from \$0.9 to \$2, the allocation will change. An incremental algorithm would be useful in such a case. The change in price is the perturbation; one could consider several other types of perturbations falling into the perturbation categories presented in Section 2.2.

3 Incremental ILP Formulation

In the previous sections, we developed the problem of incremental combinatorial auctions and presented some areas where this problem would be useful. Specifically, we showed that an incremental CA can be used to solve the dynamic supply chain problem and be quite beneficial for iterative CA protocols.

In this section, we give an ILP formulation for the incremental CA problem. The formulation allows us to study some of the properties pertaining to the incremental nature of CA. One such property is the relationship between the winning bids of the previous, unperturbed solution and quality of the solution after perturbation. In certain situations, it is beneficial to maintain the winning bids from the previous solution. Yet, we must also consider the solution quality. Often there is a tradeoff between maintaining previous winning bids and obtaining the optimal solution in terms of maximizing the objective function (Equation 3). This tradeoff is particularly important in the dynamic supply chain problem; we want to maintain the supply chain as much as possible while maximizing the overall profits of the chain. If there is a perturbation in a supply chain e.g. one company drops out, we want to minimize the effect of perturbation on the other companies. A solution which completely reallocates the items of the supply chain to companies which previously were not involved in the chain would not be acceptable, even if this solution is "optimal". Referring to Figure 2, even though producer a_4 increases its price, we may wish to keep the initial allocation (part a) because this is the only way to keep consumer c_2 and producer a_1 in the supply chain. In the optimal solution (part b), both a_1 and c_2 are no longer a part of the supply chain, even though they are not involved the perturbation. We analyze this and other properties and tradeoffs in the next section.

Before we give the incremental CA ILP formulation, we must define some terms. The original, unperturbed instance, I_0 is solved by a full algorithm yielding a solution S_0 . The solution S_0 is comprised of a set of winning bids which we call OW (original winners). I_0 is then perturbed giving an instance I_1 . We wish to find a solution S_1 to the instance I_1 . As previously mentioned, we wish to maximize the number of OW in the new perturbed solution S_1 in addition to maximizing the total valuation v_i of the selected winning bids. Also, we introduce a user defined constant α which allows the user to weight the relationship between maximizing the

valuation of the bids versus retaining previous winning bids.

For each bid $b_i \in OW$ – equivalently for each bid that was selected as a winner in the previous solution S_0 – let $z_i = \{0, 1\}$ be a decision variable that determines whether bid b_i is selected as a winner in the new solution.

$$z_i = \begin{cases} 1 & \text{if bid } b_i \text{ is not selected as a winner in } S_1 \\ 0 & \text{if bid } b_i \text{ is selected as a winner} \end{cases} \quad (8)$$

Consequently, $z_i = 1$ if an original winner (OW) is forced to drop out of the solution. Additionally, let w_i be the weight of z_i . A large w_i indicates that the bid b_i should remain a winner. The w_i allows preference for certain bids to remain as a winner. For example, a company may pay a large sum of money in order to remain in the supply chain; that company would be assigned a large w_i value.

Keep in mind that the incremental solution S_1 is formulated over the perturbed instance I_1 . Yet, we need to know the unperturbed solution S_0 in order to determine the original set of winners OW .

The remaining variables are the same as in the initial CA ILP formulation. Let N be the set of bidders, M the set of m distinct objects and B be the set of b bids in the instance I_1 . A bid is a tuple which consists of a set of items $S \subseteq M$ and a valuation v for those items. The valuation is the amount that the bidder is willing to pay for the items S . We assume that every bidder has only one bid.

Let $x_j = \{0, 1\}$ be a decision variable that determines whether bid j is selected as a winner for the solution S_1 .

$$x_j = \begin{cases} 1 & \text{if bid } j \text{ is selected as a winner} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Let $c_{ij} = \{0, 1\}$ be a decision variable relating item i to bid j .

