Learning revenue-maximizing orderings in sequential auctions

Sicco Verwer [a] Yingqian Zhang [b]

[a] *Katholieke Universiteit Leuven, Belgium*
[b] *Erasmus School of Economics, The Netherlands*

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. However, when historical data are available it is possible to *learn* good orderings that increase the revenue of the auctioneer. In this work, we show how such a learning model can be built based on previous auctions using regression trees. We provide a greedy method that finds a good sequence for a new set of items given the learned model. We design several experiment scenarios and test the performance of the proposed learning method. The experimental results are promising: they show that good orderings can be found quickly.

1 Introduction

Auctions are becoming increasingly popular for allocating resources or items in business-to-business and business-to-customer markets. When the amount of items is large, sequential auctions, where items are sold consecutively to a set of bidders (or agents), are often adopted. Such examples can be found in industrial procurement [7], and the Dutch flower auction [14]. In this paper we study sequential auctions from the auctioneer's point of view. More specifically, we examine that given a set of items, how to design an auctioning sequence for these items such that the auctioneer's revenue is optimized.

This revenue is heavily dependent on the ordering of items, especially when bidders have budget/capacity constraints [9, 6] or when they have preference over some bundles of items [12]. We demonstrate the importance of ordering on the expected revenue by the following two examples, one with budget constrained bidders and the other with heterogeneous preferences on items. In these examples, we assume agents are myopic, i.e., they bid truthfully in each auction round.

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

Example 2 *Suppose for one T and one R, agent A_2 only desires R with value 5. A_1 has complementary items, i.e., A_1 values T and R with 1 and 1 respectively if he wins only T or R at the end of the auction, however, if he wins both items R and T, the value $v(R \cup T)$ goes up to 10. Thus when A_1 has not yet won any bid, he will bid 1 for either T or R; if however A_1 has already bought either T or R, he will bid 9 for the coming R or T. It is quite obvious that in this case, the auctioneer should sell tulips T first as it will end up with revenue 10, in contrast to 6 if roses are sold first.*

Much of the existing work that studies optimal ordering in auctions focuses on theoretical analysis on bidders' strategy [5, 12] and conditions when the optimal ordering exits [6, 9, 12]. However, it is difficult to apply these results to actual auctions as they rely on many strong assumptions (for example bidders' valuation functions) which rarely hold in practice. In this paper, we develop a novel method for finding revenue-maximizing ordering 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 are held regularly, such data is readily available. Therefore, our method has high potential to be used in practice.

We briefly describe the auction setting as follows. There are a set of private value items for sale in a budget-constrained auction, where items are sold consecutively one at a time. There are a set of bidders, whose valuation functions and bidding strategies are unknown. Bidders may desire more than one item, and some of them may have complementary items, i.e., some combinations of items are preferred, as shown in Example 2. For each set of items, a sequential auction is held. This sequential auction is repeated over time, with probably different items. We do not assume any particular auction rule, i.e., sealed-bid or open-cry, first price or second price, but only that every sequential auction follows the same rule.

At the end of each sequential auction, we have the following information at our disposal: (1) the ordering of auctioning items; and (2) the revenue of each sold item. We develop 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, and finally uses a greedy method in order to find a good ordering for a new set of items (Section 3). We design several experiment scenarios and test the performance of the proposed learning method (Section 4). We end this paper with conclusions. We now start with describing some existing work.

2 Related work

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 [8] look at a setting where items are homogeneous and bidders desire only one item. Elmaghraby [6] studies procurement auction where a buyer outsources two heterogeneous jobs by sequential 2nd-price auctions. He shows that specific sequences lower procurement costs and identifies a class of suppliers' cost functions where efficient orderings exist.

Pitchik [9] 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 [12] investigate the optimal auctioning strategy of 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 [5] study how to maximize revenue in sequential auctions with second-price sealed-bid rules where bidders have a unit demand. Furthermore, bidders are homogeneous, i.e., all their valuations are drawn from public known uniform distributions. 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.

These results give some deep insights into the ordering design problem in sequential auctions. However their applicability is limited due to strong assumptions required, (highly) simplified settings used, and specific type of auction rules studied. In contrast, we intend to develop a method that can be used in practice without assuming any particular bidders' valuation distributions, no matter it is first-price or second-price, and seal-bid or open cry.

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 [11], not to mention learning one for every agent.

2. Given correctly learned utility functions, finding a good ordering is still hard [5].

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

3.1 Auction setting and modeling assumptions

There are a set of private value types of items for sale in a budget-constrained auction, where items are sold consecutively one at a time. There are a set of bidders, whose valuation functions and bidding strategies are unknown. Bidders have heterogeneous valuation functions. They may desire more than one item, and some of them may have complementary items. For each set of items, a sequential auction is held. This sequential auction is repeated over time. We do not assume any particular auction rule, but only that every sequential auction follows the same rule. In addition, we make the following two modeling assumptions.

