30

# Efficient Winner Determination Techniques for Internet Multi-Unit Auctions

Achal Bassamboo[1] Manish Gupta[2] and Sandeep Juneja[3]
*Stanford University[1]: IBM India Research Lab., Delhi[2]: Indian Institute of Technology, Delhi[3]*

**Abstract**: In this paper we develop algorithms for efficiently processing bids in a single item, $N$ unit ($N \geq 1$) open cry auction for a large class of winner determination rules. Existing techniques consider all previously submitted bids along with the new arrivals to update the current set of winners. We propose that at most $N$ "potential winner bids" amongst the previously submitted bids need to be used for such updates, thus significantly reducing the computation time and memory requirements. This is crucial when a large number of auctions are being conducted simultaneously. For a commonly used greedy auction rule we show that the expected number of potential winner bids may be much less than $N$ under reasonable probabilistic assumptions.

## 1. INTRODUCTION

The phenomenon of a website attracting millions of users in a short duration is becoming increasingly frequent. On July 14, 1998, the Guinness Book of World Records recognized the 1998 Olympic Games Web Site for two world records (see Iyenger et. al. 1998). The first was for receiving 634.7 million requests over the 16 days of the Olympic games. The second was set when 110,414 hits were received in a single minute around the time of the Women's Figure Skating. We have come a long way since then. The use of Internet to facilitate commerce, and in particular auctions, has also been growing at a rapid rate. It is not unreasonable to expect that an auction sale of a superstar's memorabilia in a sports event may attract thousands of bidders. Such a large number of participants in an auction can put immense load at a web server.

We refer the reader to McAfee and McMillan (1987) and Milgrom (1989) for a comprehensive survey on auctions theory. Varian (1995) provides an overview of the design of economic mechanisms for auctions. Wurman et. al. (2000) present an extensive breakdown of the auction space that captures the essential similarities and differences of many auction mechanisms in a format more descriptive and useful than simple taxonomies. Sandholm (1999) presents a scalable search algorithm for optimal winner determination in a combinatorial auction. Internet commerce technologies for different kinds of auctions: open-cry, sealed bid, and Dutch auction (see Kumar and Feldman 1998a, 1998b) have been developed for Websphere Commerce Suite (WCS 4.1), IBM's commerce server. While Kumar and Feldman (1998a), (1998b) describe how these auctions may be conducted on the Internet, they do not discuss issues related to improving performance and scalability of these auctions.

In this paper our focus is on quickly determining winner bids in an open-cry auction for $N$ units ($N \geq 1$) of a single item. For such auctions, an obvious allocation strategy is to assign the units to the bidders in a descending order of the price/unit offered by them. However, this may not be feasible all the time. For example, a bid desiring multiple units that wants "all or nothing" may be rejected if the quantity demanded is not available. Also, the same bid may be accepted at a later time (and a previously accepted bid rejected) if a newly arriving bid makes it attractive and feasible to include it along with the rejected bid in the list of "current winners". Due to such complications, the existing packages such as WCS 4.1, at any time during an ongoing auction, combine the newly arrived bids with all the ones that had arrived earlier to determine the current set of winners based on the auction rules. In such cases, if the number of bids is large, the processing delays may be large and the server may even breakdown. Our main contribution is that we show that under a large class of auction rules (including those that are practically implemented) significant computational improvement over naive methods is possible in determining current set of winners. In particular, we introduce the notion of maintaining a small subset of well chosen "potential winner bids" amongst all the bids that have arrived at any time during an ongoing auction. The remaining bids may be rejected as "loser bids", i.e., they have no chance to qualify as winner bids in the future. We develop two such potential winner sets. The first one is simple to identify and update and is referred to as the "Coarse Filter Structure" (**CFS**). We show that the number of bids in CFS at any time is $O(N \log N)$[45]. The second one is more intricate and is referred to as the "Refined Filter Structure" (**RFS**). We show that RFS may contain at most $N$ bids at any time. These structures considerably reduce the response times to the bidders and the computer storage requirements since this reduced list of potential winner bids can be maintained as a set of persistent objects in the main memory for quick access and modification. An

[45] A function f(N) is said to be O(g(N)) if there exists a constant K>0 so that $f(N) \leq K\ g(N)$ for all N sufficiently large. It is said to be $\Theta(g(N))$ if there exist constants $0<K_1<K_2$ so that $K_1\ g(N) < f(N) \leq K_2\ g(N)$ for all N sufficiently large.

additional benefit of this is that the loser bids get an opportunity to quickly revise their bids.

