Revenue Prediction in Budget-constrained Sequential Auctions with Complementarities

Sicco Verwer and Yingqian Zhang

Revised version 19/1/2012

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2011-020-LIS
Publication	January 2012
Number of pages	17
Persistent paper URL	http://hdl.handle.net/1765/25731
Email address corresponding author	yqzhang@ese.eur.nl
Address	Erasmus Research Institute of Management (ERIM) RSM Erasmus University / Erasmus School of Economics Erasmus Universiteit Rotterdam P.O.Box 1738 3000 DR Rotterdam, The Netherlands Phone: + 31 10 408 1182 Fax: + 31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

Bibliographic data and classifications of all the ERIM reports are also available on the ERIM website: www.erim.eur.nl

ERASMUS RESEARCH INSTITUTE OF MANAGEMENT

REPORT SERIES

RESEARCH IN MANAGEMENT

ABSTRACT AND KEYWORDS	
Abstract	When multiple items are auctioned sequentially, the ordering of auctions plays an important role in the total revenue collected by the auctioneer. This is true especially with budget constrained bidders and the presence of complementarities among items. In such sequential auction settings, it is difficult to develop efficient algorithms for finding an optimal sequence of items that optimizes the revenue of the auctioneer. However, when historical data are available, it is possible to learn a model in order to predict the outcome of a given sequence. In this work, we show how to construct such a model, and provide methods that finds a good sequence for a new set of items given the learned model. We develop an auction simulator and design several experiment settings to test the performance of the proposed methods.
Free Keywords	sequential auctions, revenue maximization, learning, experimentation
Availability	
Classifications	

Revenue prediction in budget-constrained sequential auctions with complementarities

Sicco Verwer[1] and Yingqian Zhang[2]

[1] Faculty of Science, University of Nijmegen, The Netherlands
S.Verwer@cs.ru.nl
[2] Erasmus School of Economics, The Netherlands
yqzhang@ese.eur.nl

Abstract. When multiple items are auctioned sequentially, the ordering of auctions plays an important role in the total revenue collected by the auctioneer. This is true especially with budget constrained bidders and the presence of complementarities among items. In such sequential auction settings, it is difficult to develop efficient algorithms for finding an optimal sequence of items that optimizes the revenue of the auctioneer. However, when historical data are available, it is possible to learn a model in order to predict the outcome of a given sequence. In this work, we show how to construct such a model, and provide methods that finds a good sequence for a new set of items given the learned model. We develop an auction simulator and design several experiment settings to test the performance of the proposed methods.

Keywords: Sequential auctions, revenue maximization, learning, experimentation

1 Introduction

Auctions are becoming increasingly popular for allocating resources or items in business-to-business and business-to-customer markets. Various auction formats exist, such as combinatorial auction where agents submit bids with item bundles [19], and sequential auctions where items are sold consecutively to agents [3]. Often sequential auctions are adopted in practice, especially when the number of items for sale is large (cf. the Dutch flower auction [21]), and when the buyers enter and leave the auction dynamically (cf. online auctions [15]).

In many situations, agents desire items with complementarities. For example, in the resource allocation auctions [3], obtaining one resource (i.e., a truck) without another resource (i.e., fuel) makes the first resource worthless. In addition, bidders often have budget or capacity constraints, as often seen in industrial procurement [11], or in FCC spectrum auctions, where it is found to be realistic to assume that all personal communicating services firms are budget constrained [6].

More and more auctions nowadays utilize information technology. For example, the Aalsmeer flower auction [10] is converting the traditional flower auction,

where buyers need to be present in the auction room, to a modern remote market place, where buyers can remotely place bids directly from all over the world. Increasingly popular online auctions are conducted using computers. The use of information technology makes it possible to automatically store detailed information about bidding data. This auction information provides the auctioneer the possibility to analyze the past auctions in order to gain insights into bidders' behaviors and their preferences over the auctioned resources or items. Such valuable information could support the auctioneer to manage future auctions.

In this paper we study how the earlier auction information can be used for the auctioneer to solve one managerial problem in sequential auctions—the auctioning sequence for the given set of items. Previous research has shown the revenue is heavily dependent on the ordering of items especially when bidders have budget constraints [16, 9] or when they have preference over bundles of items [20]. We use the following two examples to intuitively demonstrate the importance of ordering on the expected revenue even when bidders are truth telling. Our first example considers budget constrained bidders.

Example 1. Two agents A_1 and A_2 take part in a sequential auction of resources. For sale are resource r_1 and resource r_2. The value of these resources for the agents are given as follows: $v_{A_1}(r_1) = 5, v_{A_1}(r_2) = 5, v_{A_2}(r_2) = 4$. In addition, both agent A_1 and A_2 have a budget limit of 5. Consider one situation where the auctioneer orders the items first r_1 and then r_2. In this case A_1 will get r_1 with price 5, and then A_2 will win r_2 as A_1 has no money left. The total revenue is 9. However, if r_2 is auctioned before r_1, A_1 will win r_2 with price 5, and r_1 will not be sold since A_2 is not interested in it. In this situation, the total revenue collected becomes 5.

