A Sequence Mining Method to Predict the Bidding Strategy of Trading Agents

Conference Paper · May 2009

DOI: 10.1007/978-3-642-03603-3_11 · Source: DBLP

Pericles A. Mitkas
Aristotle University of Thessaloniki

A Sequence Mining Method to Predict the Bidding Strategy of Trading Agents

Vivia Nikolaidou and Pericles A. Mitkas

Aristotle University of Thessaloniki,
Department of Electrical and Computer Engineering,
Thessaloniki, Greece
+30 2310 996349, +30 2310 996390
vivia@ee.auth.gr, mitkas@eng.auth.gr

Abstract. In this work, we describe the process used in order to predict the bidding strategy of trading agents. This was done in the context of the Reverse TAC, or CAT, game of the Trading Agent Competition. In this game, a set of trading agents, buyers or sellers, are provided by the server and they trade their goods in one of the markets operated by the competing agents. Better knowledge of the strategy of the trading agents will allow a market maker to adapt its incentives and attract more agents to its own market. Our prediction was based on the time series of the traders' past bids, taking into account the variation of each bid compared to its history. The results proved to be of satisfactory accuracy, both in the game's context and when compared to other existing approaches.

1 Introduction

On-line auctions, especially the ones operated by autonomous agents, are a relatively new area of research, while still being extensively used in the real world. The subject is both wide and complex enough to guarantee an ample research potential, which is combined with the possibility to immediately deploy the obtained results to real applications. On the other hand, the information explosion during the last decade has yielded vast amounts of data, out of which useful results have to be extracted. Combined with the ever increasing processing power available, it has given a boost to the research of Data Mining techniques.

One such technique is Sequence Mining, which has generated considerable interest mainly due to its potential for knowledge extraction in large sets of sequential data. Sequence mining tries to identify frequent patterns in datasets where data items appear in a somewhat predictable fashion, such as text, aminoacid chains, or a sequence of process calls in a computer. Traditional data mining techniques, such as classification and clustering, cannot be directly applied but can be modified for sequence mining. Alternatively, sequential data can be transformed to a form that is suitable for processing by one of the existing algorithms.

The wide adoption of intelligent agents in research and practice is showing more and more examples where they are called to extract knowledge out of data, sequential or not. Agent developers can also exploit large data repositories by extracting knowledge

L. Cao et al. (Eds.): ADMI 2009, LNCS 5680, pp. 139–151, 2009.

models that can be embedded into intelligent agents [11]. Electronic auction environments represent the biggest application domain of software agents. It is in this domain where the successful blending of Intelligent Agents and Data Mining proves fruitful for both of these technologies.

In this work, we try to predict the bidding strategy of a large set of trading agents in the CAT game of the Trading Agent Competition [4]. Since the prediction is mostly based on analyzing the bids' history, we employ a novel Sequence Mining technique, which watches the variation of the bids according to their history, in order to classify the bid sequences. The architecture of the agent, for which we designed our technique, is also presented in this context.

The rest of this paper is organized as follows: Section 2 analyzes the bidding strategies that we were called to distinguish, while Section 3 presents the CAT game that was used as a framework. Section 4 explains the architecture of our agent, Mertacor, which incorporated the resulting model. Section 5 focuses on the challenging issues of Sequence Mining and subsequently analyzes our approach to this issue, eventually arriving to the results' presentation. Section 6 reviews existing literature approaches to similar issues. The work concludes with Section 7, which discusses the results and gives clues for future works.

2 Bidding Strategies

Intelligent agents, due to their autonomy and their reasoning capabilities, have long been used in auctions as bidding entities with great success. They can easily be designed to implement various bidding strategies and to adapt them, if needed. Agents can also make timely decisions that require complex computations. Although several more bidding strategies exist, the following ones are adopted in the CAT game.

