

# Market-Based Allocation with Indivisible Bids

**L. Julian Schwartzman and Michael P. Wellman** \*

University of Michigan

Computer Science & Engineering

Ann Arbor, MI 48109-2121 USA

{lschvart, wellman}@umich.edu

August 9, 2006

## Abstract

We study multi-unit double auctions accepting bids with indivisibility constraints. Modeling the auction problem as a Multiple Choice Knapsack Problem and using dynamic programming, we show that incremental computations during bid processing can speed the handling of key auction operations such as clearing and quoting. We propose different price-quote policies and study their influence on the efficiency of market-based allocation. Using a reconfigurable manufacturing scenario where agents trade large quantities of multiple goods, we demonstrate potential benefits of supporting indivisibility constraints in bidding. These benefits are highly sensitive to the form of price quote provided, indicating interesting tradeoffs in communication and allocation efficiency.

Keywords: Indivisible bidding, AON auctions, Incremental Multiple Choice Knapsack Problem, Quoting.

---

\*Parts of this work appeared in the Seventh International Workshop on Agent-Mediated Electronic Commerce (AMEC 2005).

# 1 Introduction

Consider a scenario with  $N$  manufacturing facilities with capabilities to produce various industrial parts. The facilities are controlled by different agents (e.g., firms, or profit-center divisions within the same large firm), and may vary in capacity, fixed and variable costs for producing the different part types, time for reconfiguring to switch between parts, transportation costs, and perhaps other factors. Each facility also has a set of customer orders, each representing a promise to pay a fixed amount contingent on delivery of a specified quantity of a particular type of part in the current period.

Since the facilities face heterogeneous cost structures, they stand to achieve potentially significant gains in efficiency by exchanging orders among themselves. We can formulate the order allocation problem as a global optimization, but of course the agents may not have the appropriate incentives to reveal their private information about costs and orders, or comply with the resulting order exchanges. Economic mechanisms such as combinatorial auctions (Cramton et al., 2006) can address these incentive problems, and provide an elegant solution when in fact they can be instituted. However, there are several organizational and computational impediments to holding large-scale (measured in numbers of goods and agents, and units per good) two-sided combinatorial auctions, and these are as yet uncommon in practice. It is substantially simpler to deploy individual two-sided multi-unit auctions for each of several goods, and these more ad hoc markets can address the allocation problem to a useful degree. Idealized models of such configurations as general-equilibrium systems demonstrate the potential of computational markets to achieve efficient allocations in convex, competitive environments (Cheng and Wellman, 1998; Ygge and Akkermans, 1999). Although the auctions in these markets operate independently, agents themselves account for good interactions and attempt to build bundles of multiple goods by conditioning their activities in one auction on the state of others. Designers of the US Federal Communications Commission (FCC) spectrum auctions similarly rely on bidders to account for preference dependencies in their pattern of bidding across simultaneous ascending auctions (McMillan, 1994).

Realistic configurations of multiple interacting markets differ from the idealized general-equilibrium model in several ways. One particularly important characteristic of this application domain is non-convexity in preferences and production technology, as manifest (for example) in fixed costs, re-

configuration switching costs, and preset order sizes. The most straightforward multi-unit auction mechanisms assume divisibility of offers: an agent willing to buy  $q$  units at some unit price would also be willing to accept  $q' \leq q$  units at that price. This assumption will not generally hold given nonconvex preferences and costs, and therefore agents with these characteristics may be hesitant to bid at all unless assured that their offers be accepted in whole or not at all.

Motivated by this manufacturing scenario, we investigate the design of multi-unit auctions accommodating such indivisibility constraints. Our focus is on how such auctions can be operated in a computationally efficient manner, and on the auctions' *price quote* policies for revealing information to agents to guide their bidding. We evaluate our designs experimentally, employing a version of the manufacturing scenario sketched above. Our main finding is that supporting indivisibility constraints can indeed improve the quality of global allocations achieved through trading, but actually realizing this improvement and to what degree depends pivotally on the form of the price quote. We also show how the computational costs of optimizing bid matching and producing meaningful quotes can be amortized over the auction's operation, calculated incrementally throughout the dynamic bidding process.

In the next section we present abstract examples illustrating the technical problems that divisibilities can cause for price-based allocation. We then describe in turn our auction mechanism, its incremental computation scheme, the model of our manufacturing scenario, and experimental results.

## 2 Examples

We present three examples showing some of the problems caused by divisible bidding. Our only assumption is that agents behave competitively, taking market prices as given and deciding their optimal allocation assuming they can buy or sell arbitrary number of units at the given prices.

EXAMPLE 1: Imagine a simple economy with two competitive agents,  $A$  and  $B$ , and a single traded good  $g$  which provides agents with a monetary utility as shown in Table 1.

Assume that  $A$  initially holds ten units of  $g$  and  $B$  holds none. In this scenario,  $A$  would be willing to sell five units of  $g$  for more than \$5, or ten for more than \$10. At the same time,  $B$  would

$H_g$	$U_A$	$U_B$
1	0	3
5	5	3
10	10	3

Table 1: Utility for agents  $A$  and  $B$  ( $U_A$  and  $U_B$ ), as a function of holdings of good  $g$  ( $H_g$ ).

be willing to buy one unit for  $\$b$  ( $b < 3$ ). If we used a standard auction accepting *divisible* bids, agents would trade *one* unit at some price between  $\$1$  and  $\$3$ , reducing the social benefit from  $\$10$  to  $\$8$ . At this point, if trading were allowed to continue,  $A$  would be willing to buy back one unit for less than  $\$5$ , and  $B$  would be willing to sell one unit for more than  $\$3$ . Trade would occur again, bringing the social surplus back to  $\$10$ . The process could continue indefinitely, reducing the net profit (i.e., utility plus cashflow) of  $A$  and increasing that of  $B$ . After each transaction, the social surplus would alternate between  $\$10$  or  $\$8$ .

In order to avoid those undesired exchanges,  $A$  could refuse to trade, or simply hedge the risk of a divisible trade by demanding arbitrarily more cash. However,  $A$  cannot decide (with certainty) whether such actions would prevent a profitable trade. Just imagine a third agent,  $C$ , interested in buying four units for less than  $\$b$  each but a total payment higher than  $5 - b$ . If  $C$  made its offer,  $A$  could engage in a profitable trade. Without  $C$ , however, trading would not be profitable for  $A$ .

The simplest solution would be to have the auction accept bids with indivisibility constraints. In this case,  $A$  would either sell five units for a total payment greater than  $\$5$ , or ten units for a total payment greater than  $\$10$ , or else no trade would occur. In this example, indivisibility constraints would always keep the social utility at its maximum possible value.

EXAMPLE 2: We have an economy with two agents,  $A$  and  $B$ , and two goods,  $g_1$  and  $g_2$ . Agents behave in a competitive manner. Their utility is shown in Table 2.

Assume that  $A$  initially holds three units of  $g_2$  and  $B$  holds three units of  $g_1$ . In this scenario, each agent gets a utility of  $\$3$ . If we allowed agents to trade using a standard auction for each good and *divisible* bids, agents could trade *one* unit of each  $g_1$  and  $g_2$ . Prices supporting such trades are shown in Figure 1(a). Once such exchange occurs, agent  $A$  holds two units of  $g_2$  and one unit of  $g_1$ ,

$U_A$				
$H_{g_1} \setminus H_{g_2}$	0	1	2	3
0	0	0	0	3
1	2	2	2	4
2	2	2	2	4
3	2	2	2	4

$U_B$				
$H_{g_1} \setminus H_{g_2}$	0	1	2	3
0	0	2	2	2
1	0	2	2	2
2	0	2	2	2
3	3	4	4	4

Table 2: Utility for agents  $A$  and  $B$  as a function of holdings of goods  $g_1$  and  $g_2$ .

and  $B$  holds two units of  $g_1$  and one unit of  $g_2$ , each agent getting a utility of \$2. The problem with this scenario is not only that both agents abandoned an optimal allocation and reduced their utility due to an undesired trade, but also that they cannot further change their new (lower) allocation (nor return to the previous one). Based on their demand curves and new holdings, there is no set of prices for  $g_1$  and  $g_2$  that would allow agents to engage in further trading. Prices at which  $A$  and  $B$  would be willing (but unable) to trade are shown in Figure 1(b). Note that agents could refuse to bid in the first place to avoid undesired trades, but they would face obstacles similar to those explained in the previous example. If we had just used bids with indivisibility constraints, however, the original (undesired) transactions would never have occurred, always keeping the utility at \$3 per agent.