In the second example, agents desire items with complementarities.

Example 2. Suppose given r_1 and r_2, agent A_2 only desires r_2 with value 5. A_1 has complementary items, i.e., A_1 values r_1 and r_2 with 1 and 1 respectively if he wins only r_1 or r_2 at the end of the auction, however, if he wins both resources r_1 and r_2, the value $v(r_1 \cup r_2)$ goes up to 10. It is quite obvious that in this case, the auctioneer should sell r_1 first as it will end up with revenue 10, in contrast to 6 if r_2 is sold first.

Much of the existing work that studies optimal ordering in auctions focuses on theoretical analysis on bidders' strategy [8, 20] and conditions when the optimal ordering exits [9, 16, 20]. However, it is difficult to apply these results to actual auctions as they rely on strong assumptions which rarely hold in practice. In this paper, we develop a novel method for finding revenue-maximizing orderings that can generalize to real-world auctions.

Unlike existing work, our method is based on techniques from machine learning. It uses historical auction data in order to quickly learn which orderings have high expected revenue. Since many auctions, such as online auctions and the Dutch flower auctions are held regularly, where the same types of items are auctioned repeatedly, such data is readily available.

We briefly describe the auction setting as follows. There are a set of private value items for sale in a sequential auction. There are a set of budget-constrained bidders, with unknown valuation functions. Bidders may desire more than one item, and some of them may prefer item bundles. For each set of items, a sequential auction is held. This sequential auction is repeated over time, with probably different items. When we evaluate our method by experiments, we assume the auctions are first-price, that is, the highest bidder will receive the item for the price bid.

Our main contribution is providing a method that first transforms this information into a data set, then learns models (in our case regression trees) for predicting the revenue of orderings (Section 3.2), and finally uses a best-first search algorithm in order to find a good ordering for a new set of items (Section 3.3). Since we make few assumptions regarding the auction setting, our method has a high potential to be used in practice. We implement an auction simulator, develop three types of agents (in terms of their bidding strategies), and design several (simple and complex) experiment settings, in order to test the performance of the proposed learning method (Section 4). We compare this performance with a random ordering strategy, a fixed ordering strategy, and a simulated lower bound on the optimal ordering.

We now start with describing some existing work.

2 Related work

A few existing papers investigate the problem of how to maximize the seller's revenue by means of changing the ordering of items in sequential auctions. Milgrom and Weber [14] look at a setting where items are homogeneous and bidders desire only one item. Elmaghraby [9] studies procurement auction where a buyer outsources two heterogeneous jobs by sequential second-price auctions. He shows that specific sequences lower procurement costs and identifies a class of suppliers' cost functions where efficient orderings exist.

Pitchik [16] points out that in the presence of budget constraints, a sealed-bid sequential auction with two bidders and two goods may have multiple symmetric equilibrium bidding functions, and the ordering of sale affects the expected revenue. If the bidder who wins the first good has a higher income than the other one, the expected revenue is maximized.

Subramaniam and Venkatesh [20] investigate the optimal auctioning strategy of a revenue-maximizing seller, who auctions two items, which could be complements or substitutes. They show that when the items are different in value, the higher valued item (among the two) should be auctioned first in order to increase the seller's revenue. This conclusion is drawn based on a large number of computer simulations with the assumption that bidders' valuations are uniformly distributed. A similar revenue-maximizing strategy is proposed by Benoit and Krishna [1] in a complete information common value auction setting. The authors conclude that in such a setting, when selling two items to budget constrained bidders, it is always better to sell the more valued item first. However,

this strategy does not optimize the revenue anymore when more than two items are to be auctioned.

Elkind and Fatima [8] study how to maximize revenue in sequential auctions with second-price sealed-bid rules where homogenous bidders have unit demand. In this setting, they analyze the equilibrium bids, and develop an efficient algorithm that finds an optimal agenda (i.e., ordering). However, the problem of selecting the optimal agenda becomes NP-complete if bidders' valuations for the same item can differ, even if each bidder's value for each item is known, and all bidders bid truthfully in each auction.

Empirical research has also been conducted to test the theoretical findings. Grether et al. [13] report on a field experiment that tests the ordering strategies of a seller in sequential, ascending automobile auctions. Several ordering strategies such as high values first or low values first have been tested. They conclude that the worst performing ordering in terms of revenue is for the seller to auction vehicles from highest to lowest values. This result contradicts the conclusion drawn from the theory of single time and single seller auctions. Raviv [18] uses an example to demonstrate that there are cases where the ordering of the auction does not affect revenue using a second price seal-bid auction. In his example, there are two heterogeneous items for sale among three risk neutral bidders. Raviv shows that when the ordering is randomized, the expected selling price stays the same, no matter which one of the items is sold first.

In this paper, we adopt numerical experiments to investigate the influence of ordering in sequential auctions instead of performing pure theoretical analysis. However, unlike the work of [18] which tests *pre-defined* orderings, we build a learning model that learns good orderings from the historical auctions.