Assumption 1 ***(Buyer independence)*** *The expected revenue for an ordering is independent of which agents participate in the auction.*

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. Furthermore, the assumption is 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 types of flowers, the interests of the group of participants remain stable. Thus, it makes sense not to model the possible dependence on auction participants.

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 still not straightforward how to apply machine learning techniques, however, 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 I, then the amount A is willing to pay for I depends on whether this is the last I being sold in the auction, or whether there are more to come. Another example, suppose A wants to buy I and J, then the amount A bids for I depends on whether A already obtained J, i.e., whether J was auctioned before I. In these examples, the exact ordering of items before and after I does not matter much for A's valuation of I. We believe this assumption slightly overgeneralizes the auction setting since there is a small possibility that the ordering determines whether A could have obtained J before 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

At the end of each sequential auction, we have the following information at our disposal: (1) the ordering of auctioning items; and (2) the revenue of each sold item. Given our two modeling assumptions, we have 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 [4] 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 [10]. These directly model the expected revenue per item, and we can build a state space that fits with our assumptions, but none of the models we know includes the possibility of auctioning different items (resulting in different available actions, and moving goal and start states).

Instead of learning a sequence model for the expected revenues of orderings, we therefore use a simple regression model for predicting the revenues of items. We provide this model with features that fit with our modeling assumptions, and that match with our intuition on why different orderings obtain different revenues (see Introduction). Currently, we provide the following features:

Feature 1 For every item type I, the amount of I items already auctioned.

Feature 2 For every item type I, the amount of I items still to be auctioned.

Feature 3 The sum of revenue so far.

Feature 4 For every item type I, the sum of revenues from I items so far.

The first two features fit with our second assumption and they model the influence of utility functions with complements (for some agents). For instance, if many agents want both I and J, and if the amount of J already auctioned is large when auctioning I, then we expect the revenue for this I to increase. The second two features are used to model the influence of budget constraints. For instance, if some agents want I, some want both I and J, and if the sum of revenues from J items is large, then we expect a higher revenue for I items. 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.

Example 3 *Suppose that packages of roses (R) and tulips (T) are being auctioned using the ordering $RRTRTTTR$, and that the revenues of the packages obtained in the auction are in order: 10,8,4,8,6,3,3,14. For every package-revenue pair in this auction we create one row in the data set with values for all the features described above:*

Type	*Revenue*	*R before (Feature 1)*	*R after (Feature 2)*	*T before (Feature 1)*	*T after (Feature 2)*	*Sum (Feature 3)*	*Sum R (Feature 4)*	*Sum T (Feature 4)*
R	*10*	*0*	*3*	*0*	*4*	*0*	*0*	*0*
R	*8*	*1*	*2*	*0*	*4*	*10*	*10*	*0*
T	*4*	*2*	*2*	*0*	*3*	*18*	*18*	*0*
R	*8*	*2*	*1*	*1*	*3*	*22*	*18*	*4*
T	*6*	*3*	*1*	*1*	*2*	*30*	*26*	*4*
T	*3*	*3*	*1*	*2*	*1*	*36*	*26*	*10*
T	*3*	*3*	*1*	*3*	*0*	*39*	*26*	*13*
R	*14*	*3*	*0*	*4*	*0*	*42*	*26*	*16*

We provide a data set obtained in this way as input to a standard regression method from machine learning. In our case, we learn regression trees using recursive partitioning techniques [3]. More specifically, we use the rpart package of the R statistical package [13]. The result is a predictive model for the expected revenue of new items.

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

Algorithm 1 Computing a good ordering

Require: A set of items S, a set of item-revenue orderings SO
Ensure: O is a good (high expected revenue) ordering
  Transform SO into a data set D, based on the method used in Example 3.
  Compute the mean future revenues of all item types, and add them to D
  for every item type T **do**
    Learn a regression model M_T from D for predicting the revenue of item type T
    for every item type T' **do**
      Learn a regression model $M_{T,T'}$ from D for predicting the future revenue of item type T' after auctioning T
    end for
  end for
  Let $O = ()$ be an empty ordering, and let $T(I)$ be a function that returns the type of item I
  while S is not empty **do**
    for every item I in S **do**
      Create a data set row R for having auctioned O with their expected revenues, then auctioning I, and having $S - I$ as remaining items.
      Compute $Rev = M_{T(I)}(R) + \sum_{I' \in S - \{I'\}} M_{T(I),T(I')}(R)$
    end for
    Select an item I with highest revenue Rev
    Remove I from S and add I to O
  end while
  Return O

