

Bidding for Customer Orders in TAC SCM: A Learning Approach

David Pardoe and Peter Stone
The University of Texas at Austin
Department of Computer Sciences
Austin, TX 78712 USA
{dpardoe, pstone}@cs.utexas.edu

Abstract

Supply chains are a current, challenging problem for agent-based electronic commerce. Motivated by the Trading Agent Competition Supply Chain Management (TAC SCM) scenario, we consider an individual supply chain agent as having three major subtasks: acquiring supplies, selling products, and managing its local manufacturing process. In this paper, we focus on the sales subtask. In particular, we consider the problem of finding the set of bids to customers in simultaneous reverse auctions that maximizes the agent's expected profit. The key technical challenge we address in this paper is that of determining the probability that a customer will accept a particular bid price. First, we compare several machine learning approaches to estimating the probability of bid acceptance. We then perform experiments in which we apply our learning method during actual gameplay to measure the impact on agent performance.

1. Introduction

Supply chains are a current, challenging problem for agent-based electronic commerce. One problem commonly faced by agents acting in supply chains is that of negotiating with customers in order to sell goods. Such negotiations are often handled through reverse auctions in which sellers submit sealed bids in response to requests for quotes (RFQs) from customers. This situation becomes particularly difficult when sellers must bid in multiple auctions simultaneously, because an agent cannot await the outcome of one auction before bidding in another. When deciding which auctions to bid in and what bids to place, an agent with limited resources must be able to judge and balance the competing risks of not winning enough auctions and of winning too many. In the former case, it is unable to fully utilize its resources towards profitability; in the latter, it will be unable to meet its obligations to customers.

The Trading Agent Competition Supply Chain Management (TAC SCM) scenario [11] provides a perfect testbed for the study of this problem. In TAC SCM, agents competing as computer manufacturers must handle three basic subtasks: acquiring components, managing a local manufacturing process, and selling assembled computers to customers. Agents receive incomplete information about the state of the game and have a limited amount of time in which to make decisions, resulting in a challenging competition. The prob-

lem studied in this paper is motivated by our work on Tac-*Tex* [9], a successful agent entered in the first TAC SCM competition. From our experience, we have identified the sales subtask as the most crucial aspect of the TAC SCM scenario.

In this paper, we focus on the problem of determining the optimal set of bids for an agent to make in response to RFQs for computers received from customers. The key technical challenge we address is that of determining the probability that a customer will accept a particular bid price. Given the ability to make such predictions, an agent can then apply a search method to find the most profitable set of bids.

The remainder of this paper is organized as follows. In Section 2 we give a brief summary of the TAC SCM scenario and provide information on related work. We give a complete description of the problem we are solving in Section 3. In Section 4 we present a comparison of several machine learning approaches to estimating the probability of bid acceptance. In Section 5 we measure the impact of learning on agent performance by performing controlled experiments involving actual TAC SCM games. Section 6 proposes directions for future work and concludes.

2. Background

In this section, we give a brief summary of the TAC SCM scenario, emphasizing the parts that are most relevant to the sales subtask, and provide information on related work.

2.1. The TAC SCM Game

In a TAC SCM game [1], six agents act as computer manufacturers in a simulated economy that is managed by a game server. The length of a game is 220 simulated days, with each day lasting 15 seconds of real time. At the beginning of each day, agents receive messages from the game server with information concerning the state of the game, such as the customer RFQs for that day. Agents have until the end of the day (i.e. $< 15s$) to send messages to the server indicating their actions for that day, such as bids on RFQs. The game can be divided into three parts: production and delivery, component supply, and computer demand.

In this paper, we focus on the computer demand, or sales, aspect of the TAC scenario. Customers wishing to buy computers send all six agents identical RFQs consisting of:

- the type of computer desired (1 of 16);

- the quantity of computer desired (1–20);
- the due date (3–12 days in the future);
- a reserve price indicating the maximum amount the customer is willing to pay; and
- a penalty that must be paid for each day the delivery is late. Orders are canceled on the fifth late day.