We are not the first who consider learning from the previous auctions. Boutilier et al. [3] propose a learning model for *bidders* to update their bidding policies in sequential auctions for resources with complementarities. The bidding strategies are computed based on the estimated distribution over prices, that is modeled by dynamic programming. They show the emergence of interesting bidding behaviors by simulating simple resource allocation problems with 5 agents. Goes et al. [12] present an empirical study of real sequential online auctions. They analyze the data from an online auction retailer, and show that bidders learn and update their willingness to pay in repeated auctions of the same item. In [15], the authors show the benefits of using earlier auction data for the management of sequential, multi-unit auctions, where the seller needs to split its entire inventory into sequential auctions of smaller lots in order to increase its profit. In their work, an auction feedback mechanism is developed based on a Bayesian model, and it is used to update the auctioneer's beliefs about the bidders' valuation distribution. The simulations were conducted with truth-telling agents. The results show the feedback mechanism outperforms the naive fixed lot size policy. Another interesting finding is that when there are many bidders, the relative benefit of using the learning algorithm seems less important.

3 Learning good orderings

We aim to find a good ordering (with a high expected revenue) for a set of items in an auction. The decision whether an ordering is good has to be determined from historical data of the same auction, but possibly with different agents and different (numbers of) items. A possible approach to tackle this problem would be to learn the utility functions of the agents. We could then use these estimated utility functions to try to find an ordering with high revenue. Although this approach sounds sensible, we believe that it will fail in practice due to two complexity issues:

1. Learning utility functions in combinatorial domains is hard [19], not to mention learning one for every agent.
2. Given correctly learned utility functions, finding a good ordering in the sequential auction is still NP-complete [8].

In our approach, we try to circumvent these difficulties by making modeling assumptions that simplify the learning problem, and by selecting learning targets that makes finding a good ordering easier.

3.1 Auction setting and modeling assumptions

We assume there is a finite set of agents A. Let $R = \{r_1, \ldots, r_l\}$ denote the collection of the item types, and the quantity of each item type can be more than 1. When it is clear from the context, we will slightly abuse the notation and use $S = \{r_1, r_2 \ldots, r_1, \ldots\}$ to denote the set of all available items. Each bidding agent A_i has a valuation for each type of item or each bundle of different item types $v_i : R \to \mathbb{R}^+$. We say that items $r_1, r_2 \in R$ are complementary for agent A_i if $v_i(r_1) + v_i(r_2) < v_i(r_1 \cup r_2)$. In addition, each agent $A_i \in A$ has a budget constraint B_i on purchasing items.

In one round of auction, a set of items S with type set $R' \subseteq R$ will be auctioned sequentially with an order that is announced before the auction starts. For example, given item types $r_1 \in R$ and $r_2 \in R$ with quantities of 1 and 2 respectively, there are three possible orderings of items: (r_1, r_2, r_2), (r_2, r_1, r_2), and (r_2, r_2, r_1). The bidder agents are drawn from the pool of A. We assume that the auction is repeated over time, and each auction sells possibly different items S, with possibly different set of agents $A' \subseteq A$.

We assume that auctions are conducted using Dutch auctions, where the auctioneer starts with a high asking price which is lowered until some agent is willing to accept the auctioneer's price, and this agent will be awarded the item for the last announced price.

At the end of each sequential auction, we have the following information at our disposal: (1) the ordering of auctioned items; and (2) the revenue of each sold item. Our goal of the sequential auction design is that given a set of items $r_1, \ldots, r_i$ for sale, deciding the ordering of items such that the revenue collected is maximized. We aim to find such a good ordering by learning from the previous

auctions. In order to simplify the learning problem, we make the following two modeling assumptions.

Assumption 1 ***(Bidder independence)*** *In every round of sequential auctions, the set of participating bidders and their valuation functions are similar.*

This assumption simplifies the problem of learning a good ordering. Instead of learning the individual utility functions of agents, we can treat the agent population as a single entity for which we need to find a single global utility function. Such an approach will fail if the valuations of the agents are radically different in every auction. However we consider this assumption sensible in many auction settings. For instance, in the Dutch flower auction there can be different participants every day, but it never occurs that one day people are only interested in roses and the next day they only want tulips. Although the different participants are interested in different item types, the interests of the group of participants remain stable.

The first assumption effectively reduces the difficulty of the learning problem to that of a standard machine learning setting: learn a single model from orderings and their rewards for predicting the expected reward for a given new input ordering. Because we can now generalize over all bids instead of only the bids of a single agent, the first assumption significantly increases the amount of available data. It is however still not straightforward how to apply machine learning techniques, since there are many possible orderings and modeling the expected revenue of every single one of them (using for instance a language model) will require a lot of data. Therefore, we make another important assumption that generalizes over different orderings in a sensible way:

Assumption 2 ***(Ordering independence)*** *The expected revenue for an item depends on which items were sold before and which items are still to be sold, but not on their ordering.*

This assumption represents our intuition on the amounts that agents will bid. Suppose an agent A needs to buy an item r_i, then the amount A is willing to pay for r_i depends on whether this is the last r_i being sold in the auction, or whether there are more to come. In another example, suppose A wants to buy r_i and r_j, then the amount A bids for r_i depends on whether A already obtained r_j, i.e., whether r_j was auctioned before r_i. In these examples, the exact ordering of items before and after r_i does not matter much for A's valuation of r_i. We realize this assumption slightly overgeneralizes the auction setting since there is a possibility that the ordering determines whether A could have obtained r_j before r_i. However, since it greatly reduces the complexity of our models, and thus also the amount of data required for learning, we gladly disregard this possibility in our model.