$$c_{ij} = \begin{cases} 1 & \text{if item } i \text{ is in bid } j \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Given this, winner determination for incremental CA can be formulated as:

$$\max \sum_{i=1}^b v_i x_i - \alpha \cdot \sum_{i=1}^{|OW|} w_i z_i \quad (11)$$

such that

$$\sum_{j=1}^b c_{ij} x_j \leq 1, i = 1, 2, \dots, m \quad (12)$$

$$x_i + z_i = 1, i | b_i \in OW \quad (13)$$

The objective function (Equation 11) maximizes the valuation of the bids while maximizing the number of selected bids which are in the previous solution. If a winning bid in the old solution S_0 is not in the new solution S_1 , then its corresponding z_i variable is 1 which is obviously detrimental in maximizing the objective function. Equation 12

insures that every item is assigned to only one winning bid much like Equation 4 of the original CA ILP formulation. Equation 13 insures that the previous winning bids (in set OW from solution S_0) either win in the solution S_1 or have their corresponding z_i variable set to 1. This equation only holds for the bids which are in the set OW . It need not (and most likely will not) hold for the remaining bids.

4 Experiments

4.1 Benchmarks

The combinatorial auction test suite (CATS) [8] was used to generate goods and bids. We focused on three specific distributions in the test suite: matching, regions, and paths. Each particular distribution is analogous to a real-life scenario. For instance, matching is used to represent the correspondence of time slices on multiple resources, e.g., airport take-off and landing rights. Regions concentrates on problems dealing with adjacency in two-dimensional space, such as drilling rights. Finally, the paths distribution involves the purchase of a connection between two points, for example, determining truck routes.

4.2 Experimental Flow

Figure 3 explains the experimental flow. First, we use CATS to generate a set of items and bids. This is the original, unperturbed instance of the problem, I_0 . The result is converted into an input file for lp_solve [11]. Then, we run the lp_solve to get the optimal objective value from the optimal set of winners. This is the solution S_0 to I_0 . We call the set of winners comprising this solution OW – the original winners. After that, we perturb the original instance I_0 by randomly removing $x\%$ of the winning bids from S_0 . This creates a perturbed instance of the initial problem called I_1 . The bids that are randomly removed are called the voluntary dropouts; we assume that these bidders elected to retract their bids for some reason. After removing these bids, we optimally solve instance I_1 in order to determine the optimal objective value for this perturbed instance. By doing this, we ignore the original winners OW as we do not try to keep them in the solution. Then, we convert the problem into an incremental ILP for winner determination as presented in Equations 11 - 13. We vary α in order to determine the tradeoff between maximizing the objective value and maximizing the number of initial winners OW in the new solution. The weights w_i are all assumed to be 1. The involuntary dropouts are $OW - S_1 - \{\text{voluntary dropouts}\}$. In other words, the involuntary dropouts are the winning bids from the initial instance I_0 who did not voluntarily retract their bids and are not included in the final solution S_1 .

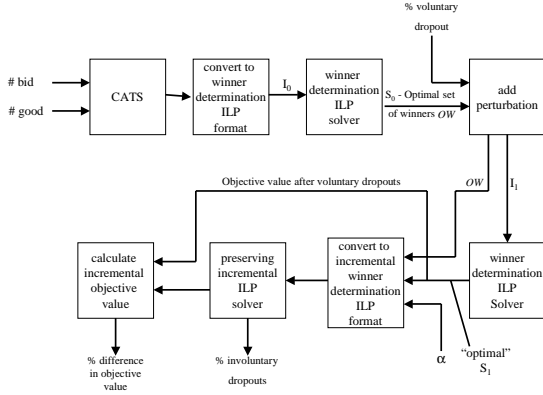


Figure 3. Experimental flow for generating the test examples and solving the winner determination problem.

The percentage of involuntary dropouts is the ratio between the number of involuntary bids and $|OW|$. The percentage difference in objective value is the ratio between the value of the bids from the solution S_1 using the incremental formulation and the optimal value of the bids from the perturbed instance I_1 . Again, the optimal value of the bids is the maximal valuation the auctioneer can achieve by not caring about retaining bids from the initial winners.

Using ILOG CPLEX version 7.0 [5], the largest instances of the ILP formulations were optimally solved in less than 6 seconds². The longest runtime occurred for the incremental ILP formulation for an instance of “matching” with 20005 bids; it took 5.85 seconds.