**EXAMPLE 3:**

Imagine an agent  $A$  with the utility function provided in Table 3. Assume that  $A$  holds two units of  $g$ . This agent would be willing to buy two units at a price below \$2 each and sell two units at a price above \$1 each. However, a divisible bid expressing such intentions would be inconsistent (the bid is required to have buy prices lower than sell prices). Of course, we could set a cutoff price at say \$ $p$  ( $1 < p < 2$ ), and determine that  $A$  will buy below such price, or sell if the price is higher. The problem is that such a bid could prevent  $A$  from engaging in profitable trades, for instance if there were an offer from another agent to sell at \$ $p'$  ( $p < p' < 2$ ). An indivisible bid would avoid this problem, because it allows the expression of arbitrary valuations.

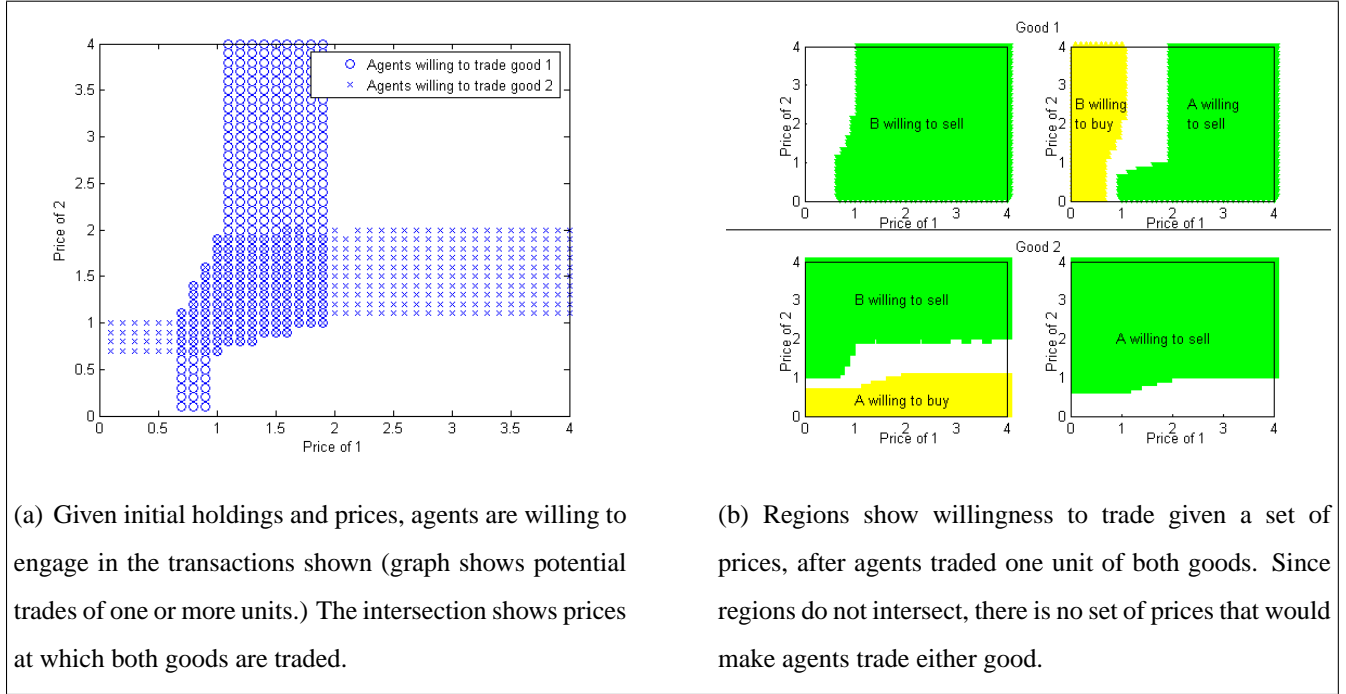


Figure 1: Prices at which transactions can (a) and cannot (b) occur, as described in Example 2.

$H_g$	$U$
0	0
2	2
4	6

Table 3: Utility as a function of holdings of  $g$ .

### 3 Auction Mechanisms

We consider separate two-sided auctions for multiple units of a single good. The auctions clear periodically at predefined intervals, and thus implement a *call market*. We distinguish two major versions of this auction, differing in their treatment of offer quantities. In the first (called “standard” for purposes of this paper), quantities appearing in bids are assumed divisible, and so the bidder effectively expresses a willingness to trade any amount up to the specified quantity at the associated unit price. In the second, offers are considered “all-or-none” (AON), and so agents explicitly specify the payment at which they would be willing to trade any acceptable discrete quantity. We

refer to this version as the “AON” auction henceforth.

In both auctions, agents may submit *bid schedules*, indicating the prices offered to trade various quantities (with negative quantities indicating amounts offered to sell). The points on the schedule are exclusive (i.e., treated as “XOR” (Nisan, 2000)), in that the resulting allocation will employ at most one of them. For divisible (standard) bids, the prices are per unit, and consistency requires that unit prices be nonincreasing in quantity. For indivisible (AON) bids, the prices represent total payments for the associated quantity, and these totals (not the per-unit payments) must be nondecreasing in quantity. Assuming only free disposal, with AON bids, agents can express arbitrary valuations for the good (Kelly, 2004; Nisan, 2000). Standard divisible bids can express only convex valuations.

Operation of the standard auction is relatively simple, as described, for example, by Wurman et al. (1998). Mechanisms resembling the AON auction have been described in the literature, and employed in practice. For example, van Hoesel and Müller (2001) consider the special case of combinatorial auctions where all goods are the same, and point out that optimal allocations can be found by dynamic programming. This corresponds to a one-sided, one-shot version of the AON auction. Kothari et al. (2003) present a one-sided, one-shot auction that supports AON bidding in the form of a minimum trade quantity, but then assumes divisibility for quantities beyond this minimum. Several other authors have considered indivisibility constraints in multi-unit auctions (Kalagnanam et al., 2001; Kellerer et al., 2004; Kelly, 2004), and have also identified the connection to knapsack methods for matching bids. Our understanding is that practical trading mechanisms admitting AON bids typically handle them in an ad hoc manner (Miller, 2002). For example, such bids might be matched in a greedy manner, as in common electronic stock trading systems, which just pass over AON bids if the entire quantity cannot be fulfilled. We describe details of the allocation algorithm, as well as other AON auction policies, in the sections below.

### **3.1 Winner Determination Algorithm**

As pointed out most explicitly by Kelly (2004), optimal winner determination for single-good, two-sided, multi-unit auctions with indivisible XOR bids (i.e., our AON auctions) reduces to the *Multiple Choice Knapsack Problem* (MCKP). MCKP is described thoroughly by Kellerer et al.

(2004); we present a formulation specialized (slightly) to the auction setting.

Consider a set of  $N$  agents, with agent  $i$  submitting bid  $B_i$ . Each  $B_i$  is comprised of  $m_i$  *bid points*,  $(p_{ij}, q_{ij})$ , specifying a payment  $p_{ij}$  offered to exchange quantity  $q_{ij}$ . Each bid includes a dummy point  $(0, 0)$ . Offers to buy are expressed as positive payment-quantity pairs, and offers to sell as negative payment-quantity pairs. Because the standard MCKP requires positive coefficients, we define transformed bid points  $(p'_{ij}, q'_{ij}) = (p_{ij} + \bar{p}_i, q_{ij} + \bar{q}_i)$ , where  $\bar{p}_i \equiv -\min_{j \in B_i} p_{ij}$ , and  $\bar{q}_i \equiv -\min_{j \in B_i} q_{ij}$ . (Note that this transformation affects only bids with sell points; for buy-only bids,  $\bar{q}_i = \bar{p}_i = 0$ .) We then define the knapsack *capacity*  $c \equiv \sum_i \bar{q}_i$ . Conceptually, the capacity  $c$  is the total number of units that are being offered for sale at any given time. We denote by  $C$  the maximum possible capacity (i.e., number of units that could possibly be traded). To ensure that  $C$  is bounded, we assume that agents have a limited ability to take short positions in the goods traded.

The MCKP is formulated as:

$$\begin{aligned} \text{maximize: } & \sum_{i=1}^N \sum_{j=1}^{m_i} p'_{ij} x_{ij} \\ \text{subject to: } & \sum_{i=1}^N \sum_{j=1}^{m_i} q'_{ij} x_{ij} \leq c, \\ & \sum_{j=1}^{m_i} x_{ij} = 1, i \in \{1, \dots, N\}, x_{ij} \in \{0, 1\}. \end{aligned}$$

We assume free disposal of units, reflected in allowing the auction to match bids with more sales than purchases. Excess units are allocated arbitrarily among sellers. Note that formulating and implementing the same problem without the assumption of free disposal is straightforward.