3.2 Representing orderings

Given our two modeling assumptions, we need to find a suitable way to model the expected revenues of such orderings. An ordering can be thought of as a

sequence of items. However, to the best of our knowledge none of the existing sequence models fits our auction setting. Language models such as deterministic automata [7] are too powerful since they do not require our second assumption. Short sequence models such as hidden Markov models [2] do not model the dependence on items sold a long time (more than the sliding window length) before. What comes closest to our auction setting are models such as Markov decision processes (MDPs) [17]. These directly model the expected revenue per item, and we can build a state space that fits with our assumptions. However, none of the models we know is capable of producing orderings of a given finite set of items. This given set of items determines both the available actions (which item to auction) and the goal state (when no items remain). Since this set is rarely the same in different auctions, representing and learning the revenues of auctions in an MDP is difficult. It is possible that with a suitable factored representation of the states and/or function approximation [17] of the rewards, we could represent our auctioning problem as an MDP. In fact, our method can be seen as a pragmatic version of the MDP approach.

Instead of representing our problem with a sequence model such as an MDP, we view the prediction of the revenue of an auction as a regression problem. Like an MDP, we split this problem into the subproblems of predicting the revenue of the auctioned items. We then sum these up to obtain the overall objective function:

$$V(r_1 \dots r_n) = \sum_{1 \leq k \leq n} R(r_k, \{r_j \mid j < k\}, \{r_l \mid k < l\})$$

where $R(r_k, J, L)$ is a regression function that determines the expected revenue of r_k given that J was auctioned before and L will be auctioned afterwards. These objective and regression functions match with both of our modeling assumptions: it is independent of the participants and does not take the item ordering into account. In addition, in contrast to the objective function typically used in MDPs, ours is not recursive. This is possible because the set of items (available actions in an MDP) is given beforehand. We use regression trees [4] as a regression function and train it using features based on the items auctioned before and after the current item r_k. Currently, we provide the following features:

Feature 1 For every item type r_T, the amount of r_T items already auctioned.
Feature 2 For every item type r_T, the amount of r_T items still to be auctioned.
Feature 3 For every pair of item types r_T and $r_{T'}$, the difference between the amount of r_T and $r_{T'}$ items already auctioned.

These features model the influence of utility functions with complementarities. For instance, if many agents desire both r_T and $r_{T'}$ types, and if the amount of $r_{T'}$ items already auctioned is large when auctioning an r_T item, then we expect the revenue for this r_T item to be high. Although the third feature can be determined using the first, it is added for convenience of learning a decision tree which requires many nodes to represent these values. The influence of budget constraints are modeled only indirectly by these three features: once the amount of r_T items auctioned reaches a certain (to be learned) bound, we can expect all

agents that only want r_T items to be out of budget. It could be better to model this influence more directly by adding the following additional feature:

Feature 4 For every item type r_T, the amount of revenue obtained from auctioning r_T items.

This feature models the influence of budget constraints much more directly. However, since the auctioneer has to provide the ordering of an entire auction before the auction starts, we can only use estimates on these amounts when determining this order. For this we can use the predictions of the regression model on the previously auctioned items. Although adding this feature could improve the performance of the regression model, it also introduces the risk of accumulating errors. Suppose the regression model incorrectly predicts the revenue of one of the first items in the auction, then this value will be incorrectly used in the fourth feature in later predictions. Hence, an incorrect prediction potentially leads to a cascade of incorrect predictions. In our experiments, we use regression trees learned with and without using this feature for the purpose of comparison.

Below we give an example of how an ordering and its obtained revenues is transformed into a data set using these 4 types of features.

Table 1. The data set created from the auction data in Example 3.

Type	Revenue	A before (Feature 1)	A after (Feature 2)	B before (Feature 1)	B after (Feature 2)	A¿B (Feature 3)	Sum A (Feature 4)	Sum B (Feature 4)
A	10	0	3	0	4	0	0	0
A	8	1	2	0	4	1	10	0
B	4	2	2	0	3	2	18	0
A	8	2	1	1	3	1	18	4
B	6	3	1	1	2	2	26	4
B	3	3	1	2	1	1	26	10
B	3	3	1	3	0	0	26	13
A	14	3	0	4	0	-1	26	16

Example 3. Suppose that a set of item A and a set of item B are being auctioned using the ordering $AABABBBA$, and that the revenues of the items obtained in the auction are in order: 10,8,4,8,6,3,3,14. For every item-revenue pair in this auction we create one row in the data set with values for all the features described above. The data set is shown in Table 1.

A data set obtained in this way can be given as input to any standard regression method from machine learning. In our case, we learn a regression tree for every item type using recursive partitioning techniques [5]. The result is a set of predictive models for the expected revenue of items, and by summing these revenues we obtain the expected revenue of an auction.