Reserve prices range from 75% to 125% of the base price of the requested computer type, multiplied by the quantity, and penalties range from 5% to 15% of the reserve price. The base price of a computer is equal to the sum of the base prices of its parts [1]. Agents respond to the RFQs by making offers to sell at a certain price, with the agent offering the lowest bid on each RFQ winning the order. Agents are unable to see the prices offered by other agents or even the winning prices, but they do receive a report each day indicating the highest and lowest price at which each type of computer sold on the previous day.

The number of RFQs that come from customers depends on the level of customer demand, represented by a parameter D . The actual number of RFQs each day is drawn from a Poisson distribution with D as its mean. Fluctuation in demand is modeled by multiplying D by an amount representing the current trend each day. This trend follows a random walk, and D is bounded between 80 and 320, with its initial value chosen uniformly randomly from this range.

2.2. Related Work

The problem of predicting the probability of winning an auction with a particular sealed bid is commonly approached through statistical methods such as those surveyed in [8]. Such methods often require extensive historical information about competitors' past bids and assume a static environment. In TAC SCM, probabilities vary considerably throughout the game, and almost no information is available about competitors' bids while the game is running. A machine learning approach similar to that used in this paper is developed by [7], which uses a naive Bayes classifier to predict the probability of a bid winning based on the bid price, features of the RFQ being bid on, and available information about other bidders.

In much of the previous work on agents trading multiple goods in simultaneous auctions, the auctions considered are English or similar auctions. For instance, [5] and [14] consider a separate TAC scenario in which agents acting as travel agents must satisfy customer preferences by bidding for hotel rooms and other items, and [10] presents an agent that purchases cargo space on flights through forward auctions and provides end-to-end shipping services to customers through reverse auctions. Agents participating in simultaneous English auctions have several advantages over agents participating in simultaneous sealed bid auctions: the bids of other agents can be viewed, bids can be revised, and auctions may have different closing times, providing agents with knowledge of some auction outcomes while there is

still time to make decisions on other auctions.

A solution to the TAC SCM bidding problem similar to the one used in this paper is presented in [2], which uses linear regression on recent bidding results to form predictions of bid acceptance and then uses stochastic programming to determine optimal bids. Additional approaches are described in [6] and [3].

3. Problem Specification

We now specify the problem we are addressing in this paper. We consider the problem of an agent participating in a TAC SCM game that must decide what bids to place on the RFQs it has received from customers on a given day. The inputs to the agent's decision process are the following:

- The set of customer RFQs;
- The agent's available resources (components and assembled computers in inventory along with the future production cycles); and
- Information about past auctions (the agent's knowledge of its own bids and the reported highest and lowest prices at which each type of computer sold)

Because there are many more computers requested each day than any one agent can produce, the goal of an agent is not to win every auction, but to find the set of bids that maximizes the agent's expected profit without committing the agent to produce more computers than it possibly can. (Viewing TAC as a game, an agent's goal should be to maximize its profit relative to the profits of competing agents, but due to the difficulty of determining the effect an agent's bids will have on other agents, we will assume our agent is only concerned with its own profit. In a real supply chain, this profit maximization would be the true goal.) A simple approach to this problem is to predict the highest price at which each auction could be won and to bid this price on several of the more profitable auctions, expecting to win each one. A more sophisticated approach involves considering the possibility of placing high bids on many auctions in the hopes of winning some fraction of them.

This second approach is the one used by TacTex, our agent in the first TAC SCM competition, and is the approach that is considered in this paper. An agent implementing this approach has two requirements:

1. The ability to form estimates of the probability of winning an auction as a function of the bid price
2. A means of using these estimates to find the set of bids that maximizes the agent's expected profit

We consider primarily the first component in this paper, and leave a comparison of implementations of the second component to future work. In Section 4, we experiment with different machine learning approaches to predicting the probability of bid acceptance. In Section 5, we experiment using actual TAC SCM games to compare the performance of an agent using learning against agents using fixed heuristic predictors.

