INDIAN INSTITUTE OF MANAGEMENT CALCUTTA

WORKING PAPER SERIES

WPS No. 627/ August 2008

Towards Designing Proxy Bidders for Online Combinatorial Auctions

by

Soumyakanti Chakraborty
Doctoral student, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700104, India

Anup K. Sen
Professor, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700104, India

&

Amitava Bagchi
Professor, Indian Institute of Science, Education and Research, Kolkata, IIT Kharagpur Extension Centre, HC Block, Sector III Saltlake, Kolkata – 700106, India

Towards Designing Proxy Bidders for Online Combinatorial Auctions

Soumyakanti Chakraborty
Indian Institute of Management Calcutta, Joka, Diamond Harbour Road, Kolkata 700104
fp072004@iimcal.ac.in

Anup Kumar Sen
Indian Institute of Management Calcutta, Joka, Diamond Harbour Road, Kolkata 700104
sen@iimcal.ac.in

Amitava Bagchi
Indian Institute of Science, Education and Research, Kolkata, IIT Kharagpur Extension Centre, HC Block, Sector III Saltlake, Kolkata 700106,
bagchi@iimcal.ac.in

Abstract

The last few years have witnessed a high rate of growth in revenue from single-item online auctions popularized by eBay and similar websites. But implementations of online combinatorial auctions are rare because of the high cost incurred by the seller in solving the WDP, and the high participation and monitoring costs that have to be borne by the bidders. Recently, an incremental dynamic programming formulation for the WDP has made this problem more tractable when the number of items is not too large. In an effort to ease the burden of the bidder, we propose the use of a proxy agent that will bid on the bidder's behalf. In our algorithmic scheme, PRACA, a main process coordinates the activities of a number of autonomous proxy agents, one for each bidder, by means of signals. We derive some interesting theoretical results, and describe how we verified the operational correctness of PRACA.

Keywords: Online Combinatorial Auctions; Participation Cost of Bidders; Proxy Agents for Combinatorial Auctions.

1. Introduction

Electronic commerce continues to flourish despite many publicized failures. By eliminating intermediaries, it provides a less expensive way for buyers to acquire goods, particularly specialized collectibles. Extensive listings and powerful search techniques help to reduce transaction costs. At the same time, a fairly large section of buyers find this mode of shopping more entertaining and trouble-free than traditional methods [4]. During the 1st quarter of 2008, E-commerce in the U.S. registered \$33,795 million dollars in retail sales. Although still only 3.3% of total retail sales, the quarter sales had grown by 13.6% in one year, in contrast to total retail sales which had grown by 2.8%. Much of the growth in E-commerce activity can be attributed to online auctions. In 2007, the total value of sold items on eBay's trading platforms was nearly \$60 billion. But online auctions still remain confined to the retail segment of the market because practical and efficient implementations of combinatorial auctions are not yet available. Auctions on eBay and similar websites sell single items only. Business concerns that buy many different goods in bulk would be more interested in combinatorial auctions that sell bundles (*i.e.*, packages) of items, if only a simple and cost effective scheme of operation could be devised.

Single-item auctions are inefficient when the goods being sold are complementary in nature, *i.e.*, when the value of a combination of items is substantially more than the sum of the values of the individual items. This situation arises in real life, for example, in the auctions of: i) take-off and landing slots at an airport [15]; ii) frequency spectrum ([7],[12]); iii) adjacent pieces of real estate [12]; etc. Combinatorial auctions resolve the complementarity problem by allowing agents to bid for bundles of items. But when there are n items, bidders can bid on any subset of the set of 2^n -1 possible non-empty bundles. This makes the Winner Determination Problem (WDP), *i.e.*, the determination of the winning bids, computationally expensive. Although combinatorial auctions have resulted in higher revenues and increased buyer satisfaction in FCC spectrum auctions [11], these advantages have not yet percolated to comparatively commonplace items because it has proved difficult to execute such auctions on line.

An online auction should allow bidders to join and leave the auction at any time. To enable bidders to bid meaningfully, feedback must be provided on the current state of the auction. It is not enough just to display the bid values. A solution to the WDP after every bid would tell prospective bidders what the provisional winning prices are at that instant. To encourage bidder participation, the seller has to provide this information, for we cannot expect bidders to solve the WDP individually. So the cost of running an online auction is high for the seller. The participation and monitoring costs for the bidder are also high.

An interesting way to reduce the seller’s burden by trading time for space has recently been announced [1]. It makes use of a dynamic programming formulation of the WDP [16] from which the provisional winning prices can be derived. But this scheme by itself is not enough to make online combinatorial auctions viable. It is also necessary to reduce the bidder’s burden. One way to do this is by providing a private copy of a proxy agent to each bidder, which would bid on the bidder’s behalf as in single-item eBay auctions. In this paper, we describe a scheme called PRACA (Proxy Agent for Combinatorial Auctions) to show how such an agent can be designed. It will allow a bidder to input lower bounds on valuations of packages. It will then bid on the bidder's behalf, increasing the bid values on packages as and when needed, until the valuations supplied by the bidder are reached. At that time, it will request the bidder to update the valuations, helping her to estimate the revised valuations by displaying the current winning values [1]. We suggest a distributed scheme in which the main process coordinates the activities of autonomous proxy agents using signals. In each cycle, the main process requests a proxy agent from the set of currently active agents to supply a fresh bid. If the selected agent is unable to respond because the current limits on the package valuations have been reached, it asks the bidder to update the valuations. Thus the human bidder no longer needs to monitor bid values continuously.