Algorithm 1 Computing a good ordering

Require: A set of items S, historical data on orderings and their revenues D, a maximum number of iterations m
Ensure: Returned is a good (high expected revenue) ordering
  Transform D into a data set
  for every item type r_T **do**
    Learn a regression model from D for predicting the revenue of item type r_T
  end for
  Initialize a hashtable H and a priority queue Q
  Add the empty data row to Q
  while Q is not empty and the size of H is less than m **do**
    Pop the row of features F with highest value v from Q
    if H does not contain F with a value $\geq v$ **then**
      Add F with value v to H
      Let L be the set of remaining items in F
      for every item type r_T of items in L **do**
        Let i_k be an item of Type r_T in L
        Let L' be a random ordering of $L - i_k$
        Use the learned models to evaluate the value v' of auctioning the ordering $i_k L'$ after F
        Create new features F' for auctioning i_k after F
        Add F' to Q with value $v + v'$
      end for
    end if
  end while
  return The highest evaluated ordering

3.3 Computing an ordering

Given the predictive model for the expected revenue per item, it is not yet straightforward to compute a good ordering. For a given ordering, we can generate a data set, predict the individual revenues of items using the regression model, and sum these up to obtain the revenue of the ordering. However, testing all possible orderings and choosing the one with the highest expected revenue is impossible due to the huge amount of possible orderings. The traditional method to overcome this computational blowup in MDPs is to use a dynamic programming method. Although this lessens the computational load by combining the different paths that lead to the same sets of auctioned items, the search space is still too large and waiting for a solution will take too long (especially if the fourth feature is being used). Instead, we therefore employ a best-first search strategy that can be terminated at anytime in order to return the best found solution so far. We show how this best-first search strategy works in Algorithm 1.

The algorithm uses a hashtable and a priority queue. The hashtable is used to exclude the possibility of visiting the same nodes twice if the obtained revenue is less than before (just like a dynamic programming method). The priority queue provides promising candidate nodes for the best-first strategy. By computing random orderings of the remaining items, the learned models can evaluate

complete orderings of all items. The best one found is stored and returned if the algorithm is terminated. Unfortunately, this does not result in an admissible heuristic for an A* search procedure. Hence, even if the algorithm pops a solution from the queue, this is not necessarily optimal. In our experience, using random orderings of the remaining items in this heuristic provides a good spread over the search space. Although some nodes can be 'unlucky' and obtain a bad ordering of the remaining items, there are always multiple ways to reach nodes in the search space and it is very unlikely that all possibilities will be 'unlucky'.

4 Experiments

In order to test our method in a bit more realistic auction settings, we developed an auction simulator and created three types of agents who have different bidding strategies. We then use the simulator to generate different auctions to see how our method performs in various auction settings.

4.1 Auction simulator

We created an auction simulator of a Dutch auction (i.e., first-price descending auction). Since the outcome of an auction is influenced by the bidding strategies of the agents, we simulate three different types of agents in auctions: myopic, smart, and simulator agents.

- The *myopic* agents bid as soon as the asking price reaches their true value. This true value is determined by partitioning the set of items that are in the agent's possession over the agents types and bundles such that the sum of their values is maximized. This difference in this value before and after adding the current item to the agent's possessions is the agent's true value. If multiple agents have the same true value, one of these is selected as winner uniformly at random.
- The *smart* agents know all the valuation functions of all agents and bid as soon as the asking price is less or equal to their true value and at most one greater than the true value of all other agents.
- The *simulator* agents have access to the auction simulator. They bid what the smart agents bid, only if a run of the simulator results in a higher utility (i.e., the total value of obtained items plus the remaining budget) when placing this bid than when not. In addition, the simulator agents will overbid their true value if this gives them higher utility at the end of the auction, due to for example being rewarded the complementary items.

Intuitively, the myopic agents are truth-telling agents. The smart agents represent agents (for example those in [3]) that have succeeded in learning the value that will be placed for the current item. The simulator agents represent agents that have truly learned to optimize their utility. They are able to determine for every bid value exactly what their final utility will be. Of course, it is very unlikely that agents will be able to predict these values perfectly in practice.

However, since such a prediction capability is the goal of learning, experiments with these agents will show how our method performs in the presence of good learning agents.

Using this simulator we generated data sets (orderings of item-revenue pairs) and tested whether our method was able to learn good orderings in the presence of these three types of agents. We now explain how we generated a small auction that consists of 8 agents bidding for 4 item types.

Generating small auctions We initialize the auction simulator using a given set of items R. We assign every type in R an average value of 10, 15, 20, or 25. Every possible value is assigned exactly once to a type. Every type has a random popularity (probability of being desired by an agent). In addition, every pair of types is assigned a pair probability of being desired as complementary items by an agent. This probability is either 0.0, 0.5, or 1.0 for every pair, decided at random. Thus in some auction settings it is possible that the expensive items are popular and always complementary to less expensive unpopular items, while in other settings, cheap items are popular but never complementary. In every auction, there are 2 to 5 items of every type being auctioned, decided at random.