We focus on two commonly used rules for selecting winners: namely, the greedy rule and the knapsack rule (explained in Section 2). The greedy rule is the most popular rule due to the simplicity both in explaining it to a lay person and in its implementation. We develop algorithms that quickly update RFS ($O(\log N)$ per bid) when the greedy rules are used. We also show that under greedy rules and under some reasonable assumptions on the probability distribution of the bid size, if the average bid size is $\Theta(N^{\alpha})$ ($0 < \alpha < 1$), then the expected number of bids in RFS is $\Theta(N^{1-\alpha})$.

There exists some literature related to the online knapsack problem (see Leuker 1998, Marchetti-Spaccamela 1995). Their focus is somewhat different from ours as they consider auctions where the final decision on whether a bid is a winner or a loser is made as soon as an arriving bid is analyzed. This benefit of speedier decision-making is at a cost that the solutions obtained may be sub-optimal. We, on the other hand, decide whether a bid is a potential winner bid or a loser bid at the time of its arrival. Our solution is always optimal.

In this paper, we theoretically demonstrate that the proposed algorithms offer significant improvement over the current naive implementations. To keep the exposition short, simulation experiments are not reported. The interested reader is referred to Bassamboo et. al. (2000) to view the orders of magnitude of the computational gains confirmed by the simulation experiments.

In Section 2 the commonly used winner determination auction rules for open cry auctions are reviewed. In particular, we review the existing implementation of WCS 4.1. In Section 3, assumptions on the auction rules considered are stated. In particular, CFS is identified and a theorem identifying the RFS is stated and proved. The updating techniques for RFS under the greedy rule are discussed in Section 4. This section also states a theorem describing the order of magnitude of the expected number of bids in RFS for large values of $N$. Finally in Section 5, we briefly discuss issues such as the bidder utility functions supported by the bid structure that we consider and the problems related to processing *order bids,* (i.e., a software agent that bids on behalf of a human bidder).

## 2. OPEN-CRY AUCTION: AN OVERVIEW

In an open-cry auction (also popularly known as "English" auction) of a single item with multiple units, the seller may specify the minimum starting bid. She may also specify a minimum bid increment that a new bid needs to have over the current best price (based on bids of current winners) to be eligible[46]. All currently eligible bids are displayed to the users. The set of current winning bids (based on pre-specified auction rules) is also identified. At the time of bidding, the bidder specifies the number of units desired. A bidder may also specify whether she would accept *partial* quantity or not. Once the bidding phase is over, the bidders with the high bids get the items being auctioned, but the price they pay could be different from what they bid. In a discriminative auction, also known as a Yankee auction, the winners pay the amount that they bid. In a non-discriminative auction the winning bidders pay the price paid by the winning bidder with the lowest bid. This is currently the trend on the Internet; sites like www.ebay.com use this methodology for auctioning off multiple items.

## 2.1 Greedy Rules for Winner Determination

Some notation is needed before we describe the greedy rules for winner determination. Recall that $N$ denotes the number of units on a single item multi-unit open-cry auction. For any bid $b$ let val($b$) denote the price per unit offered. Let Q($b$) denote the quantity requested in the bid and let T($b$) denote the time of arrival of the bid. Note that if a bid $b$ is willing to accept partial assignment of the quantity bid, then from winner determination viewpoint we may treat $b$ as comprising Q($b$) identical bids, each bidding for a single unit. This allows us to assume, without loss of generality that all bids in the auction do not accept partial assignments, i.e., **they want all or nothing.** Let R denote a set of rules of an open cry auction. For example, we set R= "greedy", when the greedy method of winner determination (see Section 2.2) is used and set it to "knapsack" when the knapsack rules (see Section 2.3) are used. Let W(R, Z, $N$) denote the winner bids amongst bids Z in an auction for $N$ units of a single item, where winners are selected using rules R.