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. We could try to construct a search algorithm that uses the structure of the learned model to prune the search tree. But since computing a good ordering for an auction when given the agents utility functions is NP-hard [5], this is not likely to succeed in general. Instead, our approach is to use a simple greedy method that iteratively selects items with largest expected revenue for the ordering. The expected revenue for choosing a certain item I is composed of two parts: the expected revenue of auctioning I now and the expected revenue of all the remaining items. The first part is determined by the learned model. For the second part, additional models are learned that can predict the expected future revenue per item type. For an item type T, the expected future revenue is simply the mean value of all revenues for items of type T after auctioning I. We can compute these mean values in the data set and add them as additional targets for learning. The table in Example 3 then becomes:

Type	Revenue	R before	R after	T before	T after	Sum	Sum R	Sum T	Mean R	Mean T
R	10	0	3	0	4	0	0	0	10	4
R	8	1	2	0	4	10	10	0	11	4
T	4	2	2	0	3	18	18	0	11	4
R	8	2	1	1	3	22	18	4	14	4
T	6	3	1	1	2	30	26	4	14	3
T	3	3	1	2	1	36	26	10	14	3
T	3	3	1	3	0	39	26	13	14	NA
R	14	3	0	4	0	42	26	16	NA	NA

Thus, for every item auctioned in an ordering, we compute the mean revenues for the remaining items. When of a certain type no items remain, its mean revenue is given a missing value (NA). With both the expected and future expected revenues, computing a good order becomes easy. Given a set of items S, choose an item I that maximizes the sum of the expected revenue of I and future revenues of all items in $S - \{I\}$, and iterate. The result will be an ordering with high expected revenue. A detailed overview of our method for computing a good ordering is given in Algorithm 1.

4 Experiments

To test our method, we created a simple auction simulator of a first-price auction, with myopic agents[1] who may have different utility functions. With this simulator we generated small data sets (orderings of item-revenue pairs) and tested whether our method was able to learn good orderings.

4.1 Learning in a simple setting

In our first experiment, we generated auction outcomes for a simple auction scenario with 4 agents who participate in an auction of 4 rose packages and 4 tulip packages. The utility functions and budgets of the 4 agents are given by the following table:

	Rose	Tulip	Rose+Tulip	Budget
Agent 1	10	-	-	10
Agent 2	6	-	-	100
Agent 3	8	5	-	20
Agent 4	4	3	20	100

The table reads as follows. The first agent desires only roses and is willing to pay 10 for a single package. He has a constrained budget of 10. The second agent also wants only roses but values them at 6 per package. If allowed by the other agents, his budget is sufficient for him to buy all of the roses in the auction. The third and fourth agents desire both roses and tulips for the shown values. In addition, however, the fourth agent likes to obtain pairs of roses and tulips and is willing to pay a large amount (20) in order to obtain these. The third agent has a constrained budget of 20, allowing him to obtain only a few possible combinations of rose and tulip packages.

In this simple auction setting, the allocation of flowers to agents is influenced by both budget constraints and complementary items. It is clear that in an optimal ordering agent 4 should get as many rose-tulip pairs as possible. For this to occur, these tulips should be ordered before these roses, otherwise agent 2 will buy the roses instead of agent 4 since he values them higher. In addition, agent 1 and agent 3 need to be out of budget, or they will buy the roses and the tulips instead of agent 4. An ordering is optimal when there are still 2 possible rose-tulip pairs after agent 3 is out of budget, for instance:

ordering:	R	T	T	R	T	T	R	R
sold to:	A_1	A_3	A_3	A_3	A_4	A_4	A_4	A_4
revenue:	10	5	5	8	3	3	17	17

This sequence is optimal and results in a revenue of 68. We demonstrate that our technique is able to learn the patterns resulting from combinatorial and budget influences by showing it can learn to produce such an optimal ordering, when given only suboptimal ones as input. Specifically, we perform the following steps:

1. Generate a set S of 40 random permutations of the 4 tulip and 4 rose packages.
2. Simulate the result of auctioning the flowers using the orderings S in an auction with the four agents.
3. Remove the orderings in S that obtain a revenue of 68.
4. Learn models M from the suboptimal orderings and resulting revenues (as in Algorithm 1).
5. Use our greedy approach on M to find a good ordering s^* (as in Algorithm 1).
6. Check whether s^* is optimal.

Since the generation of the different orderings is random, we performed these steps 10 times. In 8 of the 10 runs the result is an optimal sequence such as $RRTTTTRR$ or $RTRTTTRR$. This result shows our method correctly generalizes over input orderings. We repeated the experiments with 400 suboptimal orderings as input. In these runs, the result is always an optimal sequence, showing that the result of our method also converges to the optimal ordering.