Every agent is generated at random using the same scheme. They receive a random budget between 30 and 60. It selects at most 3 desired item types. Popular types have a higher probability of being selected. Every desired item type is assigned a value drawn uniformly at random with its average price as mean and a range of 11. At most two pairs of desired types are combined into a complementary pair based on the pair probabilities. These get assigned a value equal to the sum of the values of the individual types multiplied by 2. If necessary, the budget of the agent is increased to match this value.

Using this generating scheme, we obtain a diverse set of agents, but with similar preferences. The following is for example a set of agents generated for the experiments with four item types A, B, C, D:

```
budget=58 v(C)=12 v(D)=14 v(B)=17 v(CD)=52 v(BC)=58
budget=78 v(B)=16 v(D)=15 v(A)=23 v(AB)=78 v(BD)=62
budget=64 v(B)=20 v(C)=12 v(BC)=64
budget=70 v(D)=11 v(B)=17 v(C)=18 v(BD)=56 v(BC)=70
budget=38 v(D)=7 v(C)=12 v(CD)=38
budget=66 v(B)=19 v(C)=14 v(BC)=66
budget=42 v(C)=17
budget=66 v(A)=23 v(D)=10 v(AD)=66
```

In this list, every row is an agent where its budget and valuations are indicated. Although the set of agents is small and their preferences are similar, it is not immediately clear what the outcome of an auction would be, even for myopic agents. For example, the following three randomly generated auctions demonstrate the variance of winning bids for different item types when the bidders are myopic.

```
(D,15)(B,47)(D,15)(C,18)(C,18)(C,18)(A,23)(D,43)(B,20)
(A,0)(A,0)(D,14)(D,14)
(D,15)(A,23)(A,23)(A,23)(A,23)(D,15)(A,0)(B,20)(C,44)
```

```
(C,18)(B,52)(D,14)
(A,23)(D,43)(B,20)(C,44)(D,15)(B,47)(D,15)(A,0)(B,19)
(C,47)(C,18)(C,18)(D,14)(B,17)
```

Generating complex auctions We also used this generator for generating a setting with 30 smart agents, of which 25 participate in a round of sequential auction and bid for 8 types of items. The main change of the generator for this complex setting is an increase of the maximum amount of generated items from 5 to 10. In addition, every average price is now assigned to every item exactly twice and we also bound the number of popular items. Overall, this results in larger and more complex auctions.

Our main goal is to learn models that uncover the hidden structure underlying the winning bids in these auctions. We then use these models in order to find a high revenue ordering for a new set of items. In the next section, we describe the methods we use to test the quality of this ordering.

4.2 Experimental setup

We evaluate the performance of our method using the auction simulator. We first generate a set of agents, and run simulations of 250 random orderings of randomly selected items (2 to 5 of every type in the small auction). These 250 orderings and their obtained revenues are transformed to a data set as shown in Section 3.2 and provided to the regression tree learner. We use an implementation of regression trees from the Orange data mining toolbox[3] in Python. The resulting models are used to provide an ordering for a new set of randomly generated items using Algorithm 1 with a maximum number of 1000 iterations. We then run 10 simulations of this ordering and average the resulting revenues. The last two steps are performed for 25 different random sets of items. The resulting average revenues are again averaged. We use this average revenue of different sets of items for the learned ordering method as a performance indicator. This whole procedure is repeated 5 times in order to test for different sets of agents. For the complex setting, it is repeated 10 times.

We test two implementations of our method: one with Feature 4 (See Section 3.2), and one without. Moreover, we compare them with two ordering strategies: (i) a random ordering, and (ii) a fixed ordering suggested by the literature [20, 1]: auction the most valuable item first. In addition, we include a lower bound on the average revenue of an optimal ordering. This is computed by running simulations of 250 random orderings and selecting the one with the highest revenue.

4.3 Results

In this section, we first describe the results obtained on the small auction settings with 8 myopic, smart, and simulator agents. Afterwards, we show the results

[3] `http://orange.biolab.si/doc/`

obtained on the larger complex setting of 30 smart agents, of which 25 participate in every auction.

Small auction setting Table 2 shows the values of the performance indicator (mean revenue for a random set of items) obtained using each of the random (*random*) and fixed orderings (*most valuable first*), the two implementations of our method (*best first with sum* and *best first*), and a lower bound on an optimal ordering (*best auction found*).

Table 2. The performance of ordering methods with 8 myopic agents.

ordering strategy	1	2	3	4	5	sum
random	192	252	216	246	230	1136
most valuable first	176	214	184	251	223	1048
best first	204	247	225	245	229	1150
best first with sum	196	243	232	249	231	1151
best auction found	208	284	235	255	232	1214

There are 5 sets of values, one for each repetition of the experiment. These values are summed up to obtain the overall performance of all methods. Clearly, the lower bound method performs best, followed by our methods, the random ordering, and finally the fixed ordering.

The result that selling the most valuable item first gives the worst performance agrees with the empirical work of [13], although it contradicts the existing theoretical findings [20, 1] where much simpler settings are considered.