4.3 Results

The graphs in Figures 4, 5 and 6 display the correlation between percentage of involuntary dropouts and percentage difference in objective value. Data points are generated from varying the α value in the incremental ILP formulation. Each data point is averaged over all the instances (each of which consist of a different numbers of bids and goods) for each of the three distributions, matching, paths and regions. For matching, we ran 35 different instances; the number of bids and goods varied from [25 - 20005] and [53 - 3587], respectively. The number of bids and goods for paths were [100 - 20004] and [29 - 2812], respectively; there were 21 different instances. Finally, regions had 18 different instances with the bids varying from [101 - 10001]

²Our experiments use lp_solve as we just recently obtained a copy of CPLEX. We ran the “largest” instances on CPLEX to get an idea of their runtimes. Both solvers are optimal, therefore the only sacrifice in using lp_solve is its increased runtime. Though we must say that lp_solve performed admirably.

and goods varying from [40 - 1894]. The legend denotes the voluntary dropout rate - the percentage of original winners who chose to retract their bid. Additionally, we ran the distributions with differing amounts of perturbations. The legend represents the amount of bids that voluntarily retracted their bids. These bids were selected at random from the bids in the solution of the initial instance I_0 . For instance, the line 15% denotes that the perturbation from I_0 to I_1 was removing 15% of the initial winning bids from solution S_0 .

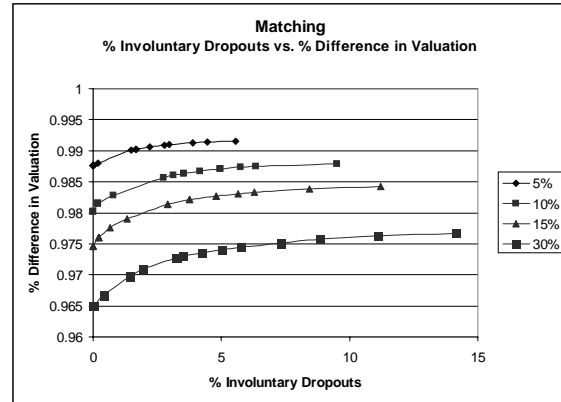


Figure 4. The trade-off between preserving the initial solution and maximizing the valuation of the winning bids for the distribution matching. The chart shows the results from varying the amount of perturbation (from 5% to 30%). Each data point is the averaged over all the instances.

As the percentage of involuntary dropouts increases, difference in total bid valuation increases as well. The optimal solution with regard to bid valuation has no regard for retaining the winners from the unperturbed solution. Therefore, as we start to maintain the original winners OW , we add constraints to the incremental ILP formulation. These additional constraints cause a drop in the total bid valuations. Essentially, this is the price that we pay in order to maintain the original solution.

A dropout rate of 0% denotes that all of the winning bids from I_0 are kept in the solution. This does not include the voluntary dropouts, whose bids were selected as winners in S_0 , but elected to retract their bids. Therefore, this point - the intersection with the y-axis - gives the solution quality of solely replacing the bidders who chose to remove their bids by other bids that were not previously selected as winners. Of course, by removing some of the original winning bids, we can increase the total valuation of the new winning bids. The slope of the line denotes the tradeoff between increasing valuation and removing bidders. Therefore, the larger the slope, the more inclined we are to remove some of the

original winning bids as we can increase our profit³. You can see that even when retaining all of the bids in *OW*, the drop-off in bid valuation stays under 5%. Also, the dropout to valuation tradeoff varies depending on the distribution. For instance, in paths, one can maintain the solution while losing very little solution quality, less than 1%. On the other hand, in order to maintain the solution in regions, one must give up 5% in solution quality.

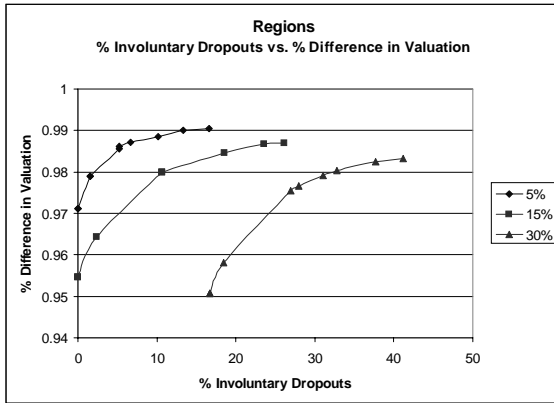


Figure 5. The trade-off between preserving the initial solution and maximizing the valuation of the winning bids for the distribution regions. The chart shows the results from varying the amount of perturbation (from 5% to 30%). Each data point is the averaged over all the instances.