Solving MCKP is NP-hard, which is shown by reduction to a basic knapsack problem (Kellerer et al., 2004, p. 318). Using dynamic programming, however, the problem can be solved in pseudopolynomial time (Dudzinski and Walukiewicz, 1987). Let  $Z_l(d)$  be the value of the optimal solution to the MCKP defined to include only the first  $l$  agents,  $1 \leq l \leq N$ , and with restricted capacity  $0 \leq d \leq c$ . We further define  $Z_0(d) = 0$  for  $0 \leq d \leq c$ , and  $Z_l(d) = -\infty$  for  $d < 0$ .

We can characterize  $Z_l(d)$ ,  $1 \leq l \leq N$ ,  $0 \leq d \leq c$ , using the following recursion:

$$Z_l(d) = \max_{1 \leq j \leq m_l} Z_{l-1}(d - q'_{lj}) + p'_{lj}. \quad (1)$$

The optimal solution is obtained when  $l = N$  and  $d = c$ . Given  $N$  bids of maximum size  $m = \max_i m_i$ , the running time to solve MCKP using dynamic programming is  $O(mNc)$  (Kellerer

et al., 2004). In the worst case agents submit demand curves over the full range  $c = O(C)$  at the finest grain ( $m = O(C)$ ), so this running time is  $O(NC^2)$ .

Many different methods exist to solve MCKP, including branch-and-bound techniques and hybrid algorithms with diverse properties (see Kellerer et al. (2004) for an extensive review). Our implementation is customized for the dynamic auction context, which may call for repeated solution of the MCKP for small changes in the set of bids. Since these computational issues are separable from the policy implemented by the auction mechanism, we defer discussion of this algorithm to Section 4.

### 3.2 Clearing and Pricing

*Clearing* the auction is the process of identifying the subset of bids that match and produce the highest possible surplus. The result of a clear operation is to determine the deals resulting from this matching, and removing the matched bids from the order book. Given the incremental calculations described in Section 4, most of the work is performed when bids are inserted into the order book. Once this is done, identifying the match takes constant time. Extracting the deals takes time linear in the number  $N'$  of bids matched. Modifying the order book to include only unmatched bids requires  $N'$  deletion or  $(N - N')$  insertion operations, which are described below.

For a fixed allocation of goods, monetary transfers do not affect overall efficiency. Therefore, since we are not addressing strategic issues in this work, the pricing choice is not pivotal for our experimental analysis. Nevertheless, to fully specify the mechanism one must identify a pricing rule. Ours starts with Vickrey prices and adjusts them proportionally to ensure budget balance. The Vickrey calculation requires that we compute the total surplus with each agent's bid excluded, for which  $O(N')$  deletion and insertion operations need to be performed.

### 3.3 Quoting

After each bid, the auction issues a *price quote*, providing to the agents some information regarding the state of the order book, intended as a guide to future bidding. In the standard auction, the quote comprises a BID-ASK pair, representing the prices at which an agent could successfully trade at least one unit. The BID quote defines the price at which an agent could sell one unit, and the ASK

quote the corresponding price to buy. For standard (divisible-bid) auctions, we can incrementally maintain the order book so that price quotes can be provided in constant time once the bids are inserted (Wurman et al., 1998).

For the AON auction, it is not immediately apparent how the auction should define its price quotes. We identified four candidate quoting policies, described here and compared experimentally in Section 6 below.

### 3.3.1 Standard Quote

One possibility is for the AON auction to provide a “standard” quote, defined as the BID-ASK pair reflecting the order book interpreted *as if the bids were divisible*. Constructing this interpretation requires some care, since simply treating each bid point as a divisible offer may violate the standard auction’s consistency condition requiring that quantity be nonincreasing in unit price. To ensure this monotonicity, we transform each bid  $B_i$  by first sorting the bid points  $(p_{ij}, q_{ij})$  (not including the dummy point with  $p_{ij} = q_{ij} = 0$ ) in decreasing order of unit price. We then traverse the list, translating each to a unit-price bid point, skipping any that would violate the monotonicity condition with respect to those already seen.

These translated bids can then be handled by the order book and quoting algorithm of the standard auction.

### 3.3.2 Marginal Unit Quote

A second quote candidate attempts to maintain the interpretation of the standard quote as a price threshold sufficient to trade *one unit*, but respecting the indivisibility constraints of AON bids. Calculating this quote requires solving the MCKP for the bids in the order book. Under this interpretation, the ASK quote is always defined as long as there is any sell offer in the order book. The same is not true for the BID quote, however, because it could be the case that no existing offer or combination of offers can be satisfied by contributing a single additional unit. The marginal unit quote takes the same form as the standard quote, but provides more conservative values. Indeed, it is even possible (and consistent) for the ASK price to be lower than the BID price, something that cannot happen in the divisible case.

Given the incremental computation scheme discussed in Section 4, these quotes can be extracted from the order book in constant time. It would also be possible to define this quote with any particular quantity defined as “marginal” (e.g., ten units instead of one).

### 3.3.3 Anonymous Full Schedule Quote

The third quote we consider provides to all agents a full schedule of payments that would be required to exchange any feasible quantity given the current state of the order book. This can be viewed as a collection of marginal unit quotes, one for each feasible quantity. The quote is anonymous because the same values are provided to every agent. Note that only relevant payment-quantity pairs need to be communicated to an agent: for a given payment, a quote for the minimum number of units the agent needs to sell to get such payment, and the maximum number of units the agent can buy with such payment. As for the marginal unit quote, the schedule may not be monotone: the unit price to exchange various quantities may be increasing or decreasing or mixed along the schedule.

Also like the marginal unit quote, the full schedule quote can be extracted directly from the order book given our incremental computation scheme, though of course extracting and communicating it will take time proportional to its size,  $O(C)$ .

### 3.3.4 Non-Anonymous Full Schedule Quote

The final quote we consider is similar to the previous one, but each agent is provided with personalized values based on its existing bid. More specifically, the quote provides agent  $i$  the schedule of payments calculated by excluding from the order book the bid sent by  $i$ .

This quote generalizes the “shortfall” idea developed for the US FCC combinatorial spectrum auction number 31 (see Federal Communications Commission (2000) and subsequent FCC Notices). In this auction, bid increments are a function of the difference, or shortfall, between the revenue of a provisional winning bid and the maximum total revenue of a particular package. The non-anonymous full schedule quote we employ provides agents with the shortfall for any possible multi-unit (and single good) indivisible bundle.

### 3.3.5 Quote Discussion

The four candidate quotes present distinct tradeoffs. The standard and marginal-unit quotes are compact, but may provide inaccurate guidance for trading particular quantities. The full schedule quotes provide high-fidelity information, but may be too large to be reasonably communicated in some applications.

We explore the implications of the various quote policies in our experiments below. Of course, the worth of a quote is intimately tied to how the agents use this information in their bidding. We discuss our assumptions about agent behavior in Section 6.2 below.

## 4 Incremental Clearing Algorithm

In a periodic auction, the basic operations of clearing and quoting may be invoked many times, often with bidding states only slightly different from previous states that have already been solved. We therefore developed an incremental version of the clearing algorithm, designed to minimize the average solution time over a sequence of auction operations.

Our method builds upon the algorithm DP-with-Lists proposed by Kellerer et al. (2004, p. 50), exploiting a separability property identified by Horowitz and Sahni (1974). DP-with-Lists does not change the worst-case running time of the standard dynamic programming procedure, but improves computation in practice by considering quantity sparseness and by pruning for dominance. The algorithm considers each pair  $(d, Z_l(d))$  of the dynamic programming table as a state  $(\bar{q}, \bar{p})$  where  $\bar{q}$  denotes the capacity and  $\bar{p}$  denotes the profit obtainable for such capacity when considering the subproblem on the first  $l$  bids. Lists of states, one corresponding to each bid, are consecutively pruned for dominance (instead of considering all possible quantities) and merged with one another to get a solution. The partitioning technique of Horowitz and Sahni (1974) divides a problem into two equally sized parts, solves each using dynamic programming, and combines the results in linear time by keeping the sets sorted. Their approach provided a square root asymptotic improvement over a complete enumeration of  $O(2^n)$ . Other authors have focused on issues related to our problem, for instance Bassamboo et al. (2001) consider online bid processing in single-good multi-unit auctions with indivisibility constraints but only for prices that are nonincreasing in quantity, and others conducted probabilistic analysis of online knapsack problems (Marchetti-Spaccamela

and Vercellis, 1995; Lueker, 1998).