In the next section, we show that it also correctly generalizes over the amounts and types of flowers that are available for auctioning, and over the number and types of agents that participate in the auction.

[1] Our method should also work for non-myopic agents, but testing this requires a much more complex simulator. We leave testing this for future work.

4.2 Learning in a complex setting

The auction setting described in the previous section uses small and fixed sets of agents and flowers. This makes it possible to test the capability of learning the optimal ordering, but it is not very realistic. In this section, we test our method using a more realistic simulation with larger and flexible sets of agents and flowers. We use the following 4 types of flowers: rose (R), tulip (T), lily (L), and orchid (O). For a single auction, we draw uniformly at random 2 to 7 flowers of every type. These flowers are then randomly ordered and offered in sequence to a set of agents. This set of agents is also randomly constructed by choosing 1 to 4 of every agent type uniformly at random from the following list:

type	R	T	L	O	$R+T$	$R+L$	budget
1	$[8, 12]$	-	-	-	-	-	$[10, 30]$
2	-	$[3, 7]$	$[6, 10]$	-	-	-	$[10, 30]$
3	-	-	-	$[13, 17]$	-	-	$[10, 30]$
4	$[2, 6]$	$[2, 6]$	-	-	$[20, 25]$	-	$[10, 30]$
5	$[2, 5]$	-	$[2, 5]$	-	-	$[26, 30]$	$[10, 30]$

The utility functions and budgets of the agents are drawn uniformly at random from the intervals in the above list. Like in the previous section, the auction outcome is influenced by budget constraints and complementary items. In addition, it is influenced by which flowers are to be sold and which agents take part in the auction. Furthermore, all these influences are determined by randomly drawn numbers. This makes finding a good ordering very challenging because one ordering can get very different revenues depending on the values of these numbers.

We simulate 250 auctions with random orderings of the flowers and provide these as input to our method. Based on these examples our goal is to find a good ordering for 5 packages of roses, 5 packages of tulips, 5 packages of lilies, and 5 packages of orchids. Since the revenue of this ordering depends on which agents take part, we simulate 250 auctions of these flowers with a fixed ordering that is determined by our method. The mean revenue of these auctions is compared with the mean revenue of random orderings of the same flowers. The obtained mean revenues are 135 and 131, respectively. This difference is statistically significant, and thus our method outperforms random orderings.

This is a nice result but it does not tell us how good the ordering found by our method really is. To determine this, we would have liked to compare its performance with an optimal ordering. Unfortunately, it is not easy to find this ordering due to the combinatorial and random nature of the auction simulation. Therefore, we simulated 250 runs of many (about 100000) orderings and computed the mean revenue for each of these orderings. Out of all of these orderings only 3% performed better than 135, showing that the ordering found by our method is very good indeed. Furthermore, the best mean revenue of all of these orderings is 136.5, which is only a little better than the one produced by our method. We point out that this best ordering is determined using a simulation model, it is not based on the 250 example orderings. Therefore, it is not fair to compare these performances, it only shows how far the ordering produced by our method is from optimal when given only 250 example orderings. We also tested the ordering of putting the most valuable item first (as suggested in [12]), but this clearly does not work in our setting as this obtains a mean revenue of only 128, which is even worse than random. We summarize the results in a small table:

ordering	mean revenue
our method	135
random	131
best found	136.5
most valuable first	128

These results are encouraging as they show that our method is able to learn a very good ordering for a complex auction setting from a small amount of examples.

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 revenue-maximizing 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 a greedy method that finds a good ordering for a new set of items from the learned models. We develop an auction simulator and design two auction scenarios, one simple and the other more complex, to test the performance of the proposed method.

The experimental results are promising: 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. 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.

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] L. Breiman, JH Friedman, R. Olshen, and CJ Stone. *Classification and regression trees*. Wadsworth International Group, 1984.

[4] Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA, 2010.

[5] 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.

[6] Wedad Elmaghraby. The importance of ordering in sequential auctions. *Manage. Sci.*, 49:673–682, May 2003.

[7] Jérémie Gallien and Lawrence M. Wein. A smart market for industrial procurement with capacity constraints. *Manage. Sci.*, 51:76–91, January 2005.

[8] Paul R Milgrom and Robert J Weber. A theory of auctions and competitive bidding. *Econometrica*, 50(5):1089–1122, September 1982.

[9] Carolyn Pitchik. Budget-constrained sequential auctions with incomplete information. *Games and Economic Behavior*, 66(2):928–949, July 2009.

[10] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[11] 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.

[12] Ramanathan Subramaniam and R. Venkatesh. Optimal bundling strategies in multiobject auctions of complements or substitutes. *Marketing Science*, 28:264–273, March 2009.

[13] TM Therneau and Atkinson EJ. An introduction to recursive partitioning using the rpart routines. Technical report, 1997.

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