Finally, we comment on the effect that the amount of perturbation has on the solution quality. The different line plots in the graph represent a differing amount of perturbation introduced into the solution. The larger percentage, the larger the perturbation. As expected, when we increase the perturbation, the solution quality decreases. But, it is interesting to note that a large increase in perturbation does not degrade the solution quality by much. For instance, the plotted lines are somewhat parallel. Going from a 5% perturbation to a 30% perturbation reduces the quality by approximately 3% in the matching distribution. As you can see, a similar trend occurs in the other distributions.

The graph in Figure 7 shows the specific example taken from the matching distribution when we keep all of the previous winning bids i.e., the involuntary dropout rate is 0%. We replace the items from the bids that voluntarily dropped out with bids that were not previously selected. We do not consider items from the other winning bids that did not choose to drop out. This is equivalent to rerunning an auction on only the bids that were perturbed and not considering the non-perturbed bids. One can see that if the

³According to the auctioneers point of view.

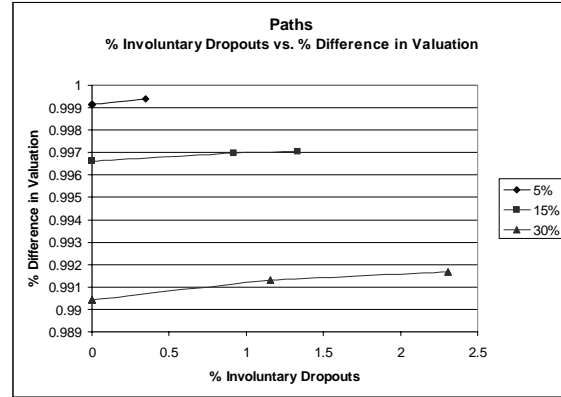


Figure 6. The trade-off between preserving the initial solution and maximizing the valuation of the winning bids for the distribution paths. The chart shows the results from varying the amount of perturbation (from 5% to 30%). Each data point is the averaged over all the instances.

total number of bids is small, then the solution quality of is greatly affected. But, as the number of bids increases, the solution quality is almost as good as in the unperturbed instance I_0 . This general trend occurs across all of the distributions regardless of the perturbation. From this, we can conclude that there is an easy solution to the incremental auction problem when there are a large amount of bids. Simply take the items from the bids that dropped out, and perform a mini-auction on those items (we call those items *DOI*, the drop out items) using the original set of bids. Of course, only bids that are a subset of *DOI* are included in this auction. Unfortunately, when the total number of initial bids is small, we must use a more complex algorithm that removes previously winning bids in order to achieve a good valuation.

5 Conclusion

In this work, we studied a solution for the incremental winner determination problem. First, we discussed the need for incremental combinatorial auctions (CA) in CA protocols and supply chain formation. In many current CA auction protocols, the winners are selected during each round. Depending on the number of items and bids, it may not be feasible or necessary to run a full algorithm to determine the optimal set of winners after each round. Additionally, a CA formulation for supply chain formation was recently proposed [22]. As supply chains become more and more dynamic, incremental solution that preserves, as much as possible, the original supply chain formation is needed. A CA incremental algorithm can handle such cases.

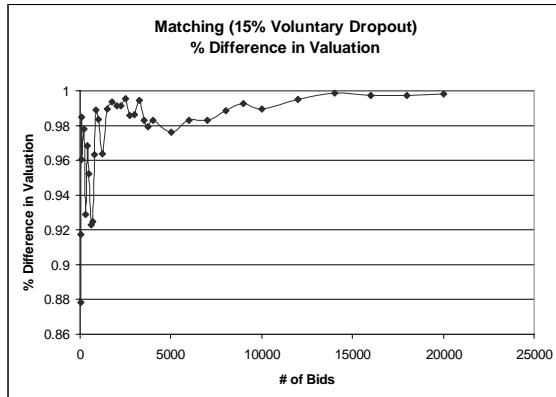


Figure 7. The effect on the winning bids total valuation when we fully preserve the initial solution.

We formulated an incremental ILP for the winner determination problem. According to the user's specification, the ILP simultaneously maximizes the valuation of the winning bids while preserving the previous solution. We explored this tradeoff using distributions from the combinatorial auction test suite. We presented data that examined the degradation in the amount of solution quality that one must sacrifice in order to maintain the initial solution. We showed that if there is a large number of bids, a greedy algorithm that simply reallocates the perturbed goods may be used with little effect on the optimality of the solution.