The most straightforward approach to incremental MCKP computation using dynamic programming would be to add one bid at a time and store the solution up to that point (i.e., using DP-with-Lists). Each new bid arrival would simply be added to the existing solution. In the worst case, however, the first bid that got inserted could later get replaced or deleted, and thus the knapsack solution would have to be recalculated for the entire order book. Then, each bid insertion would take  $O(NC^2)$ .

The algorithm we present improves significantly such worst-case running time to  $O(C^2 \log N)$ , as shown below. Our implementation is inspired by the two-part decomposition of Horowitz and Sahni (1974), but differs from existing methods in that we partition the knapsack solution by  $N$ , the number of bids in the order book, and arrange these partitions in a binary tree structure merging them pairwise from leaves to root. This partition arrangement is the key feature enabling incrementality in our auction setting.

## 4.1 Algorithm

The basic solution method remains a form of dynamic programming, in order to provide pseudopolynomial time guarantees on the auction operations. The idea is to maintain a binary tree with  $N$  leaf nodes, one for each bid inserted into the order book. A leaf node for bid  $i$  stores a list of states with pairs  $(\bar{q}_{ij}, \bar{p}_{ij})$ , where  $\bar{p}_{ij} = Z(\bar{q}_{ij})$ , the maximum profit that could be achieved for the given capacity when considering the subproblem defined for bid  $i$  only. Upon insertion of a new bid, a leaf is added to the tree, and its list of states is simply the set of  $m_i$  bid points in the bid (suitably transformed as specified in Section 3.1).

Once a new leaf is added, all  $O(\log N)$  parents are (re)calculated from bottom to top. Conceptually, each parent stores a merge of the lists of its successors, equivalent to all undominated pairs  $(\bar{q}_{ij} \leq C, \bar{p}_{ij})$  resulting from the cross sum of the children (a state  $(\bar{q}, \bar{p})$  dominates  $(\bar{q}', \bar{p}')$  if  $(\bar{q} < \bar{q}' \wedge \bar{p} \geq \bar{p}')$  or  $(\bar{q} \leq \bar{q}' \wedge \bar{p} > \bar{p}')$ ). Dominated states can never be part of the optimal solution (Kellerer et al., 2004). Upon merging, each state stores pointers to all the bid points that were added to create it, which are used later to determine bid matchings. In addition, a node stores the cumulative capacity of its subtree and bid transformation  $\bar{p}_i$ , as defined in Section 3.1.

Calculation proceeds up to the root of the tree, which stores a list of states with the maximum profit achieved for every capacity and every bid in the tree. The root, thus, provides the solution to MCKP by consolidating all the lists of every bid. Figure 2 shows a sample insertion.

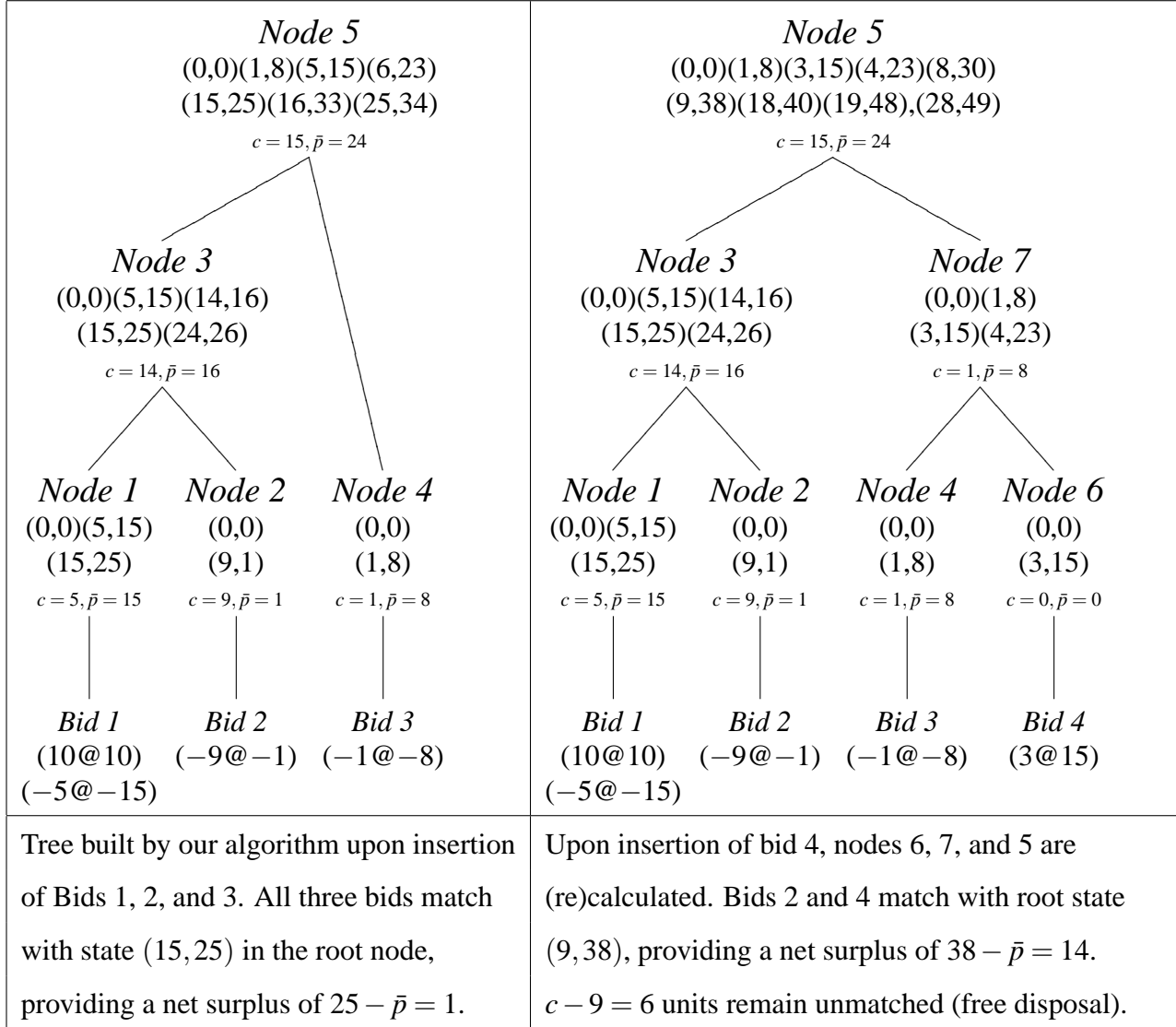


Figure 2: Each node in a tree stores a list of states, capacity, and bid transformation  $\bar{p}_i \equiv -\min_{j \in B_i} p_{ij}$  as defined in Section 3.1. Pointers corresponding to each state, which are also stored in the tree, are not shown. The solution is found in the root of the tree, in the state with highest  $q \leq c$  (largest state that fits in the knapsack).

Given two nodes with lists  $L_a$  and  $L_b$ , with  $k_a$  and  $k_b$  states each, the following algorithm merges them to create a parent node:

**Algorithm**  $Merge(L_a, L_b)$

- 1:  $M = ()$  {an empty list}
- 2: **for**  $t = 1$  to  $k_b$  **do**
- 3:  $L'_a := L_a \oplus (\bar{q}_{bt}, \bar{p}_{bt})$   
    {add state  $t$  from list  $L_b$  to each element in  $L_a$ }
- 4: delete all states  $(\bar{q}, \bar{p}) \in L'_a$  with  $\bar{q} > C$
- 5:  $M = \text{Merge-Lists}(M, L'_a)$
- 6: **end for**
- 7: return  $M$

**Procedure**  $\text{Merge-Lists}(L, L')$ , similar to the one provided by Kellerer et al. (2004) (p. 52)

add the state  $(\infty, 0)$  to the end of both lists  $L, L'$

$L'' := ((\infty, 0))$

**repeat**

choose the state  $(\bar{q}, \bar{p})$  with the smallest quantity in  $L$  and  $L'$  and delete it from the corresponding list

**if**  $\bar{q} \neq \infty$  **then**

assume that  $(\bar{q}'', \bar{p}'')$  is the largest (last inserted) state in  $L''$

**if**  $(\bar{p} > \bar{p}'')$  **then**

**if**  $(\bar{q} \leq \bar{q}'')$  **then**

delete  $(\bar{q}'', \bar{p}'')$  from  $L''$

**end if**

add  $(\bar{q}, \bar{p})$  to the end of list  $L''$

**end if**

**end if**

**until**  $\bar{q} = \infty$

return  $L''$

The algorithm begins with an empty list  $M$ . In every iteration (line 2), it picks a state  $t$  from the second list  $L_b$  and adds it to the first list  $L_a$  (line 3). This componentwise addition, denoted

with the symbol  $\oplus$ , creates a new list  $L'_a$  which includes the set of undominated states that could be reached by combining all the states in  $L_a$  with state  $t$  from list  $L_b$ . All the states in  $L'_a$  with a capacity greater than  $C$  are eliminated, because they do not contain a feasible solution. Then, the algorithm merges  $L'_a$  with  $M$ , the list that will ultimately be returned. Once all  $k_b$  states in  $L_b$  got chosen and the *for* loop ends,  $M$  effectively contains the list of undominated states that can be obtained by merging all the states of both  $L_a$  and  $L_b$ . The procedure *Merge-Lists* simply merges two sorted lists, keeping undominated states only. It is explained in detail by Kellerer et al. (2004).