4. Learning Auction-Winning Probabilities

Predicting the probability of winning an auction in TAC SCM is a challenging problem for three main reasons: (i) agents receive very limited information on auction results, (ii) no two auctions are the same due to the differing attributes of each RFQ, and (iii) winning prices can fluctuate rapidly due to changing game conditions. As a result, an approach based on analyzing past auction results from the current game is unlikely to yield accurate predictions. We therefore turn to machine learning methods using training data from many past games.

The problem we are trying to solve can be viewed as a multiple regression problem. This could be solved by using a regression learning algorithm to learn the probability of winning an auction as a function of factors including the bid price. We instead follow a modified approach used by [13] to solve a similar conditional density estimation problem from a different TAC scenario. This approach involves dividing the price range into several bins and estimating the probability of winning the auction at each bin endpoint. A post-processing step converts the learned set of probabilities to a probability density function by interpolating between bin endpoints and enforcing a monotonicity constraint that ensures that probabilities decrease as prices increase. In this method, a separate predictor is trained for each endpoint to predict the probability of winning at that point. The concept to be learned by each predictor is therefore simpler than the concept that would be learned if we used a single predictor for all prices. We leave an empirical comparison with the latter approach for future work.

In this section we focus on the task of training these individual predictors. We describe the format of the training data, compare the effectiveness of several learning algorithms, and then look at the impact that the choice of training data has on the predictions. It is important to note that training is done off-line, so the game’s time constraints are not a factor.

4.1. Training Data Format

The data for our experiments is taken from the results of the semifinal and final rounds of the first TAC SCM competition held in August 2003. Winning bids for customer RFQs can be obtained from game logs made available immediately after each game terminates. Several hundred thousand RFQs were issued over the course of the games, providing ample data for training and testing. In order to eliminate any start-game and end-game effects, only RFQs sent between day 50 and day 200 (out of a 220 day game) are considered, thereby simplifying the concept to be learned.

A training instance is created for each RFQ. The 23 attributes included in each instance reflect the details of the RFQ it represents, along with the information available to agents at the time about the level of demand in the game and

the recent prices for which the requested type of computer has been sold. Each instance contains the current date; the quantity, penalty, due date, and reserve price for the RFQ; and the highest and lowest prices at which the requested computer type was sold over the past five days. The additional attributes provided about customer demand give a picture of how the daily number of RFQs has varied over the course of the game. All monetary values are expressed as a fraction of the computer’s base price.

A separate predictor is trained for each price point x at which we want to predict the probability of winning an auction, where x is expressed as a fraction of a computer’s base price. For a given value of x , each auction is labeled with a 1 if the winning bid was greater than x and with a 0 otherwise. Instances representing RFQs receiving no bids are labeled with a 1 if x is less than or equal to the reserve price.¹

4.2. Algorithm Comparison

We first performed an experiment comparing the effectiveness of using several different regression learning algorithms to train predictors: neural networks (with a single hidden layer and using backpropagation), M5 regression trees, M5 regression trees boosted with additive regression (which successively fits a new base learner to the residuals left from the previous step), decision stumps (single-level decision trees) boosted with additive regression, J48 decision trees, J48 decision trees boosted with AdaBoost, and BoosTexter. BoosTexter [12] is a boosting method that was originally designed for text classification and is the algorithm used in [13]. It uses decision stumps as the base learner and a form of AdaBoost to weight each training instance between rounds of training, outputting a weighted sum of the learned decision stumps. Other algorithms we considered were support vector machines, naive Bayes, and k-nearest neighbors, but these did poorly in initial testing. For all algorithms other than BoosTexter, we used the implementations provided in the WEKA machine learning package [15], using default parameters. Informal attempts at tuning these parameters did not appear to significantly affect performance.