Section 2 surveys the related technical literature. Sections 3 and 4 describe PRACA in more detail, and section 5 provides a worked out example. Some theoretical results are presented in section 6. PRACA has been simulated on a computer, as explained in section 7. The last section contains some concluding remarks.

2. Related Work

Combinatorial auctions can be implemented in many different ways. In the sealed bid format, each bidder supplies to the seller, within a pre-specified closing time, the valuations for packages of interest to her. The seller solves the WDP and allocates the packages among the winning bidders. Payment is typically 'first price', i.e., a winner pays what she has bid. In the VCG scheme ([18], [6], [8]), devised to encourage truthful bidding, the winning bidders do not pay what they have bid; instead, they pay the marginal negative effect that their participation has on the reported values of other bidders [17]. An alternative to the sealed bid auction is the ascending proxy auction ([3], [13]), in which at the start of the auction, the bidders report their preferred packages and their valuations to a proxy agent that bids in a series of rounds on the behalf of the bidders. The auctioneer determines the provisional winners that would maximize her revenue at every round. At the end of each round, if a particular bidder is not amongst the provisional winners, the proxy agent for the bidder makes new bid for the most preferred package of the bidder given her reported preferences and valuations. This mechanism assumes that the valuations provided by the bidders at the start of the auction do not change in the course of the auction. The two-phase Clock Proxy Auction [2] is an improvement on this idea. In the clock phase, bids are placed on individual items; this allows the bidders to form a reasonable estimate of the price of each item, and helps them to derive valuations of packages. The second phase is an ascending proxy auction. Some of the other schemes, such as PAUSE [9] and AUSM [5] try to relieve the seller of the burden of solving the WDP; this task is transferred to the bidders, and the seller just has to confirm the validity of a bid, which is a computationally tractable problem. None of these methods are suitable for online implementation.

In an online auction, bidders need information feedback after each bid. This issue is addressed in [1] by providing a bidder with two values on each package, the deadness level (DL) and the winning level (WL), which helps her to value packages more accurately. The DL of a package is the minimum value that ensures that the bid stays in contention, while the WL of a package is the minimum value that makes the bid a part of the winning allocation. The knowledge of these values ensures that non-competitive bids are eliminated, so no time is wasted on such bids. The WDP is solved in an incremental manner using a dynamic programming formulation. An extensive table is stored in memory, so the method does not scale up to a large number of items.

3. The Proposed Scheme PRACA

PRACA reduces the participation and monitoring costs of bidders by providing a proxy agent to each bidder. All prospective bidders must be pre-registered. A bidder can log in at any time during an on-going combinatorial auction, and is then provided access to a private copy of the proxy agent software. The bidder inputs into the agent the initial lower bounds on the valuations of the packages of interest, and the agent bids on her behalf. The agent is designed to bid for the highest utility package of the bidder [13]. When the bids reach the given valuations, the agent seeks new valuations from the bidder. A bidder is able to track the progress of the auction at all times.

PRACA performs a number of tasks:

Login and Assignment of Proxy Agent: When a bidder logs in, the proxy agent assigned to her becomes the only source of contact with the auction system. When fresh valuations must be supplied, the bidder can find out the DLs and WLs of packages through the agent. It is assumed here for simplicity that a bidder always makes use of a proxy agent. In practice, we might want to allow bidders to bid on their own without the help of an agent.

Selection of Unhappy Agent for Fresh Bidding: After the bidder logs in, her agent gets added at the end of the queue of active but unhappy agents. An *unhappy* agent is one who is *not* a provisional winner after the immediately preceding bid. At every step, the next proxy agent in the queue is invited to submit a fresh bid. The main process waits until the agent responds.

Bid Selection: The agent invited to submit a fresh bid determines which package has the maximum utility for the bidder at that instant. The utility for bidder *b* and package *p* is determined using the formula:

$$U_{bp} = V_{bp} - (DL_p + \varepsilon)$$

where U_{bp} is *b*'s utility for *p*, V_{bp} is *b*'s current valuation of *p*, DL_p is the current deadness level of *p*, and $\varepsilon > 0$ is the predefined bid increment. We first explain the significance of DL and WL with the help of an example. Let there be two items A and B, and suppose the current maximum bids on the packages A, B and AB are 10, 20 and 25 respectively. Although the maximum bid on AB is 25, the minimum that must be bid on AB to ensure that the bid remains in contention is 30. A bid below 30 is wasted and is known as a 'dead' bid. So DL(AB) = 30; in this case, WL(AB) = 30 also. The utility is computed for all packages of interest to the bidder, and a bid is placed on the package having maximum utility. A tie is resolved in favor of a package having a larger number of items; if there is still a tie, it is broken arbitrarily. If all packages have negative utility, fresh valuations are sought from the bidder.

Computation of DLs and WLs: After an agent submits a fresh bid, the provisional winning allocation is recalculated using the method described in [1]. The DLs and WLs are updated for all the packages and stored in a global database.

Termination: The auction has a pre-specified duration and terminates at the end of this period. Before termination, all ready but unhappy bidders are processed. The winning bids are determined for the final time, and the seller's revenue is computed, assuming payments by winning bidders are made on a first price basis.