## 4.2 Analysis

Lines 3 and 4 are both linear in the number of states in a list. Since the lists are kept sorted, line 5 is also linear in the number of states in a list. Given that the *for* loop in line 2 iterates over every state in a list, each merge of two lists takes a computation time proportional to the length of the lists, squared. Given  $N$  bids and a maximum number of bundles per bid  $m$ , the maximum length of any list in the tree is given by  $\min(C, m^N)$ . Thus, the computation for each bid insertion is bounded by  $O(\min(C, m^N)^2 \log N)$ . Note that lists of states in non-leaf nodes are kept sorted in the procedure *Merge-Lists* without incurring an additional cost. Only states stored in leaf nodes require sorting, which does not change the asymptotic time. When requiring a MCKP solution on every bid update—typically to calculate a quote—our algorithm provides an improvement over a non-incremental DP-with-Lists approach from linear to logarithmic time in the number of bids in the order book.

Bid replacement is analogous to bid insertion, with the difference that no new leaves are added to the tree but the old one corresponding to the agent doing the update gets overwritten. The time taken for a replacement is the same as the one needed for an insertion, given that a similar number of nodes needs to be recalculated. We currently treat a deletion as a replacement with an empty bid. If we needed to truly eliminate a bid, the updates to the structure of the tree would not change the worst-case running time.

If we had  $N$  bids, our tree would include  $N$  leaves. Space requirements are given by  $O(N^2C)$  to store in every node a list of up to  $C$  states, each of which stores pointers to up to  $N$  bid points that were merged to create such state. Note that we could reduce storage to  $O(NC)$ , by storing in

each state pointers to the children states that were merged to create it (instead of pointers to each bid point). This, however, would increase the time to extract matching points.

We evaluate the performance of our incremental algorithm in a series of experiments described in Section 6.6.

### 4.3 Extracting Matched Bids

As noted above, once all the bids are inserted into the tree, clearing takes constant time and extracting the deals takes  $O(N')$ . Given a set  $S$  of states in the root of the tree, the solution is in state  $t$ , where  $\bar{p}_t \geq \bar{p}_s$  for all  $s \in S$ , and  $\bar{q}_t \leq c$ . Such state has direct pointers to the matching bid points of each agent. The solution state can be identified upon merging both children of the root at no additional cost. In order to change the order book to include only unmatched bids, we could either remove  $N'$  matched bids from the order book in time bounded by  $O(C^2 \log N)$  for each bid, or discard the entire order book and reinsert the remaining  $N - N'$  bids. Note that reinserting  $N - N'$  bids *in batch* takes only  $O((N - N')C^2)$ , to calculate a number of parent nodes bounded by  $N - N'$ , each of which involves a merge that takes  $O(C^2)$ , as explained above. The time complexity to adjust the order book is then  $O(\min(N'C^2 \log N, (N - N')C^2))$ .

### 4.4 Calculating Quotes

Once bids are inserted into the tree, the information required to generate quotes that take into account indivisibility constraints is readily available in the root node. Given root-node states  $S$  and a solution state  $t$  with  $\bar{p}_t \geq \bar{p}_s$  for all  $s \in S$  and  $\bar{q}_t \leq c$ , a quote for the marginal  $v$  units,  $1 \leq v \leq C$ , is given by:

$$\text{BID} = \max_{s \in S | \bar{q}_s \leq c+v} (\bar{p}_s - \bar{p}_t),$$

$$\text{ASK} = \min_{s \in S | \bar{q}_s + v \leq c} (\bar{p}_t - \bar{p}_s).$$

Conceptually, BID is equivalent to the incremental profit obtained if we sold  $v$  additional units (i.e., assuming we increased the capacity of the knapsack by  $v$  units), and ASK is equivalent to the additional profit required to match the current maximum surplus if we bought  $v$  additional units

(i.e., fit  $\nu$  additional units into the knapsack). We can perform the calculation upon creating the root node with no additional asymptotic cost.

Similarly, when calculating an anonymous full schedule quote, each state in the root node provides a quote point with either a BID or a ASK payment, as follows:

$$\begin{aligned} \text{QUANTITY}_s &= |\bar{q}_s - c| \\ \text{BID}_s &= (\bar{p}_s - \bar{p}_t) \text{ if } \bar{q}_s > c \\ \text{ASK}_s &= (\bar{p}_t - \bar{p}_s) \text{ if } \bar{q}_s \leq c \end{aligned}$$

Since we have up to  $C$  states in the root node, the extraction of this quote takes time  $O(C)$ .

The non-anonymous version of the full schedule quote requires a temporary deletion of the bid of each agent in the order book, in order to calculate a personalized quote that excludes the agent's own bid. If quotes are calculated upon bid insertion, the computation of each insertion is increased to  $O(C^2 N \log N)$ , while extraction remains  $O(C)$ .

## 5 Manufacturing Domain

We evaluate the AON auction in a market-based allocation problem based on the manufacturing scenario sketched in the Introduction. The setting comprises a set of  $N$  manufacturing *modules*, defined as arrangements of (possibly reconfigurable) manufacturing machines, with capabilities for producing a variety of parts. Each module is controlled by an agent, whose objective is to maximize profit by fulfilling customer orders over an  $L$ -day production period. In our market-based model, agents may increase their individual and collective profit by exchanging orders among themselves, thus exploiting their comparative advantages and configuration decisions.

We provide a full specification of the model below, describing the goods traded, utility and cost functions of the manufacturing modules, and the market configuration. Specific parameter settings for the model, and trading policies implemented by agents in our simulations, are described in Section 6.

## 5.1 Goods Traded

The core allocation problem in this domain is deciding which manufacturer will produce what quantity of each of  $M$  types of parts in the current period. The total quantity demanded of part type  $r$  is  $D_r$ , and initially each agent is given orders for some share of that demand. Producing part  $r$  entitles the manufacturer to a fixed income of  $I_r$  per unit, up to the number of units for which it holds orders.

The purpose of the market is to enable trading of orders among manufacturing modules. The goods traded are the rights to produce parts for orders. A unit of good  $r$ , therefore, entitles the holder to produce a unit of the corresponding part and receive the corresponding payment  $I_r$  from the customer. The parameter  $D_r$  bounds the maximum quantity of good  $r$  that can be exchanged at one time, and thus plays the role of  $C$  in the definition of the AON auction in Section 3.1.

## 5.2 Agent Objectives

Agents aim to maximize profit, defined as

$$\text{income} - \text{production costs} + \text{trading cash flow.}$$

Income is simply the total payment for producing parts. Trading cash flow represents the balance of payments from trading orders with other agents. Production costs include several components, depending on the quantity and types of parts produced. These are defined by a set of agent-specific parameters:

- $F_i$ : Fixed cost, a one-time payment if module  $i$  produces one or more parts.
- $B_i$ : Labor cost, paid for every day in which the module is in production.
- $V_{i,r}$ : Variable cost, paid for each unit of part  $r$  that gets produced.
- $G_i$ : Set of possible *configurations*. Each manufacturing configuration provides distinct production capabilities. Only one configuration can be used in any given day. For each configuration  $f \in G_i$ , each module has:
  - $P_{f,r}$ : Production capacity per part type: a bound on the units of type  $r$  produced per day.

- $R_f$ : Reconfiguration cost to be paid if the configuration is used.
- $T_f$ : Reconfiguration time (in days) that takes the agent to set up configuration  $f$ , during which no part can be produced.

The configuration capacities and times, along with the period length  $L$ , define the production possibilities for module  $i$ . The various cost parameters define the total cost for any feasible production plan.

Although complicated, the foregoing determines well-defined optimization problems for the agent:

- Determining an optimal production plan given holdings of goods  $r$ .
- Determining optimal demand for goods  $r$  given current holdings and market prices.