For comparison, we also include the results obtained from using a simple heuristic predictor that gives reasonably good results. For each of the past five days, the predictor forms a uniform density function on the interval between the highest and lowest prices reported for the requested computer type. A weighted sum of these density functions, with those from more recent days receiving more weight, is then used as a probability density function from which estimates of bid acceptance are taken.

¹ Note that this formulation represents the standpoint of a seventh agent wanting to know the probability that none of the other six agents would place a bid below x .

In our experiment we evaluated each of the learning algorithms on a data set taken from the final round of the competition. We used cross-validation, meaning that training and test data came from the same games. While the true value of a probability prediction is the utility gained from using it, determining this in the context of a TAC SCM agent is not feasible, and so we instead use root mean squared error between the predicted probabilities and actual outcomes as the measure of comparison. We ran separate tests predicting the probability of winning an auction at several different values of x . The results of three 10-fold cross-validations with $x = 0.7$ are presented in Figure 1 and are similar to results for other values of x . With a large number of training instances, the tree-based methods clearly had the best performance, followed by BoosTexter. The errors of the non-tree-based methods level off after a limited number of training instances, while the errors of the tree-based methods continue to decrease until the point at which all available training data is used. For training sets of size 200,000 and 370,000, the difference observed between each pair of algorithms in Figure 1 is statistically significant at the 95% confidence level.

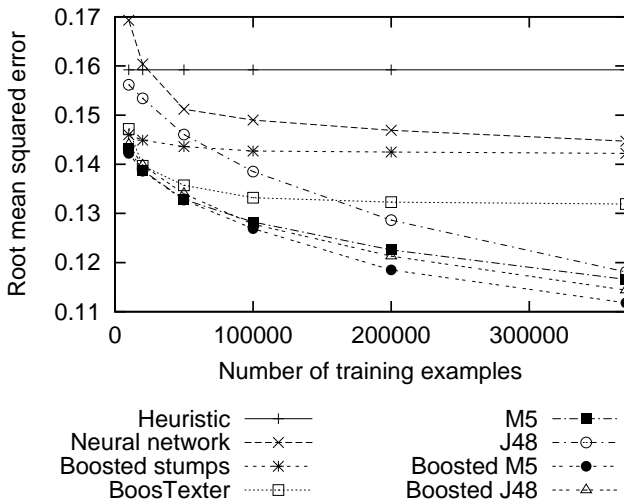


Figure 1. Results for $x = 0.7$

4.3. Choice of Training Data

In the previous experiment, the training data and test data were taken from the same set of games, with the same agents participating in each game. This raises the possibility that the algorithms learned concepts that pertained to specific games and set of agents but were not applicable in general. This is also unrealistic in the TAC setting, as an agent could not have predictors trained on data from the game it is currently participating in. In practice, it is important to know whether a predictor trained for one set of agents will be reliable in games with a different set of agents. Our next experiment addresses these issues.

We consider the case of an agent participating in the final round of the competition. The agent would want to train predictors on the data most relevant to the situation. At the beginning of the final round, the most relevant data would likely come from the results of the semifinal round, which contained two brackets of six agents each. As a result, this data would reflect on both the agents in the final round and the agents defeated in the semifinal round. After the finals begin, the agent would be able to analyze the results of completed games from the finals and would have the option of retraining its predictors with this new data, either by itself or in combination with the data from the semifinals.

We performed an experiment comparing the results of training with these choices of training data. First, we divided the games from the final round into two halves, labeled finals1 and finals2. We then used finals2 as the test data for predictors trained on data from different sources: the semifinals, finals1, the semifinals combined with finals1, and finals2 (using cross validation). The results for M5 trees and BoosTexter, the top two performing algorithms, are shown in Figures 2 and 3. Again, $x = 0.7$. The learning curves are labeled with the source of data used for training.