PRACA can be implemented using one main (or coordinating) process and a number of autonomous processes, one for each bidder, that serve as proxy agents. The major tasks of the main process include initiating a login process, selecting one of the ready but unhappy proxy agents for the next bid, and determining the new provisional allocation after a fresh bid is submitted. A new bidder is assigned a proxy agent, which then enters the queue of ready but unhappy agents. At any time the main process is in one of six states (see Table 1), and moves from one state to another as shown in the state transition diagram (see Figure 1). It begins in the MInit state where it does the necessary initialization and spawns a login process. It then moves to the MWaitSelect state, where it performs three jobs: i) it handles the signal *AddNewBidderRequest* from a proxy agent by moving to the state MUpdate; (ii) it selects an active but unhappy proxy agent, sends the signal *FreshBidRequest* to it, and moves to the state MRequest; (iii) on arrival of the *GlobalTimeout* signal, it prepares for termination. In the state

Table 1: States of the Main Process

State	**Description**
MInit	Initialize; spawn login process
MWaitSelect	Wait for signal from proxy agents; select the next proxy agent for bidding
MRequest	Wait for a fresh bid to arrive from the selected proxy agent
MCompute	Compute new DLs and WLs of all packages
MUpdate	Update the queue of ready but unhappy agents
MTerminate	Terminate auction, and do final allocation of packages to winning bidders

MRequest, the main process either: i) receives the *FreshBidSend* signal from the selected proxy agent and moves to MCompute to compute the new DLs and WLs of packages; or, ii) receives the *BidderInformed* signal from the proxy agent indicating that updated valuations have been sought (but not yet received) from the bidder. The DLs and WLs of packages are accessible to all agents. Section 4 gives the details of the algorithm.

The proxy agent communicates with the bidder, helping her to supply a shortlist of packages and their valuations to the main process whenever requested to do so. It also displays the DLs and WLs of packages to the bidder. At any time the proxy agent is in one of four states (see Table 2). Its state transition diagram is shown in Figure 2. In the state PInit it initializes variables and spawns a process

WaitforInput which waits to receive input from the bidder, initially the selected packages and their valuations. WaitforInput runs continuously on the bidder terminal, displaying current DLs and WLs of packages at the request of the bidder, and permitting the bidder to submit fresh valuations. The agent moves from the state PInit to the state PWait after sending the *AddNewBidder* signal to the main process. In Pwait, it waits for the signals *FreshBidRequest* or *Terminate* from the main process.

Table 2: States of the Proxy Agent

State	Description
PInit	Initialize; spawn WaitforInput
PSend	Send a bid to the main process
PWait	Wait for a signal from the main process
PTerminate	Terminate

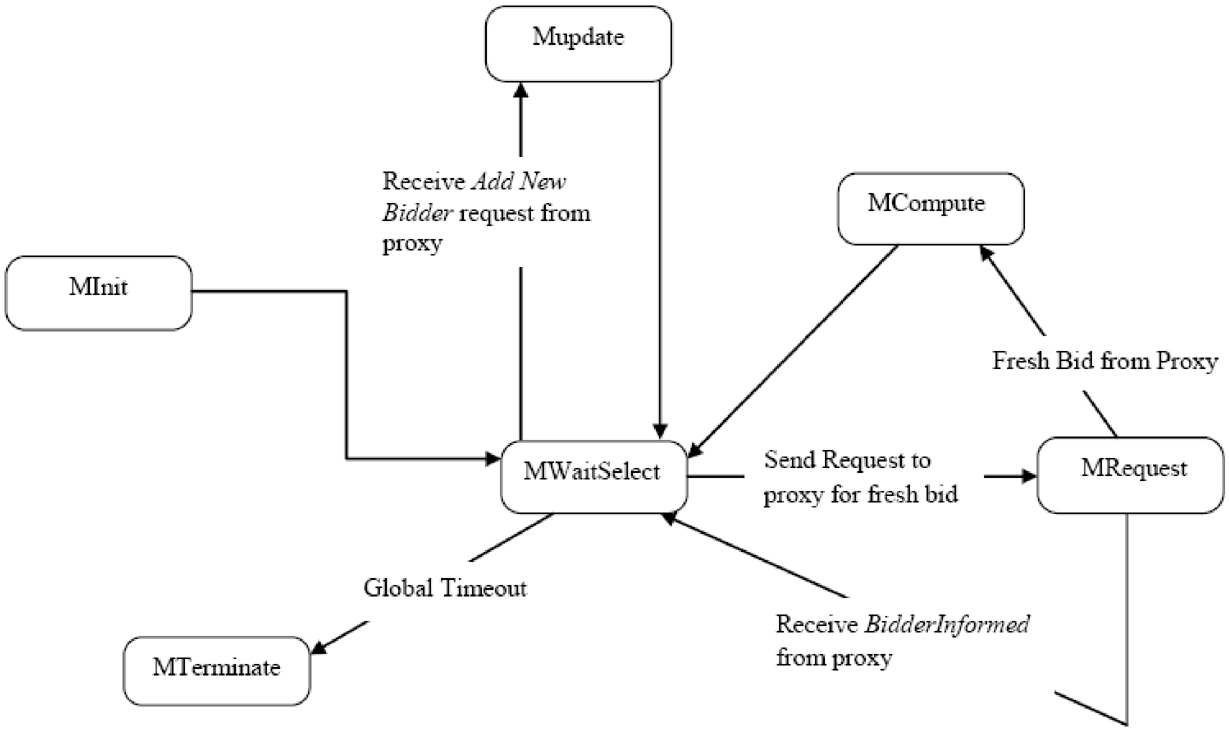
Mupdate
Receive *Add New Bidder* request from proxy
MCompute
MInit
Fresh Bid from Proxy
MWaitSelect
Send Request to proxy for fresh bid
MRequest
Global Timeout
Receive *BidderInformed* from proxy
MTerminate