Given a set of bids Z, *an example* of greedy rules for winner determination would typically sort all the bids so that given any two bids $b_i$ and $b_j$, the sorting places $b_i$ above $b_j$ iff:

- $\text{val}(b_i) > \text{val}(b_j)$ or,
- $\text{val}(b_i) = \text{val}(b_j)$ and $Q(b_i) > Q(b_j)$, or,
- $\text{val}(b_i) = \text{val}(b_j)$, $Q(b_i) = Q(b_j)$ and $T(b_i) < T(b_j)$ .

We use the convention $b_i > b_j$ (resp., $b_i < b_j$) to mean that bid $b_i$ is placed above (resp., below) bid $b_j$ in the sorted list. In case $b_i = b_j$ one may break the tie arbitrarily by assuming small perturbations in the time of bid arrival. We refer to this method of sorting as *gsort.*

[46] A bid is considered eligible if at the time of submission it passes all the checks such as check on bidder's creditworthiness, minimum bid criteria, etc.

## 2.2 Winner determination algorithm

In this section we outline the algorithm, **A1,** that considers a set of bids Z sorted according to gsort and determines W(greedy, Z, *N*). The algorithm **A1** proceeds in a straightforward recursive manner. At any step it tries to satisfy the demand of the top-most bid in the sorted list with the available quantity (initially, this equals *N*). If the demand cannot be satisfied, the corresponding bid is rejected. Otherwise the demand is met and the available quantity is reduced by that demand, and the next bid in the sorted list is considered. The algorithm terminates when either all the bids have been considered or when the available quantity reduces to zero, whichever occurs first. Some notation is needed to facilitate the listing of the pseudocode for **A1.** Let $(b_1, b_2, ..., b_n)$ denote the list of bids in Z sorted in descending order using gsort. Let $X_i$ denote the available quantity when the bid $b_i$ is considered for allocation. The pseudocode is as follows:

```
Ψ = ∅, X1 = N, j = 1;
while (Z ≠ ∅ and Xj ≠ 0) {
   if    (Q(bj) > Xj) Xj-1= Xj;
   else  Ψ = Ψ ∪ { bj }, Xj+1= Xj - Q(bj);          ...............(1)
   Z = Z - {bj}, j = j + 1;
}
```

The set Ψ returned from this algorithm denotes the set W(greedy, Z, *N*). Later we shall see that the RFS under greedy rules can be obtained by a slight modification of the above algorithm.

In practice, some variants of greedy rules may sort in a slightly different manner from gsort (e.g., if the two bids quote the same value per unit, then one may be selected over the other at random). To keep things simple we focus on the greedy rule that sorts using gsort and selects winners as described by **A1.** As will become apparent to the reader, *the key ideas of this paper can be adjusted to suit variants of the greedy rule through simple modifications.*

## 2.3 Knapsack rules for Winner Determination

A seller's revenue is maximized when winners are computed by solving an integer knapsack problem (see, e.g., Horowitz and Sahni 1990). To see this precisely, consider again the set $Z = (b_1, b_2, \ldots, b_n)$. Let $(\boldsymbol{I}_i: i=1, \ldots, n)$ be a solution to the integer program:

maximize $\sum_{i=1}^{n} I_i * val(b_i) * Q(b_i)$ such that

$\sum_{i=1}^{n} I_i * Q(b_i) \leq N$, and ($I_i$ : $i$ = 1, ..., n) equals 1 or 0. Then W(knapsack, Z, $N$) consists of each bid $b_i$ for which $I_i$ =1. Since the winner bids obtained using the greedy rule form a feasible solution to this problem, the seller's revenue under greedy rule is less than or equal to his revenue under knapsack rule. However, it is easily seen that if all the bids either request a single item or are partial bids, i.e., bidders are ready to accept any quantity less than what is requested in their bids, then the greedy solution is equivalent to the knapsack solution.