When the predictors were trained on data other than finals2, the performance gap between M5 trees and BoosTexter disappeared, and the performance of the other tree-based methods, even boosted M5 trees, fell behind. The errors of the tree-based methods no longer continued to decrease as more training instances were used, and sometimes the error increased, as observed in Figure 2 when data from the semifinals was used for training. This suggests that the strong performance of the tree-based methods in the first experiment was largely due to their ability to learn game-specific factors that do not generalize well. While BoosTexter appears to achieve somewhat lower errors than M5 trees in this experiment, further testing on different game scenarios would need to be done to determine whether this is the case in general.

As we expected, the predictors trained on data from finals2 outperformed the predictors trained on data from different games. Still, the performances of the latter were better than that of the heuristic. The predictors trained on finals1 performed better than those trained on the semifinals, confirming that more relevant training data produces better results. Somewhat surprisingly, the predictors trained on the combination of finals1 and the semifinals performed better than the predictors trained on finals1 only. It may be that a predictor trained on data from a variety of sources will generalize the best to a new situation, even if some of the training data is less relevant for the new situation.

The results of these experiments suggest that with the right choice of learning algorithm and training data, we can learn the probability of winning an auction reasonably well.

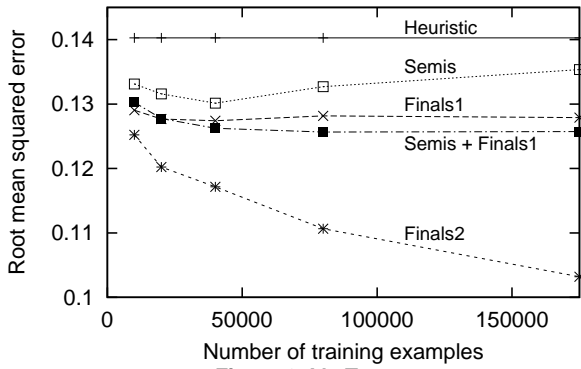


Figure 2. M5 Trees

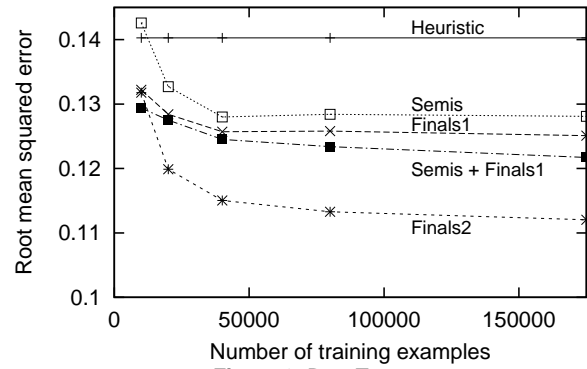


Figure 3. BoosTexter

However, to truly test the value of our predictions, we need to use them as the input to a method of selecting bids in actual TAC games. That is the focus of the next section.

5. Agent Performance

In this section, we evaluate the effectiveness of our learning approach when used as part of a complete agent in TAC SCM gameplay. We do this through controlled experiments in which agents differing only in their predictions of bid acceptance play against each other repeatedly.

5.1. Agent Design

The basic design of each agent is essentially that of our entry in the 2003 TAC SCM competition, TacTex [9], with some minor modifications. Production and delivery are handled by a greedy production scheduler that gives near-optimal performance in practice. The module responsible for bidding on customer RFQs determines what resources will be available after filling current orders, creates a function for each RFQ that maps bid prices to the probability of bid acceptance, and performs a heuristic search using these functions to try to find the set of bids that maximizes the agent’s expected profit. In order to isolate the effects of bidding, we modified the game settings to allow each agent to receive an effectively unlimited quantity of each component on the 15th game day at no cost, eliminating the need for a strategy for purchasing components from suppliers. This is not entirely unrealistic, as many agents in the competition actually ordered the majority of their components on the first game day [4]. Agents were only allowed to carry up to 200 of each type of computer in inventory, to prevent them from using their limitless components to build up large computer inventories during periods of low customer demand. The effect of this limitation was to increase the responsiveness of computer prices to changes in demand, creating a more dynamic and interesting game scenario.