Figure 1. State Transition Diagram for Main Process

On receiving the former signal, the agent moves to the state PSend. If there is a package of non-negative utility, a bid is sent to the main process; otherwise the bidder is asked to revise the valuations. Either the signal *FreshbidSend* or the signal *BidderInformed* is sent to the main process, and the agent returns to state PWait. On receiving the *Terminate* signal in the PWait state, it moves to state PTerminate for cleanup operations. The algorithm for the proxy agent is given in the next section. The status of each proxy agent in the main process can assume one of the following four values.

After the *GlobalTimeout* signal is received, PRACA does not accept any new bidders, *i.e.*, it does not thereafter process any *AddNewBidder* signals; it also does not accept revisions in valuations from RN agents (see section 4, next column for status of agents). All RU agents in the queue get processed prior to the actual end of the auction, so that finally the queue contains only agents of status RH or RN. Any RH agent whose status changes to RU also gets processed. This is not the only way the auction could have ended. We could have adopted one of the following alternative methods:

- A hard close at the *GlobalTimeout* signal, simply ignoring the agents in the queue;

- After the RU bidders in the queue are processed, a signal could have been sent to all RN agents to bid one last time; bidders who had already revised their valuations would then get a chance to bid.

4. Algorithms: Main Process, Proxy Agent

In the main process, all active proxy agents are in the FIFO queue; the status of each agent takes one of the four values shown below.

RU (Ready Unhappy):	The proxy agent is ready to bid and it is not in the current provisional allocation.
RH (Ready Happy):	The proxy agent is in the current provisional allocation.
EB (Expecting Bid):	The proxy agent has been asked to bid and the main process is waiting for the agent to respond.
RN (Ready Negative Utility):	For this proxy agent, the current utility of every package of interest has a negative value.

Only RU and RN agents are selected for submission of bids; RH and EB agents are skipped over. Unless specified in the algorithm, agents do not change their relative positions in the queue. The main process and the proxy processes communicate using signals as explained earlier. For ease of understanding we list the main signals in Table 3 below.

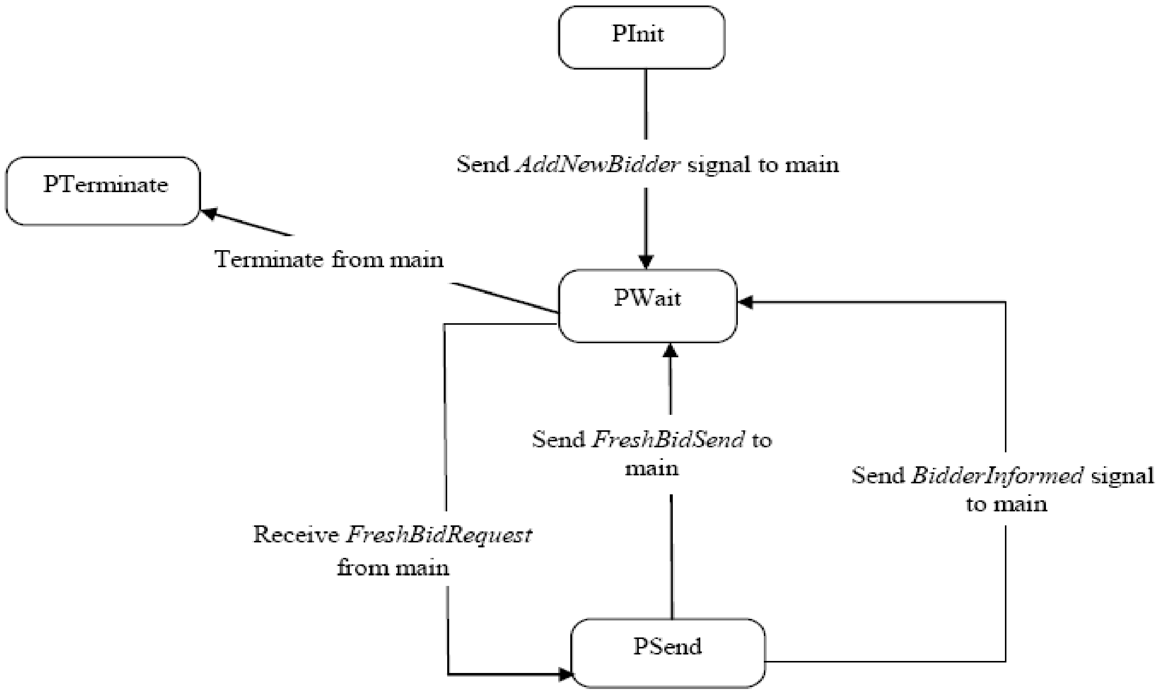
PInit
Send *AddNewBidder* signal to main
PTerminate
Terminate from main
PWait
Send *FreshBidSend* to main
Send *BidderInformed* signal to main
Receive *FreshBidRequest* from main
PSend