2.1 Zero-Intelligence Constrained (ZI-C)

One of the most well-known strategies is the Zero-Intelligence trading strategy, first developed by Gode and Sunder [6]. This strategy randomly selects a bid based on a uniform distribution, without taking into account any market conditions or seeking any profit, hence the term Zero-Intelligence. In order to avoid possible loss, they define the Zero-Intelligence Constrained (ZI-C) strategy, used in the CAT game. This strategy sets the item's cost value as a minimum boundary to the bidding price, whereas the maximum value remains the same as in ZI-U.

2.2 Zero-Intelligence Plus (ZIP)

Since the ZI-C strategy, described in 2.1, draws bids randomly, the efficiency achieved is not enough to reach the efficiency of markets with human trading agents. As a result, Cliff in [2] introduced the ZIP strategy, in which the trading agents increase or reduce their profit margin by watching the market's conditions. More specifically, they monitor the following values: all bid and ask prices in the market, irrespective of whether the corresponding shouts led to a transaction or not, as well as the transaction price itself. The agents then adjust their profit margin accordingly.

At the beginning of a trading day, all ZIP agents have an arbitrarily low profit margin. When a transaction occurs that indicates that they could acquire a unit at a more convenient price, their profit margin is increased. However, ZIP agents have a back-up strategy, which prevents them from raising their profit margin too high. When a buyer's shout gets rejected, the shout price is increased and, when a seller's shout gets rejected, the shout price is decreased. Similarly, they watch transactions made by competing sellers and lower their profit margin if needed, so as to not be undercut by competing sellers or buyers.

2.3 Gjerstad-Dickhaut (GD)

Gjerstad and Dickhaut in [5] defined a bidding strategy based on belief functions, which indicate how likely it is that a particular shout will be accepted. This is achieved by watching the history of observed market data - namely, the frequencies of submitted bids and asks, as well as the frequencies of bids and asks that lead to a transaction. Since more recent bids and asks are of higher importance than older ones, the authors introduce a sliding window function in the agents' memory, that only takes into account the latest L shouts in the market's history.

Their belief functions are based on the assumptions that, if an ask is accepted, all asks at a lower price will also be accepted and, if an ask has been rejected, all asks at a higher price will also be rejected. Similarly, if a bid is accepted, all bids at a higher price will also be accepted and, if a bid is rejected, all bids at a lower price will also be rejected.

2.4 Roth-Erev (RE)

Roth and Erev's purpose was to create a strategy that would mimic the behavior of human players in games with mixed strategy equilibria [3]. Therefore, they used reinforcement learning algorithms on the agent's profit margins, in order to adjust them to the market's conditions.

This strategy only depends on the agent's direct feedback with the market mechanism and is therefore independent of the auction mechanism itself. More specifically, both ZIP and GD require the trading agents to have access to the history of bids and asks, as well as all accepted transactions and their prices. However, RE does not require any such data, but only relies on the same agent's interaction with the market mechanism. Therefore, it is generic enough to be used in any auction environments.

3 Reverse TAC ("CAT") Game

The CAT game is being held in the context of the Trading Agent Competition or TAC [4]. Since the markets in this game are not fixed, but instead created by each competing agent, it is called Reverse TAC, or CAT. The name CAT also refers to Catallactics, the science of economic exchange.

In the CAT game, a set of trading agents is generated by the game itself, while the contestants' purpose is to design specialist agents. Each specialist agent will operate a single market and set the rules for it. Trading agents are free to choose only one market in each operating day, and they can only buy and sell in the market that they choose.

All trading agents are either buyers or sellers and remain so for the entire game. They all buy or sell the same item in single-unit auctions. However, they are allowed to trade several items per day, placing a new bid or ask after a completed transaction.

The trading agents, buyers or sellers, incorporate one of the four trading strategies described in Section 2. Their private values, or estimations of the value of the goods traded, are drawn from a random distribution. They also incorporate a market selection strategy, which helps them choose the market that they judge to be most profitable to them. The traders' private values, bidding strategies, market selection strategies, as well as trade entitlement (the number of items that they are allowed to trade each day) are unknown to the specialists.