5.2. Predictions of Bid Acceptance Probabilities

The sole experimental variable we consider is the method used to predict the probability of bid acceptance. The names of the six competing agents and the

prediction methods used are as follows:

- **Learner:** Uses learning as described in Section 4.
- **Heuristic (2 agents):** Use the heuristic described in Section 4.2.
- **OldHeuristic (2 agents):** Use a simpler prediction heuristic originally used in TacTex [9]. Testing on game data has shown this heuristic to be less accurate than the one used by Heuristic.
- **Simple:** To provide some variety in bidding behavior, this agent simply bids on all RFQs at the same fraction of the base price. The fraction has a lower bound of 0.7 and increases proportionally with the number of orders the agent currently has.

5.3. Experimental Setup

Three rounds of 30 games were played between the six agents. During the first round, Learner used the same heuristic as Heuristic. The game logs from the first round were then used to train a set of predictors to be used by Learner in the second round. We trained a separate predictor for each of the 26 price points between 0 and 1.25 times the base price spaced at an interval of 0.05, using BoosTexter as the learning algorithm and using 10% of the available data (about 100,000 instances). Because the learning agent is trying to outbid only the other agents and not itself, its own bids were ignored when determining the winning bid for each training instance. The functions mapping bids to probabilities of acceptance are created from the 26 predictions by enforcing a monotonicity constraint and interpolating as described in [13], with the added step of setting all probabilities for bids above the reserve price to 0. A second round of games was then played.

In the third round, Learner used a set of predictors that had been trained on the logs from the semifinal and final rounds of the 2003 TAC SCM competition. The purpose of this was to determine how well the predictors would generalize to a different set of agents.

Agent	Relative Score		
	Round 1	Round 2	Round 3
Learner	4.32 ± .32	12.73 ± .56	6.77 ± .54
Heuristic (2)	4.24 ± .34	2.98 ± .46	3.34 ± .53
OldHeuristic (2)	-.74 ± .25	-3.38 ± .27	-1.93 ± .19
Simple	-11.32 ± .89	-11.93 ± .75	-9.59 ± .99

Table 1. Average relative score (in millions of dollars)

5.4. Results

The results are presented in Table 1. The results for the two Heuristic agents were nearly identical, and so only an average is presented. The same is true for the two OldHeuristic agents. The average relative score of each agent is given along with the standard deviation. An agent’s relative score in a game is its score minus the average score of all agents for that game. The average score over all agents and games in each round was around \$90 million. Because all agents were initially given sufficient components to last the whole game, no component costs are included in any of the scores presented.

From the results of the first round, we can see that using more accurate predictions leads to a clear improvement in performance. In fact, Learner and Heuristic had higher scores than OldHeuristic in all 30 games.

The results of the second round show exactly what we had hoped to see: using learning significantly improved agent performance. Learner scored higher than Heuristic by a large margin in all 30 games and by an average margin of nearly \$9 million.

In the third round, Learner still managed to outperform the other agents, but by a smaller margin, and not in every single game. Considering that the predictors used by Learner were trained on games involving a completely different set of agents, and a somewhat different game scenario (i.e., a limited component supply), this result is very promising. In actual competition, we would not have access to a large number of games involving only the agents we are competing against, and this experiment suggests that learning could still be successfully applied in such a case.

6. Future Work and Conclusion

In this paper, we considered the problem faced by an agent acting in a supply chain that must bid in simultaneous reverse auctions to win orders from customers. Using TAC SCM as a test domain, we presented a learning approach to the task of predicting the probability that a bid will be accepted by a customer. A comparison of learning algorithms showed that M5 regression trees and BoosTexter result in similar prediction accuracy when testing and training data come from separate games. When used as part of a complete agent, learned predictors were shown to provide a significant improvement in performance over heuristic predictors.