Surprisingly the differences between our methods and the random ordering strategy is very small and not significant. In fact, the mean values for a single repetition are significantly different (typically) only if their difference is greater than 25. This holds for neither the difference between our methods and the random strategy, nor the difference between our methods and the lower bound of the optimal ordering (i.e., best auction found). Our methods do significantly outperform the most valuable first strategy.

Table 3. The performance of ordering methods with 8 smart agents.

ordering strategy	1	2	3	4	5	sum
random	260	237	231	158	262	1148
most valuable first	259	235	223	180	253	1150
best first	269	242	235	172	264	1182
best first with sum	269	228	232	176	266	1171
best auction found	278	259	237	183	279	1236

We conducted another set of auctions with 8 smart agents and the results are shown in Table 3. The main conclusion that we draw from these results is that ordering does matter: the best ordering found in 250 random orderings does outperform the other methods. Our methods perform better than the random strategy and the fixed ordering strategy, but not significantly.

Table 4. The performance of ordering methods with 8 simulator agents.

ordering strategy	1	2	3	4	5	sum
random	225	183	203	225	192	1028
most valuable first	212	182	204	225	180	1003
best first	240	181	209	234	193	1057
best first with sum	237	181	211	234	194	1057
best auction found	239	191	216	323	197	1166

Finally, with the simulator agents (Table 4), our methods outperform the most valuable first method significantly. However the performance increase between our methods and the random one is not significant. The lower bound does provide a significant difference compared to all the other methods, showing that even in the presence of very intelligent agents, ordering the items in a good way can really benefit the collected revenue.

Complex auction setting We now show the results obtained on the larger complex setting of 30 smart agents, of which 25 participate in every auction (Table 5). There are 8 different types of items and 2 to 10 of them are auctioned in every sequential auction. The results confirm again that ordering items according to their values performs worst in our auction setting and that the regression trees are able to produce better orderings than random orderings. In fact, the difference between the best first method with the sum information performs significantly better than random. This is surprising since we expected that this method would be troubled by the possibility of cascading errors (see Section 3), especially when many items are being auctioned.

Table 5. The performance of ordering methods with 25 smart agents.

ordering strategy	1	2	3	4	5	6	7	8	9	10	sum
random	1068	1121	1037	1206	1143	876	991	849	822	1138	10251
most valuable first	959	1075	1048	1154	1148	879	1073	896	907	1059	10198
best first	1062	1149	1072	1198	1107	885	1019	882	836	1161	10371
best first with sum	1115	1156	1062	1219	1137	872	1029	901	815	1125	10431
best auction found	1115	1163	1097	1263	1163	937	1050	918	862	1199	10767

Our preliminary results are encouraging as they show that our method is able to learn a good ordering for a complex auction setting from a small amount of examples (250 historical orderings along with their revenues). In addition, surprisingly they show that a random ordering strategy also performs well in complex auction settings. This is counter-intuitive. Further investigations (theoretical and empirical analysis) are required to explain this result.

5 Conclusions

The ordering of auctions plays an important role in the total revenue collected by the auctioneer in sequential auctions. This is true especially with budget constrained bidders and the presence of complementarities among items which occur often in real-world auctions.

In this paper, we propose a novel method to find a good ordering. We show how historical auction data can be transformed into a data set for learning models (in our case regression trees) that predict the expected revenue of orderings for new auctions. We then provide best-first search based methods that find a good ordering for a new set of items from the learned models. We develop an auction simulator to generate simple and complex auction settings. Three types of agents participate in the auctions: myopic, smart, and simulator agents. We compare the performance of the proposed methods with different ordering strategies.

The experimental results are interesting: our method is able to learn the influences of budget limits and complementary items on the total revenue, and to learn the good ordering with high expected revenue. However, surprisingly the random ordering strategy also works well in most test cases. This contradicts intuition drawn from the existing theoretical results [9]. We are currently investigating when the benefit of using the learning method is important or unimportant in sequential auction setting.

Compared to existing work on revenue maximization in auctions, our method requires much fewer assumptions regarding the auction setting, the types of bidders, and their valuations. As long as there exists historical data, and the bidders do not change radically, our method can be applied. We are therefore very interested to test our method on real data in the near future.

Our assumptions of ordering and bidder independence make it possible to use standard machine learning models in order to quickly learn the hidden structure of a good ordering. In the future, we would like to investigate whether these assumptions can be relaxed and what the effect will be on the learning performance. In addition, we are investigating whether our problem can be modeled and solved by an MDP. This is not straightforward due to the given set of items in an auction, which will rarely be the same in different auctions, and the fact that an ordering has to be provided before the auction starts. We also plan to test different regression methods.