### 5.3 Market Game Configuration

The overall market system comprises the agents representing manufacturing modules, plus one auction for each part type. The simulations are implemented using our configurable market game server, AB3D (Lochner and Wellman, 2004), developed at the University of Michigan. AB3D provides a flexible bid-processing architecture, with a rule-based scripting language to specify particular auction policies and temporal control structure. The standard call market was already supported by AB3D. To handle indivisible bidding, we added a new bid language specifying quantity-payment schedules, and new matching, pricing, and quoting modules.

We simulate an instance of this setup by generating parameter values from prespecified probability distributions, and communicating these values to the respective agents. Each agent is initially allocated customer orders corresponding to equal shares,  $D_r/N$ , of the overall demand for each part  $r$ .

Each game instance lasts twenty minutes, with each auction clearing periodically every 48 seconds. The auctions are staggered, so that the initial clears occur at multiples of  $48/M$  seconds.

The agents operate asynchronously, submitting bids to the auctions iteratively according to the policy described in Section 6.2. Agents can request price quotes reflecting the latest auction state, and retrieve notices of any transactions from prior bids.

At the end of a game instance, the server calculates final holdings based on cumulative transactions, and determines a score for each agent. The score depends on an agent’s production plan given its total available orders, which entails solving an optimization problem for each agent. AB3D solves these using a commercial optimization package (AMPL/CPLEX), given an integer linear programming (ILP) formulation specified as part of the game description.

The overall value of the resulting allocation is simply the sum of the scores over the  $N$  agents. For comparison, we can also calculate (offline if necessary) the global optimum of the system without trading, assuming a central planner that can allocate orders across manufacturing modules.

## 6 Experiments

We conducted a series of experiments in order to compare allocation performance using standard or AON auctions, and to measure the computational savings obtained with our incremental algorithm.

In order to compare allocation performance using both standard and AON auctions, we ran a set of 58 and 216 paired trials with 4-agent and 8-agent games, respectively. For AON auctions, we tested *standard*, *marginal*, and *full schedule* (both anonymous and non-anonymous) quotes. The following sections describe the specific problem instance we chose for our manufacturing scenario, the behavior of the agents, and the results obtained.

### 6.1 Manufacturing Problem Setting

For each trial, we obtained a new set of randomly chosen parameter values, as specified in Table 4. Each paired trial used the same set of parameter values, and compared all five alternatives, namely, standard auctions and AON auctions with the four quoting alternatives discussed. Note that these parameter values were chosen not to explore the resulting space exhaustively, but to provide an interesting range of problems for our experiments.

### 6.2 Agent Bidding

Our design maintains the assumption of competitiveness, modeling agents essentially as price takers in defining their bidding policies. Each agent refines its bid incrementally (Cheng and

Parameter		Values		
		(a)	(b)	
General	# agents ( $N$ )	4	8	
	# parts ( $M$ )	4	4	
Public information	$I_r$	[1000, 2000]	[1000, 2000]	
	$D_r$	[2000, 6000]	[4000, 12000]	
	$L$	[250, 300]	[250, 300]	
Private information for agent $i$	$F_i$	[300000, 400000]	[300000, 400000]	
	$B_i$	[15000, 20000]	[15000, 20000]	
	$V_i$	[250,350]	[250,350]	
	# configs ( $ G_i $ )	2	2	
	For each $f \in G_i$	$P_f$	[20,60] (*)	[20,60] (*)
		$R_f$	[400000, 800000]	[400000, 800000]
		$T_f$	[5,15]	[5,15]

Table 4: Settings of the manufacturing scenario used for our experiments. Column (a) shows parameters for 4-agent games, and column (b) shows parameters for 8-agent games. Parameters specifying a range are drawn from a uniform distribution. (\*) parameter specifies total for all parts in a configuration, each part getting a random proportion.

Wellman, 1998), according to the following procedure.

**Main loop:**

- 1: **repeat**
- 2:   Get price quotes.
- 3:   Get transactions (i.e., matching bids).
- 4:   **for** each auction  $g$  (an auction  $g$  corresponds to a different good  $r$ ) **do**
- 5:     Select a new point to be added to the bid in  $g$  (see 6.3).
- 6:     Fix inconsistencies in bid.
- 7:     Submit updated bid to  $g$ .
- 8:   **end for**

9: **until** Timeout {allocation process is over}

Fixing inconsistencies after adding a new bidding point requires making the smallest possible changes to the old points in the bid in order to maintain divisible prices nonincreasing in quantity and indivisible payments nondecreasing in quantity.

The results described in Section 6.4 were obtained by using the same agent structure, with some variations in terms of selection of new bidding points which are explained below.

### 6.3 Selection of New Bidding Points

In each iteration  $a$  of the main loop, an agent updates its bid for the good in auction  $g$  with one new point  $(q_{g,a}, p_{g,a})$ , taking into account current holdings and assuming that other goods (not in auction  $g$ ) could be freely bought or sold at the most recent quote. This key assumption allows an agent to account for the existing interactions among the different goods, which trade in auctions that function independently.

We used two different methods for picking incremental points, one for dealing with divisible bids and another for indivisible ones.

**DIVISIBLE:** For divisible bids, an agent selects a new bidding point for the good in auction  $g$  by picking a price  $p_{g,a}$  and calculating the quantity  $q_{g,a}$  the agent would be willing to buy or sell at such price in order to maximize its profit. Calculation is done using an ILP model that encodes the agent's utility function as explained in Section 5.2. Prices  $p_{g,a}$  are selected in the following arbitrary order:

1.  $p_{g,a} = \text{BID}$
2.  $p_{g,a} = \text{ASK}$
3. If the bid in  $g$  already contains prices for 1 and 2 above,  $p_{g,a}$  is selected from a normal

distribution  $N(\mu, 1)$ ,

$$\mu = \begin{cases} (hb + \text{ASK})/2 & \text{if } q_{a-1} > 0 \vee (q_{a-1} = 0 \wedge pr < .25) \\ (ls + \text{BID})/2 & \text{if } q_{a-1} < 0 \vee (q_{a-1} = 0 \wedge pr < .5) \\ lb & \text{if } (q_{a-1} = 0 \wedge pr < .75) \\ hs & \text{otherwise} \end{cases}$$

where  $hb$  ( $hs$ ) and  $lb$  ( $ls$ ) are the highest buy (sell) and lowest buy (sell) offers already in the bid and  $pr$  is a random value uniformly distributed between 0 and 1.

(Note that BID and ASK refer to the most recent quote obtained by the agent.)

The basic idea behind the approach described above is to help agents find feasible trades by gradually making them place their highest buy and their lowest sell offers. We empirically tested other alternatives to ensure that our comparison of divisible versus indivisible bidding was not biased by an unreasonable point-selection approach. Specifically, we compared the procedure described with a random selection of points analogous to the one described by Cheng and Wellman (1998) in their *incremental bidding*, and also with another in which prices are picked by finding the maximum possible gap between any two consecutive pairs of (sorted) prices already in the bid and selecting their average. Our results indicated that the approach chosen provided the best average performance among the alternatives we evaluated.

**INDIVISIBLE:** For indivisible bidding, the agent selects a new bidding point for the good in auction  $g$  by picking a quantity  $q_{g,a}$ . The payment  $p_{g,a}$  is given by the maximum (minimum) value at which the agent is willing to buy (sell)  $q_{g,a}$  units, which is calculated using an ILP model that encodes the agent's utility function as explained in Section 5.2. Quantities  $q_{g,a}$  are selected in the following order:

1.  $q_{g,a} = -H_{g,a}$  (sell all holdings available in iteration  $a$ )
2.  $q_{g,a} = D_g - H_{g,a}$  (buy all available items, i.e., demand minus holdings)
3.  $q_{g,a} =$  random value uniformly distributed in the range  $[-H_{g,a}, D_g - H_{g,a}]$  (excluding 0)

4. If the bid already contains quantities for 1, 2, and 3 above:

$$q_{g,a} = \begin{cases} -H_{g,a} & \text{with probability .1} \\ D_g - H_{g,a} & \text{with probability .1} \\ \text{average between any two} \\ \text{consecutive (sorted) quantities} \\ \text{that are further apart} & \text{with probability .8} \end{cases}$$

The method described gradually fills the largest gaps in the bid being constructed, and “refreshes” each extreme occasionally with a 0.1 probability.