## 2.4 Implementation of Winner Determination in WCS 4.1

Assume that the auction starts at time t=0 and continues till time $t=T_{end}$. Foar $0 \leq t_1 \leq t_2 \leq T_{end}$, let $S(t_1, t_2)$ denote the set of bids that arrive after time $t_1$ and before or at time $t_2$. Let S(t) denote S(0,t). The winner determination algorithm in WCS 4.1 works as follows: All arriving bids are stored in a database. A process that computes winner bids runs periodically every $\tau$ seconds (currently $\tau$ is kept at 90 seconds). During run $k \geq 1$ of this process (at time $k\tau$), all the bids in the database are considered, i.e., the set of bids $S((k-1)\tau)$ that were present at the time of the last run, as well as the set of bids $S((k-1)\tau, k\tau)$ that arrived after the last run. The greedy rules (i.e., algorithm **A1**) are used to determine the new set of winners from the resulting set $S(k\tau)=S((k-1)\tau) \cup S((k-1)\tau, k\tau)$.

To get an approximate idea of the computational effort involved, let m denote the cardinality of the set $S((k-1)\tau, k\tau)$ and l denote the cardinality of the set $S((k-1)\tau)$. Clearly, in the above algorithm, the computational effort for sorting the newly arrived m bids along with the existing (already sorted) l bids is O(m log(m+l)) (see, e.g., Cormen et. al. 1990). In addition, in the worst case **A1** may require considering each bid from the sorted list to determine the current set of winners (for large m, this will happen if most bids do not get the quantity they desire). Therefore, the additional computational effort in finding the current winners is in the worst case O(m+l). During peak load, m may be quite large (much larger than $N$) and thus large amount of computational effort may be required. In the next section we present algorithms that drastically reduce this effort.

# 3. IDENTIFYING THE SET OF POTENTIAL WINNERS

We first motivate the need to identify a set of potential winners through an example. Denote each bid by a pair of numbers; the first denotes the price/unit and the second denotes the quantity required (assume that all bids ask for all or nothing). Assume that the greedy rules are used to determine the winners. Now suppose that $N = 5$ and the following 6 bids have arrived in an ongoing auction: (25, 5), (23, 3), (20, 4), (18, 4), (17, 2) and (10, 1). Clearly the first bid is the only one in the current winners set. We address the problem of determining which amongst the loser bids may in future become a winner bid and which amongst them can never in future become a winner bid and hence need not be considered in future winner determination. Note that if the only bid that arrives next is (30, 4) then along with it bid (10, 1) becomes a winner. Similarly, if (30, 3) is the only other bid to arrive, then along with it bid (17, 2) becomes a winner. Similarly, arrival of an appropriate bid could make (23, 3) a winner bid. It is easy to see that the bids (20, 4) and (18, 4) can never be winners, since the bid (23, 3) always supersedes both of them. Thus, the potential winner set includes (25, 5), (23, 3), (17,2) and (10, 1).

In this section we make a set of assumptions that impose minimal restrictions and cover a large class of practically implemented rules for open-cry auctions. We also identify CFS and RFS and show that under these assumptions, at any time t, CFS (resp., RFS) retains at most $O(N \log N)$ (resp., $N$) bids from the set S(t) to determine winners of the auction at any time in future. Some notation is needed for this purpose.

Let $S^*(t)$ denote the bids $S(t, T_{end})$. Let $S_q(t) \subseteq S(t)$ denote the set of all bids in S(t) asking for quantity q for $q \leq N$. For notational simplicity we suppress the reference to t and let S, $S^*$, $S_q$ denote S(t), $S^*(t)$, $S_q(t)$ whenever this does not cause confusion. It is worth keeping in mind that the results shown below involving S and $S^*$ hold for *any* two disjoint sets S and $S^*$.

**Assumption 1** *Under the set of rules R, there exists a sorting criteria **C** for the bids in $S_q(q \leq N)$ such that any bid in the list sorted using **C** belongs to the winner set W(R, S, N) only if all the bids above it in the list belong to W(R, S, N).*

Note that this assumption is satisfied by the greedy and the knapsack rules when the sorting amongst bids asking for the same quantity is based on the bid value, e.g., under gsort (for example, (20, 4) is preferred over (18, 4) both under greedy as well as knapsack rules).