References

1. Jean-Pierre Benoit and Vijay Krishna. Multiple-object auctions with budget constrained bidders. *Review of Economic Studies*, 68(1):155–79, January 2001.
2. Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
3. Craig Boutilier, Moises Goldszmidt, and Bikash Sabata. Sequential auctions for the allocation of resources with complementarities. In *Proceedings of the 16th international joint conference on Artifical intelligence - Volume 1*, pages 527–534, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
4. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.
5. L. Breiman, JH Friedman, R. Olshen, and CJ Stone. *Classification and regression trees*. Wadsworth International Group, 1984.
6. Peter Cramton. Money out of thin air: The nationwide narrowband pcs auction. *Journal of Economics & Management Strategy*, 4(2):267–343, 1995.
7. Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA, 2010.
8. Edith Elkind and Shaheen Fatima. Maximizing revenue in sequential auctions. In *Proceedings of the 3rd international conference on Internet and network economics*, WINE'07, pages 491–502, Berlin, Heidelberg, 2007. Springer-Verlag.
9. Wedad Elmaghraby. The importance of ordering in sequential auctions. *Manage. Sci.*, 49:673–682, May 2003.
10. FloraHolland. http://www.floraholland.com/en/pages/default.aspx.
11. Jérémie Gallien and Lawrence M. Wein. A smart market for industrial procurement with capacity constraints. *Manage. Sci.*, 51:76–91, January 2005.
12. Paulo B. Goes, Gilbert G. Karuga, and Arvind K. Tripathi. Understanding willingness-to-pay formation of repeat bidders in sequential online auctions. *Info. Sys. Research*, 21:907–924, December 2010.
13. David M. Grether and Charles R. Plott. Sequencing strategies in large, competitive, ascending price automobile auctions: An experimental examination. *Journal of Economic Behavior & Organization*, 71(2):75–88, August 2009.
14. Paul R Milgrom and Robert J Weber. A theory of auctions and competitive bidding. *Econometrica*, 50(5):1089–1122, September 1982.
15. Edieal J. Pinker, Abraham Seidmann, and Yaniv Vakrat. Using bid data for the management of sequential, multi-unit, online auctions with uniformly distributed bidder valuations. *European Journal of Operational Research*, 202(2):574–583, 2010.
16. Carolyn Pitchik. Budget-constrained sequential auctions with incomplete information. *Games and Economic Behavior*, 66(2):928–949, July 2009.
17. Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.
18. Yaron Raviv. New evidence on price anomalies in sequential auctions: Used cars in new jersey. *Journal of Business & Economic Statistics*, 24:301–312, July 2006.
19. Tuomas Sandholm and Craig Boutilier. Preference elicitation in combinatorial auctions. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 2. MIT Press, 2006.
20. Ramanathan Subramaniam and R. Venkatesh. Optimal bundling strategies in multiobject auctions of complements or substitutes. *Marketing Science*, 28:264–273, March 2009.

21. Eric van Heck and Pieter M. A. Ribbers. Experiences with electronic auctions in the dutch flower industry. *Electronic Markets*, 7(4), 1997.

Publications in the Report Series Research* in Management

ERIM Research Program: "Business Processes, Logistics and Information Systems"

2011

Sequencing Heuristics for Storing and Retrieving Unit Loads in 3D Compact Automated Warehousing Systems
Yugang Yu and René B.M. De Koster
ERS-2011-003-LIS
http://hdl.handle.net/1765/22722

A Local Search Algorithm for Clustering in Software as a Service Networks
Jelmer P. van der Gaast, Cornelius A. Rietveld, Adriana F. Gabor, and Yingqian Zhang
ERS-2011-004-LIS
http://hdl.handle.net/1765/22723

Implementing Standardization Education at the National Level
Henk J. de Vries
ERS-2011-007-LIS
http://hdl.handle.net/1765/22812

The Power Trading Agent Competition
Wolfgang Ketter, John Collins, Prashant Reddy, and Christoph Flath
ERS-2011-011-LIS
http://hdl.handle.net/1765/23263

Real-time Tactical and Strategic Sales Management for Intelligent Agents Guided By Economic Regimes
Wolfgang Ketter, John Collins, Maria Gini, Alok Gupta , and Paul Schrater
ERS-2011-012-LIS
http://hdl.handle.net/1765/23339

Knowledge Sharing in Non-Knowledge Intensive Organizations: When Social Networks do not Matter?
Joey van der Capellen, Otto Koppius, and Koen Dittrich
ERS-2011-013-LIS
http://hdl.handle.net/1765/23489

Towards a Value-based Method for Risk Assessment in Supply Chain Operations
Lingzhe Liu, and Hennie Daniels
ERS-2011-014-LIS
http://hdl.handle.net/1765/23492

Beyond Waste Reduction: Creating Value with Information Systems in Closed-Loop Supply Chains
Otto Koppius, Oznur Ozdemir, and Erwin van der Laan
ERS-2011-024-LIS
http://hdl.handle.net/1765/26892

Function Approximation Using Probabilistic Fuzzy Systems
Jan van den Berg, Uzay Kaymak, and Rui Jorge Almeida
ERS-2011-026-LIS
http://hdl.handle.net/1765/30923

The Power Trading Agent Competition
Wolfgang Ketter, John Collins, Prashant Reddy, Christoph Flath, and Mathijs de Weerdt
ERS-2011-027-LIS
http://hdl.handle.net/1765/30683

---

* A complete overview of the ERIM Report Series Research in Management:
https://ep.eur.nl/handle/1765/1

ERIM Research Programs:
LIS Business Processes, Logistics and Information Systems
ORG Organizing for Performance
MKT Marketing
F&A Finance and Accounting
STR Strategy and Entrepreneurship