## 6.4 Allocation Results

The average efficiency relative to a global optimal allocation (i.e., assuming an offline central planner) as calculated from our 58 and 216 trials is given in Table 5. Results show that AON auctions quoting full schedule provided the highest efficiency, and that AON auctions using either a standard or marginal unit quote performed worse than standard auctions with divisible bids. The non-anonymous version of the full schedule quote appears to provide a slightly higher efficiency than its anonymous counterpart, although such difference appears to decrease when more agents participate in a game. This decrease is to be expected, as the anonymous and non-anonymous quotes tend to be more similar when more agents are bidding. Using AON auctions with standard quotes and marginal units provided the lowest average efficiency.

We are not suggesting based on this particular experiment that the differences shown in Table 5 will be an indicator of the differences to be found under any possible parameter configuration of our manufacturing scenario or other settings. Before we ran the systematic paired tests described above, we informally experimented with other parameter settings. Even though we observed that AON auctions quoting full schedules always provided the best average efficiency, in several settings the differences detected were not as stark.

#	Auction	Quote	Average efficiency	
			4 agents - 58 trials (a)	8 agents - 216 trials (b)
1	AON	Non-Anonymous Full schedule	92.8%	87.8%
2	AON	Anonymous Full schedule	91.3 %	87.2%
3	Standard	Standard	79.2%	75.2%
4	AON	Marginal unit	70.7 %	47.0%
5	AON	Standard	61.7%	49.7%

Table 5: Results of paired trials calculated as average efficiency in terms of a global optimal. Differences between 1a and 2a are significant at the .06 level, 2a and 3a at the  $10^{-7}$  level, 3a and 4a at the .03 level, 4a and 5a at the 0.07 level, 1b and 2b at the .23 level, 2b and 3b at the  $10^{-16}$  level, 3b and 5b at the  $10^{-28}$  level, 5b and 4b at the .12 level.

## 6.5 Influence of Quoting

Quoting marginal prices with standard auctions makes sense from two perspectives. First, it provides an accurate value for marginal units in order for agents to construct their bids. Second, it provides a lower (upper) bound on both the unit price and total payment to be paid (received) when bidding to buy (sell) an arbitrary number of units. On the contrary, marginal values for AON auctions do not contain the same valuable information. By assuming divisibility with AON auctions (i.e., using a standard quote), the marginal value provided is neither accurate for the marginal nor a bound on the price of or total payment for additional units. In this case, the quote provides a very loose approximation of value. Similarly, if we take into consideration indivisibility constraints when quoting marginal values with AON auctions, the resulting quote turns out to be very conservative. The reason is that this quote can very often be undefined for the BID (in practice we need an agent or combination of them who value and intend to buy a single unit), and the ASK can often be excessively high (i.e., when the auction matches bids with sell quantities much larger than the marginal). Moreover, little information is provided by this quote regarding additional units beyond the marginal: we know nothing about the unit price of additional units, and we only know that the total payment for more units will be no less than the price for a single unit (which is a very loose

bound or almost like having no bound at all).

The effects described above were confirmed in part by measurements applied to our 4-agent simulation results. We define a trade as *desired* (A) with respect to agent  $i$  if, once executed, it increases or maintains the profit (i.e., income minus costs plus cashflow) of this agent, assuming that no further trades occur. We identified three possible reasons for agent  $i$  to engage in undesired trading: *outdated information* (B1), *misleading non-anonymous quotes* (B2), or *misleading anonymous quotes* (B3). A bid can contain outdated information because its points were calculated incrementally or due to the asynchronous nature of the bidding process. Outdated information (B1) thus refers to the case in which  $i$  engaged in a trade that it would have rejected had it reevaluated its bid using the most up-to-date information. Such up-to-date information includes most current holdings and non-anonymous quotes (i.e., quotes calculated by excluding from the order book the bid sent by  $i$ ). Misleading non-anonymous (B2) or anonymous (B3) quotes are those that made  $i$  believe that it could buy or sell goods at the quote, when that was actually not possible. Finally, every transaction that decreased utility and cannot be explained by outdated information or misleading quotes must have occurred because of *dependency on other auctions* (C). Such dependencies exist because agents construct their bids in an auction assuming they could trade in other auctions at the quote. Since communication is asynchronous and auctions clear at different times, some intermediate decreases in utility are normal and expected.

Suppose we had  $T$  transactions, and transaction  $t$  ( $1 \leq t \leq T$ ) occurred in auction  $g$  for quantity  $q_t$  and payment  $p_t$ . We perform two different optimizations for each agent  $i$  (see Section 5.2):

- $R^*(H)$  is the profit achieved by  $i$  when calculating its optimal production plan based on holdings  $H$ , assuming that  $i$  cannot trade further.
- $P^*(H, Q)$  is the highest payment that  $i$  is willing to offer to trade quantity  $q_t$  in auction  $g$ , assuming it holds goods  $H$  and that it could freely trade goods in auctions other than  $g$  at the prices given by quotes  $Q$ .

We further define  $Q_i$  as the most up-to-date non-anonymous quotes for agent  $i$ , and  $Q$  as the most up-to-date anonymous quotes. Holdings  $H^t$  are the goods held by  $i$  in all auctions right after  $t$  occurred; holdings  $H^0$  are based on initial endowments; holdings  $H^{t_i}$  and  $H^{t'}$  are the goods held right after  $t$  and an hypothetical clear of all auctions other than  $g$  occurred, assuming that  $i$  bid to

achieve optimal holdings as calculated for  $P^*(H^{t-1}, Q_i)$  and  $P^*(H^{t-1}, Q)$ , respectively; and  $p'_t$  is the lowest hypothetical payment that  $i$  could have bid in order to trade  $q_t$  in  $g$ . Negative coefficients for payments and quantities are used for sell offers.

Given these definitions, we can classify transaction  $t$  for agent  $i$  as follows.

- (A) Is desired If  $R(H^{t-1}) \leq R(H^t)$
- (B1) Occurred due to  
outdated information If  $P^*(H^{t-1}, Q_i) < p'_t$
- (B2) Occurred due to a  
misleading non-anonymous quote If  $R(H^{t_i}) < R(H^{t-1})$
- (B3) Occurred due to a  
misleading anonymous quote If  $R(H^{t'}) < R(H^{t-1})$
- (C) Was necessary due to  
auction dependencies If  $t \notin A, B1, B2, B3$

Using the same data obtained for the experiments reported in Table 5, we measured the percentage of transactions in (A), (B1), (B2), (B3), or (C) for the different quoting mechanisms we tested with AON auctions. The results are shown in Table 6.

We notice that the best performance was provided by the Non-Anonymous Full Schedule quote, which cannot be misleading and provided the highest proportion of desired trades. The second least misleading was the Anonymous Full Schedule quote. Finally, the marginal unit and standard quotes were similarly misleading, providing comparable results regardless of whether the quote was anonymous or not. Thus, personalizing these two quotes for decision making does not seem to help. Marginal unit quotes appeared to provide a relatively high percentage of desired trades, which is somewhat expected given such a highly conservative quote. The overall results provided by this quote, however, were relatively poor (see Table 5).

## 6.6 Performance of the Incremental Algorithm

In order to evaluate the performance of the incremental algorithm described in Section 4.1, we conducted a series of experiments comparing it against a non-incremental straightforward implementation. The two methods tested shared most of their Java code, implementing an adaptation

#	Quote	Classification of transactions						Total <sup>a</sup>
		Desired	Not desired				Necessary	
		(A)	(B1)	(B2)	(B3)	(B) <sup>b</sup>	(C)	
1	Non-Anonymous Full schedule	65.8%	20.1%	0%	N/A	20.1%	14.1%	100.0%
2	Anonymous Full schedule	62.1%	20.7%	N/A	17.8%	21.8%	16.1%	100.0%
3	Marginal unit	64.8%	17.6%	26.2%	25.2%	31.7%	3.5%	100.0%
4	Standard	57.4%	22.8%	24.8%	23.1%	35.5%	7.1%	100.0%

Table 6: Classification of transactions in AON auctions based on 4-agent simulations. Non-anonymous Marginal unit and Standard quotes were simulated offline and were not available for decision making during the actual experiments.

<sup>a</sup>Union of disjoint categories (A), (B) and (C).