For any set Z, let $|Z|$ denote its cardinality. For any number x, let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. When Assumption 1 holds, and if $|S_q| \geq \lfloor N/q \rfloor$, then let $T_q(S_q)$ denote the set of top $\lfloor N/q \rfloor$ bids of the set $S_q$ under **C**. Otherwise, if $|S_q| < \lfloor N/q \rfloor$ then let $T_q(S_q) = S_q$. Let $T(S) = \bigcup_{q=1}^{N} T_q(S_q)$. The following proposition easily follows from Assumption 1:

**Proposition 1** *Under Assumption 1,* $W(R, S \cup S^*, N) = W(R, T(S) \cup S^*, N)$.

This result implies that in any ongoing auction, if Assumption 1 holds, then all potential winner bids from set S lie in the subset T(S). In particular, the bids in the set S - T(S) clearly are the loser bids and can be rejected. We refer to T(S) as the Coarse Filter Structure (**CFS**).

**Lemma 1** *The following relation holds:* $|T(S)| \leq N(1+\log N)$.

**Proof** : Note that $|T(S)| = \sum_{q=1}^{N} |T(S_q)| \leq \sum_{q=1}^{N} N/q = N(1+\sum_{q=2}^{N} 1/q)$. The result follows by noting that $1/q \leq \int_{q-1}^{q} 1/x\, dx$. □

Thus, under Assumption 1, O($N$ log $N$) bids need to be kept at any time and the rest can be discarded. We now show that the number of potential winners can be further reduced under the following two assumptions:

**Assumption 2** *Under the set of rules R, for all N, if* $b \in Z$ *and* $b \notin W(R, Z, N)$, *then* $W(R, Z, N) = W(R, Z - \{b\}, N)$.

Thus, under this assumption, the removal of any loser bid from the set of bids Z does not alter the set of winner bids.

**Assumption 3** *Under the set of rules R, for all N, if* $b \in W(R, Z, N)$ *then:* $W(R, Z - \{b\}, N - Q(b)) = W(R, Z, N) - \{b\}$

Thus, if one winner bid b is assigned Q(b) units of the item, the remaining winners (call them B) are not assigned any unit and the remaining $N$-Q(b) units are re-auctioned to the Z - {b} bids, then the winners of the re-auction are the unassigned winners of the previous auction, i.e., the set B. Many practically conceivable rules for the open-cry auction satisfy Assumptions 2 and 3. In particular, it is easily seen that the greedy and the knapsack rules do satisfy these assumptions.

For notational brevity let G(R, S, N) denote the set $\bigcup_{q=1}^{N} W(R, S, q)$, let $Z_1$ denote the set $W(R, S \cup S^*, N) \cap S$ and let $Z_2$ denote the set $W(R, S \cup S^*, N) \cap S^*$. Thus $Z_1$ denotes the final auction winners that have already arrived (i.e., belong to the set S) and $Z_2$ denotes the final auction winners that are yet to arrive (i.e., belong to the set $S^*$). Note that if the bids in $Z_2$ require q units then the bids in $Z_1$ require $N$ - q units (assuming that all $N$ units are allocated). Intuitively one then expects that $Z_1 = W(R, S, N - q)$, since this is the best way to allocate $N$ - q units to bids in S. Since, q can take any value from 0 to $N$, it is reasonable to expect that $Z_1 \subseteq G(R, S, N)$. This is proved in the following theorem (its proof is given in the Appendix). We also show that the number of bids in G(R, S, $N$) (and hence $Z_1$) is upper bounded by $N$.

**Theorem 1** *If R satisfies Assumptions 2 and 3, then*

$$Z_1 \subseteq G(R, S, N). \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

*In addition, for all* $j \geq 1$,

$$|G(R, S, j)| \leq |G(R, S, j - 1)| + 1 \leq j. \quad \ldots\ldots\ldots\ldots \quad (3)$$

Thus, the above theorem states that if Assumptions 2 and 3 hold then at any time t, at most $N$ bids amongst S are potential winners. In particular, the bids in S - G(R, S, $N$) are the loser bids that can be rejected. We shall henceforth refer to G(R, S, $N$) as the Refined Filter Structure (**RFS**). Note that it is easy to construct an $S^*$ so that $Z_2$ requires any quantity from 0 to N. Therefore any set (W(R, S, q): $1 \leq q \leq N$) can be a subset of a winner set for an appropriate $S^*$. In particular, for any bid $b \in$ G(R, S, $N$), there exists an $S^*$ so that b is a final winner bid. It follows that RFS cannot be further trimmed. Suppose that, in an auction, we wish to compute the current set of winners at discrete time intervals ($t_1$, $t_2$, ..., $t_k$). Then the above theorem suggests that, at any time $t_i$, we retain only the bids in G(R, S($t_i$), $N$) and use them to compute W(R, S($t_i$), $N$).