One important result demonstrated was that the learned predictors generalize well to new situations, both in terms of

prediction accuracy and of agent performance. This gives us hope that our learning approach can be used successfully in competition when facing different sets of agents or agents that change their behavior over time.

There are several possible ways in which predictions could be improved. The results of Section 4.3 suggest that acquiring data from a variety of situations might aid in training a more robust predictor. Further experiments could determine the best combinations of data for an agent to use. Also, additional information available to an agent could be included as features, such as knowledge of the availability and prices of components. Finally, an agent could make use of its knowledge of auction results during a game to make on-line improvements to its predictors. Boosting-based predictors would lend themselves well to this approach, since making incremental modifications to the existing predictors would be straightforward.

An agent’s bidding performance depends on both the accuracy of its predictions and the method it uses to select a set of bids using these predictions. While we have considered only the former issue in this paper, we plan to address the problem of searching for optimal sets of bids in future work.

Acknowledgments

This research was supported in part by NSF CAREER award IIS-0237699.

References

- [1] Raghu Arunachalam, Joakim Eriksson, Niclas Finne, Sverker Janson, and Norman Sadeh. The TAC supply chain management game. Technical report, Swedish Institute of Computer Science, 2003. Draft version 0.62.
- [2] Michael Benisch, Amy Greenwald, Ioanna Grypari, Roger Lederman, Victor Naroditskiy, and Michael Tschantz. Botticelli: A supply chain management agent designed to optimize under uncertainty. *SIGecom Exchanges*, 4(3):29–37, February 2004.
- [3] Erik Dahlgren and Peter Wurman. PackaTAC: A conservative trading agent. *SIGecom Exchanges*, 4(3):38–45, February 2004.
- [4] J. Estelle, Y. Vorobeychik, M.P. Wellman, S. Singh, C. Kiekintveld, and V. Soni. Strategic interactions in a supply chain game. Technical report, University of Michigan, 2003.
- [5] Amy Greenwald and Justin Boyan. Bid determination in simultaneous auctions—a case study. In *Proceedings of the Third ACM Conference on Electronic Commerce*, pages 115–124, Tampa, FL, 2001.
- [6] C. Kiekintveld, M.P. Wellman, S. Singh, J. Estelle, Y. Vorobeychik, V. Soni, and M. Rudary. Distributed feedback control for decision making on supply chains. In *International Conference on Automated Planning and Scheduling*, 2004.
- [7] R. D. Lawrence. A machine-learning approach to optimal bid pricing. In *Proceedings of the Eighth INFORMS Computing Society Conference on Optimization and Computation in the Network Era*, Arizona, 2003.

- [8] V. Papaioannou and N. Cassaigne. A critical analysis of bid pricing models and support tool. In *IEEE International Conference on Systems, Man and Cybernetics*, Piscataway, NJ, 2000.
- [9] David Pardoe and Peter Stone. TacTex-03: A supply chain management agent. *SIGecom Exchanges*, 4(3):19–28, February 2004.
- [10] Chris Preist, Claudio Bartolini, and Andrew Bye. Agent-based service composition through simultaneous negotiation in forward and reverse auctions. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 55–63. ACM Press, 2003.
- [11] Norman Sadeh, Raghu Arunachalam, Joakim Eriksson, Niclas Finne, and Sverker Janson. TAC-03 a supply-chain trading competition. *AI Magazine*, Spring 2003.
- [12] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
- [13] Robert E. Schapire, Peter Stone, David McAllester, Michael L. Littman, and János A. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimation. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.
- [14] Peter Stone, Robert E. Schapire, János A. Csirik, Michael L. Littman, and David McAllester. ATTac-2001: A learning, autonomous bidding agent. In *Agent Mediated Electronic Commerce IV: Designing Mechanisms and Systems*, volume 2531 of *Lecture Notes in Artificial Intelligence*, pages 143–160. Springer Verlag, 2002.
- [15] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.