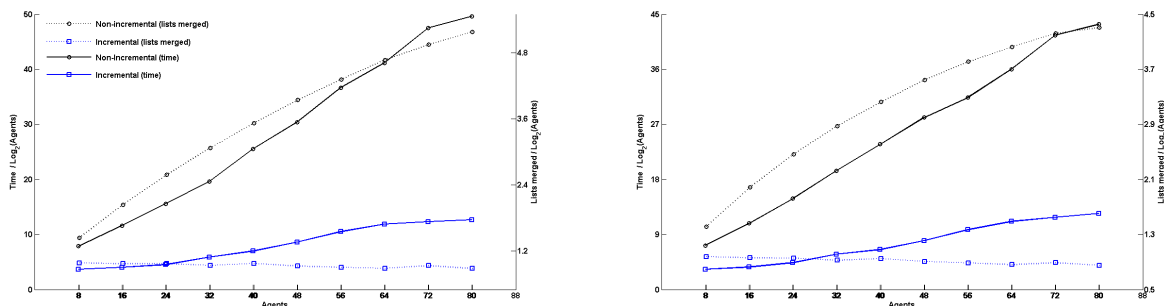
<sup>b</sup>Union of non-exclusive categories (B1), (B2) and (B3).

of DP-with-Lists. Our incremental algorithm arranged lists of undominated states in a binary tree structure, whereas the non-incremental approach arranged these lists sequentially for each bid inserted into the order book, as in a standard MCKP solution. Replacing a bid using the non-incremental approach required recalculating the entire MCKP solution starting from the bid getting replaced.

We based our experiments on the 216 8-player games described above. Each of these games contained four order book histories (i.e., sequence of bid insertions), one per auction. We randomly picked for our analysis 300 out of 864 order book histories available. The chosen histories provided an average of 557.6 bid submissions/replacements. Our experiments measured time spent on bid insertion as well as number of merge operations for each of the chosen histories. We simulated bid insertions for multiples of  $N = 8$  agents, up to 80, by considering three different distributions of insertions. The first was a sequential distribution that reassigned the sequence of bids in the order book to agents as if each agent had submitted bids taking fixed turns, i.e., agent  $1, 2, \dots, N, 1, 2, \dots$ , and so on. The second distribution consisted of randomly picking the sequence agents submitting bids. Finally, we considered the original sequence of bids sent by the actual 8 agents involved

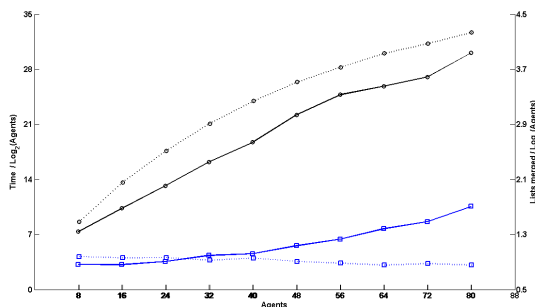
in each game, and simulated more agents by randomly assigning bids by agent  $i$  to either  $i$  or  $i + N$  (for 16 agents), to  $i, i + N$  or  $i + 2N$  (for 24 agents), etc. Our overall approach was very conservative because we kept the total number of bids fixed. In an actual game, more agents would generally involve more bids, making the performance difference between the incremental and non-incremental approaches even larger.

All these experiments were run on Dell OptiPlex GX620 (3.4 GHz) computers under RedHat Linux, and the results are shown in Figure 3. The experiments showed that our incremental algorithm provided substantial computational savings when compared to a non-incremental approach, regardless of the sequence of bid insertions tested. This gain became more relevant when more agents were involved, as we had expected.



(a) Sequential distribution

(b) Random distribution



(c) Original distribution

Figure 3: Performance of the incremental and non-incremental algorithms. Solid lines (left axes) show total time (in seconds) required to insert all the bids in an order book history divided by  $\log_2(\text{agents})$ , averaged over 300 histories. Dotted lines (right axes) show average number of lists of undominated states merged for each bid insertion divided by  $\log_2(\text{agents})$ .

It is important to note that the time taken by our incremental approach, given more agents, increased at a faster than logarithmic rate. This is explained by the fact that our 8 agents (individually and combined) submitted bids smaller than  $C$  (see Section 3.1), and thus more agents created more undominated states to join. If the bound  $C$  were reached, the rate of increase of computation time given more agents would actually be logarithmic.

## 7 Discussion and Future Work

Our study of the AON auction provides evidence for the viability and potential benefits of accounting for indivisibility constraints in market-based allocation, without resorting to fully combinatorial auction designs. Whether one should adopt an AON auction or standard divisible auction depends on the specific setting. Relevant factors include:

1. Expressivity. Divisible bids allow expressing only convex valuations, whereas indivisible ones do not have such a limitation. Would agents with nonconvex valuations refrain from participating in auctions with mandatory divisibility?
2. Undesired trades. If agents with nonconvex valuations do participate in a divisible auction, they risk loss-producing transactions. How much they suffer as a result depends on the degree of nonconvexity.
3. Computation. Our incremental algorithms provide a relatively efficient way to operate AON auctions, which should be fast enough for several practical applications. Standard auctions, however, are still faster to operate, and more predictable since performance is less dependent on the number of units offered for sale.
4. Quote communication. Our experiments showed that the level of detail in price-quote information can play an important role in overall efficiency, and in particular that simple marginal quotes were not enough for AON auctions to improve on the performance of a standard auction. Even though much work remains to be done in this area, it is obvious that the communication burden of the quote used should be evaluated when choosing an auction mechanism over the other.

Further work will refine our comparisons and evaluate additional quoting policies. For example, it would be interesting to measure the potential benefit of providing full schedule quotes in standard auctions, as we have for the AON auction. It would be particularly beneficial to identify intermediate quoting policies for AON auctions that provide much of the benefit of full schedule quotes without the full expense. Understanding this tradeoff remains an important goal. Finally, we are quite interested in exploring the strategic bidding issues posed by indivisibility constraints as well as alternative price quote policies.

## 8 Acknowledgments

We thank Terence Kelly for useful discussions, suggestions and review of our paper, Kevin Lochner for assisting in the development of our AB3D extensions, and the anonymous reviewers for their constructive comments. This work was supported in part by the Engineering Research Center for Reconfigurable Manufacturing Systems at the University of Michigan under Award EEC-9529125 of the National Science Foundation.

## References

- Bassamboo, A., Gupta, M., and Juneja, S. (2001). Efficient winner determination techniques for Internet multi-unit auctions. In *First IFIP Conference on E-Commerce, E-Business, and E-Government*, pages 417 – 430, Zurich. Kluwer, B.V.
- Cheng, J. Q. and Wellman, M. P. (1998). The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24.
- Cramton, P., Shoham, Y., and Steinberg, R., editors (2006). *Combinatorial Auctions*. MIT Press.
- Dudzinski, K. and Walukiewicz, S. (1987). Exact method for the knapsack problem and its generalizations. *European Journal of Operations Research*, 28:3–21.
- Federal Communications Commission (2000). Public Notice DA 00-2404 on auction no. 31. Available at <http://wireless.fcc.gov/auctions/31/releases/da002404.pdf>.

- Horowitz, E. and Sahni, S. (1974). Computing partitions and applications to the knapsack problem. *Journal of the ACM*, 21:277–292.
- Kalagnanam, J. R., Davenport, A. J., and Lee, H. S. (2001). Computational aspects of clearing continuous call double auctions with assignment constraints and indivisible demand. *Electronic Commerce Research*, 1(3).
- Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer Verlag.
- Kelly, T. (2004). Generalized knapsack solvers for multi-unit combinatorial auctions: Analysis and application to computational resource allocation. In *AAMAS-04 Workshop on Agent-Mediated Electronic Commerce (AMEC-VI)*, New York.
- Kothari, A., Parkes, D. C., and Suri, S. (2003). Approximately-strategyproof and tractable multi-unit auctions. In *Fourth ACM Conference on Electronic Commerce*, pages 166–175, San Diego, CA.
- Lochner, K. M. and Wellman, M. P. (2004). Rule-based specification of auction mechanisms. *Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, New York, pages 818–825.
- Lueker, G. S. (1998). Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 29(2):277–305.
- Marchetti-Spaccamela, A. and Vercellis, C. (1995). Stochastic on-line knapsack problems. *Mathematical Programming*, 68(1):73–104.
- McMillan, J. (1994). Selling spectrum rights. *Journal of Economic Perspectives*, 8(3):145–162.
- Miller, R. M. (2002). *Paving Wall Street: Experimental Economics and the Quest for the Perfect Market*. Wiley.
- Nisan, N. (2000). Bidding and allocation in combinatorial auctions. In *Second ACM Conference on Electronic Commerce*, pages 1–12, Minneapolis.
- van Hoesel, S. and Müller, R. (2001). Optimization in electronic markets: Examples in combinatorial auctions. *Netnomics*, 3:23–33.

Wurman, P. R., Walsh, W. E., and Wellman, M. P. (1998). Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24:17–27.

Ygge, F. and Akkermans, H. (1999). Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, 11:301–333.