Figure 2. State Transition Diagram for Proxy Agent

Table 3: List of Signals

Signal	Description
AddNewBidderRequest	After a bidder logs in and a proxy agent is assigned, the agent sends this signal to the main process, which inserts the agent at the end of the queue of active bidders with a status of RU.
FreshBidRequest	The main process sends this signal to the agent selected for the next bid.
FreshBidSend	The agent, on getting the *FreshBidRequest* signal, determines the maximum utility bid and sends it to the main process together with this signal.
BidderInformed	When an agent cannot find a package of non-negative utility, it informs the bidder and also sends this signal to the main process, which puts the agent at the end of the queue with a status of RN.
GlobalTimeout	This signal is triggered by a timer when the auction duration expires.
Timeout	When the *GlobalTimeout* signal is received, the main process sends this signal to all active agents.

4.1. Algorithm for the Main Process

We give the algorithm separately for each state of the main process.

```
State MInit  /* initialize the main process */
    Start timer for GlobalTimeout;
    Initialize all the lists and variables including the DLs and WLs of all packages;
    Spawn the login process;
    /* The login process runs continuously to handle bidder login requests. Only pre-registered
        bidders can log in. Once a bidder logs in, the login process supplies a private copy of the
        proxy agent to the bidder. */
    Set state of main process to MWaitSelect;
    Transfer control to MWaitSelect;

State MWaitSelect /* wait for proxy agent signal */
    While state is MWaitSelect do
     {  While exists signal AddNewBidder
            and GlobalTimeout is not set
            {       Set the status of the main process to MUpdate; Transfer control to MUpdate; }

            If there is no agent with status RU
                If GlobalTimeout  is set {
                    Change the state of the main process to MTerminate; Transfer control to
                    MTerminate; }

    /* Select from active agents in queue having status RU or RN on FIFO basis */
            If GlobalTimeout  is set
                Take next agent b from queue with status RU;
            Else
                Take next agent b from  queue with status RU or RN;
           Send signal to proxy agent b for fresh bid;
           Set status of b to EB; /*expecting bid*/
```

```
            Change state of main process to MRequest; Transfer control to MRequest;
    }
```

State MUpdate /* update the list of active users */

```
    Add the proxy agent to the end of the queue of active agents;
    Change status of proxy agent to RU;
    Change state of main process to MWaitSelect; Transfer control to MWaitSelect;
```

State MRequest /* waiting for signal from proxy agent */

```
    While state is MRequest do
    {   Check whether there exists any request from a proxy agent to be processed;
            If such a signal exists
            {Case signal of
              FreshBidSend:
                Set status of proxy agent to RU;
                Move proxy agent to end of queue;
                Set state of main process to
              MCompute; Transfer control to MCompute;

            BidderInformed:
              Set status of proxy agent to RN;
              Move this agent to end of queue;
              Set state of main process to MWaitSelect; Transfer control to MWaitSelect;
        }
    }
```

State MCompute /* determine WLs and DLs */

```
    Compute DLs and WLs of all packages;
    Determine winning proxy agents;
    Set status of winning agents to RH;
    Set status of agents who were RH before and are no longer winning now to RU; /* relative
    positions in queue do not change */
    Set state of main process to MWaitSelect; Transfer control to MWaitSelect;

State MTerminate /* terminate the auction */
    Declare winning allocation;
    Declare payments to be made by winners;
    /* we assume payments are made on a first
            price basis  */
    Send Terminate signal to all proxy agents;
```

0.2. Algorithm for Proxy Agent

As already explained, there is a proxy agent corresponding to each human bidder; the logic of operation is identical for all proxy agents. We give the algorithm separately for each state of the agent.

```
State PInit /* initialization */
    Initialize the process;
    Request the bidder to submit a list of the
        packages of interest;
    Initiate WaitforInput process;
    /*  WaitForInput process runs continuously accepting bidder valuations and displaying WLs
        and DLs of packages when requested. On receiving input, it sets the flag NewValue to
        TRUE (this is checked in the state PSend), and again waits for bidder input. Thus this
        process helps the bidder to update the valuations of packages at any time.  */
    Send AddNewBidder signal to main process;
    Set state of this process to PWait; Transfer control to PWait;

State PWait /* waiting for fresh bid request from the main process */
    While state is PWait do
    {       If there exists a Terminate signal from the main process
            {       Set the state of the proxy agent to PTerminate; Transfer control to PTerminate;
            }
            If there exists a *FreshBidRequest* signal from main process
            {       Set state of proxy agent to PSend; Transfer control to PSend;
             }
    }
State PSend /* send the highest utility bid to the main process */
    If NewValue is TRUE /* NewValue is set by the WaitforInput process on bidder input */
    {
        For each package of interest to bidder
         /* Let the DL of the package be DL; Let V
          be the valuation supplied by the bidder; */
           Compute Utility $U = V – DL + \varepsilon$; /* $\varepsilon$ is bid increment */

    Determine the package that has the maximum utility $U_{max}$;
    If  $U_{max}$ ` > 0
    {       Send the bid value ($DL + \varepsilon$) and the package identification to the main process;
            Send signal FreshBidSend to the main process; /* NewValue remains TRUE */
    }
    Else
    {       Request bidder to revise package valuations; NewValue = FALSE;
            Send BidderInformed signal to main process;
    }
    }
    Else /* NewValue is FALSE */
    Send BidderInformed signal to main process;
    Set the state of this process to PWait; Transfer the control to PWait;

  State PTerminate /* Cleanup and terminate *
    Cleanup temporary files and variables;
    Kill the corresponding WaitforInput process;
    Kill proxy process;
```