In the future, we plan to develop methods that create fault tolerant solutions to the winner determination problem. We envision that these methods will create an initial solution that can quickly and locally handle any future perturbations.

References

- [1] Andrew Kahng and Stefanus Mantik. "On Mismatches Between Incremental Optimizers and Instance Perturbations in Physical Design Tools". In *Proc. IEEE International Conference on Computer Aided Design*, November 2000.
- [2] David Parkes and Lyle Ungar. "Iterative Combinatorial Auctions: Theory and Practice". In *National Conference on Artificial Intelligence*, 2000.
- [3] Frank Kelly and Richard Steinberg. "A Combinatorial Auction with Multiple Winners for Universal Services". Technical Report Technical Report, University of Cambridge, 1998.
- [4] H. Varian and J.K. MacKie-Mason. "Generalized Vickrey Auctions". Technical report, University of Michigan, 1995.
- [5] ILOG, Inc. ILOG CPLEX. <http://www.ilog.com/products/cplex/>.
- [6] John McMillan. "Selling Spectrum Rights". *Journal of Economic Perspectives*, 1994.
- [7] K.D. Boese. "Models for Iterative Global Optimization". PhD thesis, Computer Science Department, University of California, Los Angeles, 1996.
- [8] Kevin Leyton-Brown, Mark Pearson and Yoav Shoham. "Towards a Universal Test Suite for Combinatorial Auction Algorithms". In *ACM Conference on Electronic Commerce*, October 2000.
- [9] Kevin Leyton-Brown, Yoav Shoham and Moshe Tennenholtz. "An Algorithm for Multi-Unit Combinatorial Auctions". In *National Conference on Artificial Intelligence*, 2000.
- [10] M. Garey and D. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness". W.H. Freeman and Company, New York, NY, 1979.
- [11] Markus Weidenauer. `lp_solve`. ftp://ftp.es.ele.tue.nl/pub/lp_solve.
- [12] Micheal Rothkopf, Aleksandar Pekac and Ronald Harstad. "Computationally Manageable Combinatorial Auctions". *Management Science*, 1998.
- [13] Peter Wurman and Micheal Wellman. "AkBA: A Progressive, Anonymous-Price Combinatorial Auction". In *ACM Conference on Electronic Commerce*, October 2000.
- [14] P.J. Brewer. "Decentralized Computation Procurement and Computational Robustness in a Smart Market". *Economic Theory*, 1999.
- [15] Preston McAfee and John McMillan. "Analyzing the Airwaves Auction". *Journal of Economic Perspectives*, 1996.
- [16] S.J. Rassenti, V.L. Smith and R.L. Buffin. "A Combinatorial Auction Mechanism for Airport Time Slot Allocation". *Bell Journal of Economics*, 1982.
- [17] Sven de Vries and Rakesh Vohra. "Combinatorial Auctions: A Survey". Kellogg School of Management, Northwestern University. Evanston, IL, Department of Managerial Economics and Decision Sciences, 2000.
- [18] Toumas Sandholm. "An Algorithm for Optimal Winner Determination in Combinatorial Auctions". In *The International Joint Conferences on Artificial Intelligence Workshop on Agent-Mediated Electronic Commerce*, August 1999.
- [19] Toumas Sandholm. "Approaches to Winner Determination in Combinatorial Auctions". *Decision Support Systems*, 1999.
- [20] William Walsh and Micheal Wellman. "Efficiency and Equilibrium in Task Allocation Economies with Hierarchical Dependencies". In *The International Joint Conferences on Artificial Intelligence Workshop on Agent-Mediated Electronic Commerce*, August 1999.
- [21] William Walsh and Micheal Wellman. "Modeling Supply Chain Formation in Multiagent Systems". In *The International Joint Conferences on Artificial Intelligence Workshop on Agent-Mediated Electronic Commerce*, August 1999.
- [22] William Walsh, Micheal Wellman and Fredrik Ygge. "Combinatorial Auctions for Supply Chain Formulation". In *ACM Conference on Electronic Commerce*, October 2000.
- [23] Yujo Fujishima, Kevin Leyton-Brown and Yoav Shoham. "Taming the Computational Complexity of Combinatorial Auctions: Optimal and Approximate Approaches". In *The International Joint Conferences on Artificial Intelligence Workshop on Agent-Mediated Electronic Commerce*, August 1999.