## 3.1 Updating CFS

Note that CFS is easy to update as the new bids arrive. If the newly arrived bid requires q units then it may be inserted in $T_q(S_q)$ in O(log ($N$)) time (using binary sort techniques). Furthermore, if, after announcing the current set of winners, an auction receives a large number of bids (much larger than $N$), then there is a significant chance that an arrived bid will not change CFS. It may then be desirable to first compare the newly arrived bid requiring q units with the last bid in the set $T_q(S_q)$, and reject the lower of the two (only if $|T_q(S_q)| = \lfloor N/q \rfloor$).

# 4. UPDATING RFS UNDER GREEDY RULES

It is noteworthy that when knapsack winner determination rules are used, updating RFS may be computationally expensive and hence CFS may be preferred over RFS. We refer the reader to Bassamboo et. al. (2000) for a discussion on how RFS may be updated using standard dynamic programming techniques under knapsack rules. Fortunately, greedy rules have a nice recursive structure that may be exploited to efficiently compute RFS. Lemma 2 states the key recursive relationship that proves useful in efficiently determining RFS under greedy rules. To aid in its statement let r(x, y) denote max(x - y, y - 1), let n denote the cardinality of the set of bids Z and let ($b_1$, $b_2$, ..., $b_n$) denote the list of these bids sorted in descending order using gsort.

**Lemma 2** *For* $j \geq Q(b_1)$,

$$G(greedy, Z, j) = \{b_1\} \cup G(greedy, Z - \{b_1\}, r(j, Q(b_1))). \quad \ldots..(4)$$

*For* $j < Q(b_1)$,

$$G(greedy, Z, j) = G(greedy, Z - \{b_1\}, j). \quad \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Proof of Lemma 2 is given in the appendix. In view of this lemma, the algorithm for determining G(greedy, Z, *N*) is a simple modification of **A1.** This algorithm, call it **A2,** is identical to **A1** except that in (1), the step $X_{j+1} = X_j - Q(b_j)$ is replaced by $X_{j+1} = r(X_j, Q(b_j))$.

Note that the output Ψ is the set G(greedy, Z, *N*) and that the elements in Ψ may easily be maintained as a sorted list (based on gsort). Also, analogous to **A1,** for all j, after bids $\{b_1, b_2, \ldots, b_{j-1}\}$ have been examined, **A2** proceeds recursively to solve the smaller problem of determining G(greedy, $Z^j$, $X_j$), where $Z^j = \{b_j, \ldots, b_n\}$. The computational effort required in the above algorithm is O(n log n) for sorting Z, and O(n) for finding G(greedy, Z, *N*). For large set Z the effort required in sorting it can be high. We refer the reader to Bassamboo et. al. (2000) for updating procedures that do not require the sorted Z. In particular, it discusses RFS updating techniques that require O(log N) updating computational effort per bid.

## 4.1 Expected number of potential winner bids

Although, we have an upper bound of *N* on G(greedy, Z, *N*) and it is easy to create cases where this bound is achieved (e.g., when all the top bids request a single item), simulation experiments show that number of bids in G(greedy, Z, *N*) is typically much smaller (see Bassamboo et. al. 2000). We now conduct an order of magnitude analysis of the expected number of bids in G(greedy, Z, *N*) under some reasonable assumptions on the probability distribution of bid sizes. Our conclusion is that for large *N,* this expectation typically is much smaller than *N.* Let τ denote the number of bids in the set G(greedy, Z, *N*).

Assumption 4 explains the class of probability distributions on the sizes of bids that we consider. This assumption is somewhat restrictive to avoid undue mathematical technicalities. It however covers many cases of practical relevance (as discussed later) and provides insight explaining the order of magnitude of the expected number of bids in G(greedy, Z, *N*).