5. Worked Out Example

We consider an online combinatorial auction of three items, A, B and C. Let there be three bidders, Bidder1, Bidder2 and Bidder3, who log in and are provided private proxy agents. The bidders then input the list of packages and their valuations to their respective agents: Bidder1: (A, 20), (AB, 30), (ABC, 60); Bidder2: (B, 25), (BC, 35), (ABC, 60); Bidder3: (C, 20); (AC, 40), (ABC, 25). Table 4 shows, in time sequence, the proxy agent selected by PRACA, the package on which the bid is submitted, DLs of all the agents, and the provisional allocation after each bid. We assume bidders are selected from the queue in a FIFO manner. At the end of round 3, Bidder4 logs in, gets a private proxy agent, and inputs: (B, 60), (ABC, 70). Bidder4 is inserted at the end of the queue of unhappy bidders. At instant 10, Bidder1 is selected for the next bid, but the DLs are such that all packages have negative utility. So the agent requests the bidder to revise the valuations. The main process then selects Bidder2. At instant 12, Bidder4 is asked to revise her valuations. Suppose Bidder1 and Bidder4 do not supply new valuations. At instant 13, Bidder3 is asked to revise her valuations and does so: (C, 40), (AC, 40), (ABC, 65). The proxy agent bids for package C which has the highest utility. This changes the seller's revenue to 100 and the provisional allocation to [4,AB], [3,C]. At this point, there is no unhappy bidder with status RU. Suppose the *GlobalTimeout* signal is received at this time. Then the auction terminates and the final winning allocation is: [3,C], [4,AB]. Bidder3 pays 40 for item C and Bidder4 pays 60 for package AB.

6. Theoretical Results

Let us assume that the bid increment ε is greater than 0 and finite, and that all valuations are also finite. It is then clear that the use of proxy agents will in general reduce the participation and monitoring costs of bidders significantly, since the bidders will need to interact with the auction system only at infrequent intervals. Two other interesting theoretical results are given below.

Result 1: PRACA always terminates.

Proof: After the *GlobalTimeout* signal is received, all RU agents in the queue still get processed. Some RH agents might become RU as a result, and they are also processed. Bidders who have reached their final valuations or have delayed submitting updated valuations will have RN agents, and they will not be able to bid any more. Since $\varepsilon > 0$, either an agent will be happy because it is in the final winning allocation, or it will no longer get an opportunity to bid. Thus an agent will finally attain a status of either RH or RN, so PRACA will terminate. □

Table 4: Worked-Out Example

Bid	Bidder Selected	Highest Utility Package	Deadness Level of Packages							rovisional Allocation
			A	B	C	AB	AC	BC	ABC	
At the start of the auction, there are three bidders present. The proxy agents for these bidders are in the active list with the status set to RU. The main process selects them in ‘first come first serve’ order which we assume to be Bidder1, Bidder2, Bidder3.										
1	Bidder1	ABC	0	0	0	0	0	0	20	[1,ABC]
2	Bidder2	ABC	0	0	0	0	0	0	40	[2,ABC]
3	Biddr3	AC	0	0	0	0	20	0	40	[2,ABC]
A new bidder Bidder4 joins the auction; its proxy agent with state RU enters the active agents list at the end after Bidder3. At this point, Bidder2 is happy with status set to RH and is not selected in the next iteration.										
4	Bidder1	AB	0	0	0	20	20	0	40	[2,ABC]
5	Bidder3	ABC	0	0	0	20	20	0	60	[3,ABC]
6	Bidder4	AB	0	0	0	40	20	0	60	[3,ABC]
7	Bidder1	A	20	0	0	40	20	0	60	[3,ABC]
8	Bidder2	BC	20	0	0	40	20	20	60	[3,ABC]
9	Bidder4	AB	0	0	0	60	20	20	60	[3,ABC]
Bidder1 gets selected; however there is no positive utility package remaining for the bidder. Its proxy requests for new valuations and send the *BidderInformed* signal to main. The status of the agent is updated to RN by main in the active list. Bidder3 is happy at this point.										
10	Bidder2	B	0	20	0	60	40	20	60	[3,ABC]
Bidder4 gets selected but it has no positive utility package. She is requested to submit revised valuations.										
Bidder1 again gets selected but no *NewValue* is available with the proxy. Its status remains as RN in the active list										
11	Bidder2	ABC	0	20	0	60	40	20	80	[2,ABC]
Bidder4 again gets selected but no *NewValue* is available with the proxy. Its status remains as RN in the active list										
Bidder1 again gets selected but no *NewValue* is available with the proxy. Its status remains as RN in the active list										
12	Bidder3	C	0	20	20	60	40	20	80	[2,ABC]
Bidder3 gets selected again because it is the only unhappy agent in the active list but it has no positive utility package. She is requested to submit revised valuations.										
13	Bidder3	C	0	20	40	60	40	20	100	[4,AB],[3,C]

Definition 1: At any instant during the auction, the RN agents in the queue are of two types, RN_W and RN_N. An RN_W agent corresponds to a bidder who has not yet reached her final valuation on some package of interest to her; such an agent will subsequently receive revised valuations from the bidder. An RN_N agent has a bidder who has reached her final valuations on all packages of interest to her; such a bidder will no longer revise her valuations on any package.