**Assumption 4** *There exist positive numbers* $\overline{Q}, \beta, \gamma$ *and* $0 \leq \alpha \leq 1$ *such that* $\overline{Q} = \Theta(N^{\alpha})$, *and* $\beta/N^{\alpha} \leq P(Q(b_i) = q) \leq \gamma/N^{\alpha}$ *for* $q \leq \overline{Q}$, *for all* $i \leq n$ *and* $P(Q(b_i) = q) = 0$ *otherwise for all* $i \leq n$.

For example, the case where each $Q(b_i)$ takes values from 1 to $N$ with equal probability is modeled by taking $\alpha = 1$, $\overline{Q} = N$ and $\beta = \gamma = 1$. To capture the setting where the bids are predominantly small in size, we may consider a smaller $\overline{Q}$, for example, $\overline{Q} = \Theta(\sqrt{N})$ or $\overline{Q} = \Theta(1)$. We now state the main theorem of this section. Its proof involves lengthy technicalities and is given in Bassamboo et. al. (2000).

**Theorem 2** *Under Assumption 4,* $E(\tau) = \Theta(N^{1-\alpha}) + O(\log N)$.

# 5. DISCUSSION OF SOME PRACTICAL ISSUES

## 5.1 Bidders with Decreasing Marginal Utility Functions[47]

Economic theory postulates that most bidders have decreasing marginal utility functions. It is noteworthy that the bids permitted in our discussion can support such bidders. Thus, suppose that the bidder is willing to bid $v_1$ for the first unit, $v_2$ for the second unit, . . . ,$v_k$ for k-th unit, where $v_1 > v_2 > \ldots > v_k > 0$. Then she may maximize her utility function by sending k bids each of size 1 and price $v_1, v_2, \ldots$, $v_k$, respectively. Clearly, a bid with price $v_i$ wins only if bids with price $v_j$ win for all $j < i$ (both under greedy and knapsack rules). However, when $v_i$'s are not monotonically non-increasing (e.g., first unit is worth \$0, second is worth \$5 and the third is worth \$4), further research is needed to determine efficient computation techniques.

## 5.2 Order Bids

At most of the Internet sites conducting open-cry auctions, the bidders are given the option of placing a regular bid (i.e., bid where a unique bid value is specified) or placing an *order bid* (also called *proxy bid*), i.e., a software agent that bids on behalf of a human bidder. An order bid differs from the regular bid in the sense that it specifies the bidder's *maximum bidding limit* instead of a unique bid value. The following simple assumption (satisfied by both greedy and knapsack rules) on the auction rules makes winner determination straightforward, even when order bids are involved:

**Assumption 5** *Under auction rules R, if b ∈ W(R, S, N) for a given val(b), then, keeping all else fixed, b ∈ W(R, S, N) even if val(b) is increased.*

47 We thank Terence Kelly (Univ. of Michigan) for pointing this out and for related discussions.

Thus, to determine the winners in an auction that allows regular bids and order bids, each order bid may be treated as a regular bid with the maximum bidding limit treated as its bid value. Thus an order bid may only send a single bid. The issue of the amount the order bid needs to pay is a complex one, that will be addressed by the authors elsewhere.

# 6. CONCLUSION AND FURTHER RESEARCH AREAS

In this paper we introduced the concept that for a large class of winner determination rules, at any time in an ongoing auction, amongst all the arrived bids, it is possible to identify a small subset that contains all potential winner bids. We focussed on multiple N unit single item auction. For this auction we identified the minimal subset of potential winner bids and showed that it contains at most N bids. We further showed that on an average this number may be much less than N if the bids arriving require large quantities. We demonstrated that this approach may provide computational benefits as well as improved response times to customers.

The following are some issues arising from this work that need further attention: In our analysis we assumed that the bidder may not retract or modify her bid. Theory needs to be developed to address this. Research is also needed to see how these ideas extend to combinatorial auctions where more than one item is auctioned. In this paper we focussed on bids that demanded all or nothing. We discussed that if a bidder has a marginally decreasing utility function then her bids can be easily incorporated in our framework. Further research is needed to efficiently identify winner bids and potential winner bids when the bidder utility function is more general.