Definition 2: We will say that the duration of an auction is *sufficiently long* if, at all time instants from the arrival of the *GlobalTimeout* signal until the actual end of the auction, all RN agents in the queue are of type RN_N.

Result 2: For a given set of packages, bidders, and bidder valuations, the seller's revenue is identical in all auctions that have sufficiently long duration; this revenue is independent of the order of selection of unhappy (*i.e.*, RU and RN) bidders from the queue.

Proof: Consider an auction that has a duration that is sufficiently long. Then from the time instant the *GlobalTimeout* signal arrives, the queue does not contain any RN agents that are of type RN_W. So the RN agents all have bidders who no longer play any role in the bidding. Any RU agent who becomes RN also ceases to play any role. At the actual end of the auction, all agents in the queue have status RH or RN_N, and the RH agents all have packages in the winning allocation. If we solve the WDP with the final valuations of all the bidders, we will get this same winning allocation. So the final allocation obtained from an auction that has sufficiently long duration is identical to the one obtained by solving the WDP, and it is this allocation that determines the seller's revenue. We assume this revenue is calculated on a first price basis. This argument does not depend on the manner in which unhappy (*i.e.*, RU and RN) agents are selected from the queue during the auction. □

7. Experimental Verification

To verify whether PRACA works correctly on practical data, a C++ program was written to simulate the algorithm. No human bidders were involved; even the revision of valuations was automated. Multiple data sets were generated using the Combinatorial Auction Test Suite (CATS 2.0, [12]). Figure 3 shows two specimen plots obtained using an input data set generated from the 'regions' problem domain. This particular data set consisted of 15 items, 200 bidders and a total of 12,000 bids. We chose a bid increment of 10,000. Since there were a total of 200 bidders, the 12,000 bids were distributed among them with each bidder being allotted 60 bids. The bids were preprocessed to ensure no bidder had more than one bid on the same package. When the same package appeared more than once, we eliminated the bid with the lower bid value from the data set. The CATS bid values lay between 230 and 987,296. In effect this meant that all bids below the bid increment were ignored. We started the simulation with 15 items, 100 bidders and 6,000 bids, with 60 bids per bidder, and slowly introduced new bidders at periodic intervals until we reached 200 bidders. The initial valuations in the different runs were percentages (0%, 20%, 40%, 60% and 80%) of the final valuations, which corresponded with the values obtained from CATS. When the current valuation was reached, it was updated to the sum of the bid increment and the WL of the package at that instant. Figure 3 shows the revenue generated by PRACA as a function of the ratio of the initial valuation to the final valuation. The plot shows that as expected the revenue generated from PRACA remains the same irrespective of the initial valuations. We experimented with an initial valuation of 0 also, to check how PRACA performs without proxy agents. The plot shows that the revenue generated remains the same. The slight variation observed has been caused by the relatively large value of ε. For smaller values of ε the variation is less. Figure 4 shows the change in the number of valuation revisions as a function of the ratio of the initial valuation to the final valuation. As expected, the number of revisions reduces as the initial valuation is increased.

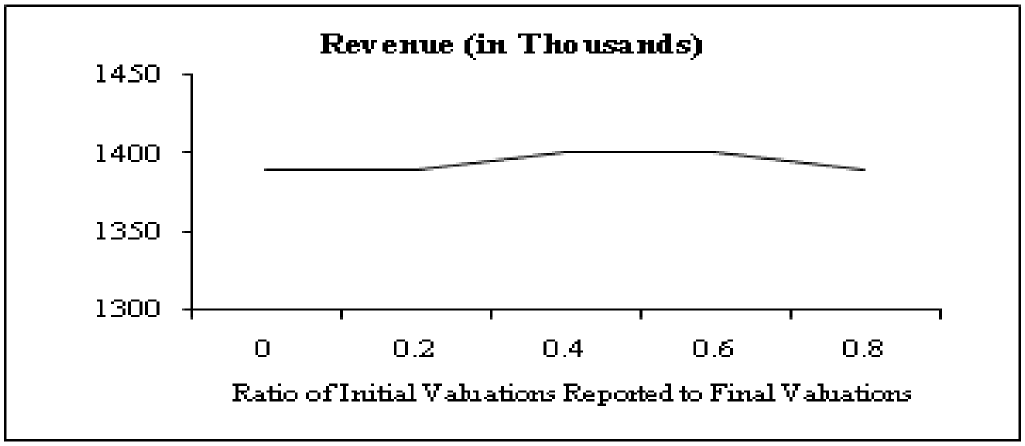

Figure 3: Seller's Revenue

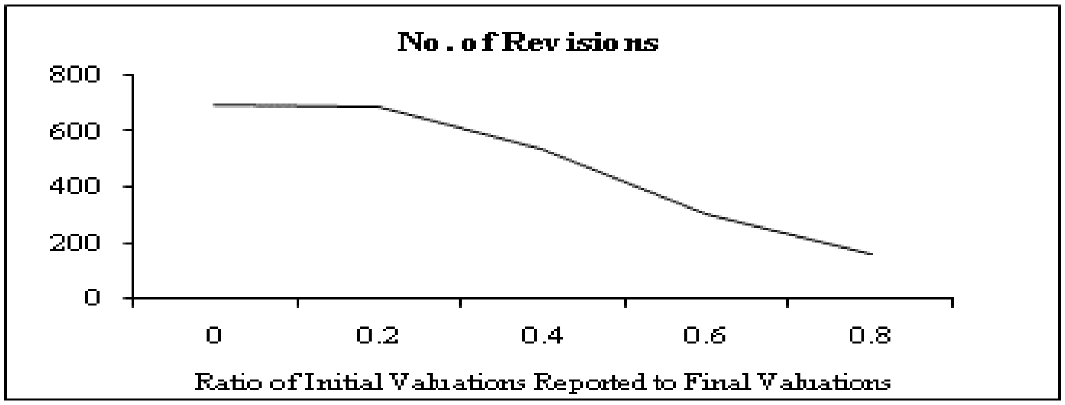

Figure 4: Number of Revisions

8. Concluding Discussion

In this paper we have proposed a scheme called PRACA for reducing the participation and monitoring costs of bidders in online combinatorial auctions. PRACA supplies a private autonomous proxy agent to a bidder, which then bids on her behalf. The method has been simulated experimentally and found to work correctly. PRACA does not require bidders to divulge their valuations to the seller or to other bidders. Valuations are fed to a private proxy agent, which tries to gradually increase the bid values to a level that makes the bidder happy, *i.e.*, a winner. This is an important advantage in situations of long term strategic involvement, where bidders are reluctant to make their true valuations publicly known. The scheme also provides information feedback to the bidders on the current state of the auction. We plan to execute a real time auction with human participants and check how the scheme performs. The use of dynamic programming to solve the WDP has the limitation that too much memory is needed when the number of items is large. We plan to address this difficulty in our future work. We also plan to examine the manner in which the sum of the utilities of the winning bidders will change as the selection procedure for unhappy proxy agents from the queue of ready bidders is varied.

9. References

1. Adomavicius, G., and Gupta, A. "Toward Comprehensive Real-Time Bidder Support in Iterative Combinatorial Auctions," *Information Systems Research* (16), June 05, pp 169 - 185.
2. Ausubel, L.M., Cramton, P., and Milgrom, P. "The Clock Proxy Auction: A Practical Combinatorial Auction Design," In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. The MIT Press, Cambridge, Massachusetts, USA 2005a, pp 115 - 138.

3. Ausubel, L.M., and Milgrom, P. "Ascending Proxy Auction," In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. The MIT Press, Cambridge, Massachusetts, USA 2005b, pp 79 - 98.
4. Bajari, P., and Hortacsu, A. "Economic Insights from Internet Auctions," *Journal of Economic Literature* (XLII: June, 2004) 2004, pp 457 - 486.
5. Banks, J., Ledyard, J.O., and Porter, D. "Allocating Uncertain and Unresponsive Resources: An Experimental Approach," *RAND Journal of Economics* (20) 1989, pp 1 - 25.
6. Clarke, E.H. "Multipart pricing of public goods," *Public Choice* (11) 1971, pp 17 - 33.
7. Goeree, J.O., Holt, A.C., and Ledyard, J.O. "An Experimental Comparison of the FCC's Combinatorial and Non-Combinatorial Simultaneous Multiple Round Auctions", *Report for Wireless Telecommunications Bureau of the Federal Communications Commission)*, 2006.
8. Groves, T. "Incentives in Teams," *Econometrica* (41) 1973, pp 617 - 631.
9. Kelly, F., and Steinberg, R. "A Combinatorial Auction with Multiple Winners for Universal Service," *Management Science* (46) 2000, pp 586 - 596.
10. Klemperer, P. "*Auctions Theory and Practice*", Princeton University Press 2004.
11. Kwasnica, A.M., Ledyard, J.O., Porter, D., and Demartini, C. "A New and Improved Design for Multiobject Iterative Auctions," *Management Science* (51:3) 2005, pp 419 - 434.
12. Leyton-Brown, K., Pearson, M., and Shoham, Y. "Towards a Universal Test Suite for Combinatorial Auction Algorithms," *In Proceedings of the ACM Conference on Electronic Commerce*: October 2000, pp 66 -76.
13. Parkes, D.C. "iBundle: An Efficient Ascending Price Bundle Auction," *Proceedings of the 1st ACM Conference on Electronic commerce* 1999.
14. Parkes, D.C. "Iterative Combinatorial Auctions," In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*. The MIT Press, Cambridge, Massachusetts, 2006, pp 41 - 77.
15. Rassenti, S.J., Smith, V.L., and R.L., B. "A combinatorial auction mechanism for airport time slot allocation," *Bell Journal of Economics* (13) 1982, pp 402 - 417.
16. Sandholm, T. "Algorithm for Optimal Winner Determination in Combinatorial Auctions," *Artificial Intelligence* (135) 2002, pp 1 - 54.
17. Varian, H., and MacKie Mason, J.K. "Generalized Vickrey auctions," *Tech Report, University of Michigan* 1995.

18. Vickrey, W. "Counterspeculation, Auctions, and Competitive Sealed Tenders," *Journal of Finance* (16) 1961, pp 8 - 37.

19. Walsh, W.E., Wellman, M.P., and Ygge, F. "Combinatorial auctions for supply chain formation," *In Proceedings of ACM Conference on Electronic Commerce* (EC-00) 2000, pp 260 - 269.