# 7. APPENDIX

**PROOF of Theorem 1:** Assumption 2 implies that $Z_1 \cup Z_2 = W(R, S \cup Z_2, N)$. Assumption 3 further implies that $Z_1 = W(R, S, N - \sum_{\beta \in Z2} Q(b))$ from which (2) follows. We prove (3) using induction on the number of identical items on auction. Note that $|G(R, S, 1)|$ equals 0 or 1, depending on whether $W(R, S, 1)$ is empty or not. Assume that (3) holds for all $j \leq m$. We now show that it holds for m+1 by assuming the contrary and showing a contradiction. Thus, suppose that there exist bids $b_1, b_2 \in G(R, S, m+1)$ and $b_1, b_2 \notin G(R, S, m)$. This implies that $b_1, b_2 \in W(R, S, m+1)$. From Assumption 3, we have that $W(R, S - \{b_1\}, m+1 - Q(b_1)) = W(R, S, m+1) - \{b_1\}$. Since $W(R, S - \{b_1\}, m+1 - Q(b_1)) \subseteq G(R, S, m)$, it follows that $b_2 \in G(R, S, m)$ giving us the desired contradiction. ◻

**PROOF of Lemma 2:** Note that (5) straightforward since $b_l$ is too big to belong to any W(R, S, i) for ($i \leq j$). Now we prove (4). Again for $i < Q(b_1)$, it is obvious that:

$$W(greedy, Z, i) = W(greedy, Z - \{b_1\}, i). \ldots\ldots\ldots\ldots (6)$$

From greedy rules it is clear that for ($i = Q(b_1), \ldots, j$), $b_1$ will belong to the set W(R S, i) and the remaining quantity $i - Q(b_1)$will be satisfied from amongst the bids $Z - \{b_1\}$, i.e.,

$$W(greedy, Z, i) = b_1 \cup W(greedy, Z - \{b_1\}, i - Q(b_1)). \ldots\ldots (7)$$

Hence, from (6) and (7), taking the union of W(greedy, Z, i) for $i \leq j$, it follows that

$$G(greedy, Z, j) = (\cup_{i=1}^{Q(b_1)-1} W(greedy, Z - \{b_1\}, i)) \cup \{b_1\} \cup (\cup_{k=1}^{j-Q(b_1)} W(greedy, z - \{b_1\}, k))$$

and (4) follows.

## 8. REFERENCES


BASSAMBOO, A., M. GUPTA AND S. JUNEJA. 2000. Efficient Winner Determination Techniques for Internet Single Item Multi-Unit Open-Cry Auctions. IBM Research Report RI 0027.

CORMEN, T. H., C. E. LEISERSON AND R. L. RIVEST. 1990. *Introduction to Algorithms.* The MIT Press.

HOROWITZ, E. AND S. SAHNI. 1990. *Fundamentals of Computer Algorithms.* Computer Science Press.

IYENGER, A., J. CHALLENGER, D. DIAS AND P. DANZIG. 1998. Techniques for Designing High-Performance Web Sites. IBM Research Report RC21324.

KUMAR, M. AND S. I. FELDMAN. 1998. *Internet Auctions.* http://www.ibm.com/iac/papers/auction_fp.pdf.

KUMAR, M. AND S. I. FELDMAN. 1998. Business Negotiations on the Internet. *Inet '98.* Geneva, Switzerland.

LEUKER, G. 1998. Average-Case Analysis of Off-line and On-line Knapsack Problems. *Mathematical Programming* **29** (2): 277-305.

MARCHETTI-SPACCAMELA, A. AND C. VERCELLIS. 1995. On-line Knapsack Problems. *Mathematical Programming* **68** (1): 73-104.

MCAFEE, R. P. AND J. MCMILLAN. 1987. Auctions and Bidding. *Journal of Economic Literature* **25**: 699-738.

MILGROM, P. 1989. Auctions and Bidding: a primer. *Journal of Economic Perspective* **3** (3):3-22.

SANDHOLM, T. 1999. An Algorithm for Optimal Winner Determination in Combinatorial Auctions. *International Joint Conference on Artificial Intelligence (IJCAI).* Stockholm, Sweden: 542-547.

VARIAN, H. R. 1995. Economic Mechanism Design for Computerized Agents. *Usenix Workshop on Electronic Commerce.* New York.

WURMAN, P., M. WELLMAN AND W. WALSH. 2000. A Parametrization of the Auction Design Space, *Games and Economic Behavior*.