The specialists' goal is to design the rules and conditions of their markets. More specifically, they have to define:

a) The agent's charging policy. Agents announce their fees at the beginning of each day. They may include fees for one or more of the following:
   - ⋆ Registration fee: paid by each trading agent who registers on this particular market on this day
   - ⋆ Information fee: paid by each trading agent and each other specialist who requests information on the market's shouts and transaction
   - ⋆ Shout fee: paid for each shout placed
   - ⋆ Transaction fee: a standard sum paid for each transaction by each agent involved
   - ⋆ Profit fee: a percentage on the transaction's price, paid by both the buyer and the seller
b) The market's accepting policy. This policy defines which shouts, bids or asks made by the traders are accepted and which are rejected.
c) The market's closing condition. Although a trading day consists of a number of trading rounds, announced at the beginning of the game, the market does not have to close at the end of each round, but instead is free to close at any time during the trading day.
d) The matching policy for the shouts – more particularly, which bid is matched to which ask, in order to form a successful transaction.
e) The pricing policy for each transaction. Each transaction closes at a different price, same for both the buyer and the seller.

The game consists of an unknown number of days, given in the server's configuration file but not announced. Each day consists of a number of trading rounds, with each round having a certain duration. At the beginning and at the end of each day, as well as between each day, there is some free time in order for agents to complete any possible calculations. All these durations, apart from the game's duration, are announced at the beginning of the game.

Scoring is only made during assessment days, which start and end at some random point in the game's duration. The randomness of these days, as well as the game's duration being unknown, have as a purpose to avoid exploiting initial and final conditions by the specialists, but instead focus on constructing stable markets that are able to function properly at any duration.

Scoring on each trading day and for each specialist is determined by the sum of the following three factors:

i. The agent's profit for each day, divided by the total profit of all agents for the day, in order to be normalized on a scale from 0 to 1.
ii. The agent's market share, as in, the ratio of traders subscribed to this particular market on this day, again normalized in a scale from 0 to 1.
iii. The agent's transaction rate, as in, the ratio of shouts accepted that led to a successful transaction. This is again a number from 0 to 1. Rejected shouts are not calculated.

The interested reader can find more information on the CAT game in [4].

4 Agent Mertacor

Agent Mertacor employs a modular architecture, as shown in Figure 1. Using a combination of microeconomics theory and heuristic strategies, it tries to maximize its profit without losing a significant portion of market share. The main parts of the agent are briefly discussed below.

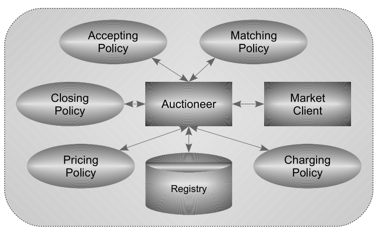

Fig. 1. Mertacor's architecture

4.1 Auctioneer

The Auctioneer is the central part in Mertacor's architecture. It is in charge of coordinating the other parts of the agent, especially when it comes to communication between the Market Client, the Registry and the policy modules of the agent. It is also in charge of computing the global equilibrium point, based on bid history. The equilibrium point is the price where the curve of offer distribution, sorted from lowest to highest, meets the curve of demand distribution. The term "global" refers to the fact that all agents are taken into account for this computation, whether they belong to the agent's particular market or not.

4.2 Market Client

The Market Client is the element that is responsible for the communication with the server. It transforms all information from the internal representation structure of the agent to the form that is understandable by the server and back. It is also responsible for subscribing to the markets of competing trading agents, in order to receive information on shouts and transactions made in their market. The reason behind this is that, normally, a specialist only has information on other specialists' profit and market share at the end of each day. Further information can only be acquired by subscribing to the other specialist's market, paying the appropriate information fee, if set.

4.3 Accepting Policy

The accepting policy is a tricky part in the agents' design. It must be low enough to keep a high transaction rate, but still high enough to guarantee many transactions. Mertacor's accepting policy is divided into two parts.

The first part is activated at the beginning of the game, before the global equilibrium point is calculated. At this point, Mertacor implements the same rule used by NYSE: namely, in order for a shout to get accepted, it must be at a price better than the day's best shout, or the day's "quote".

As soon as the global equilibrium point is calculated, Mertacor switches to an equilibrium-beating accepting policy. This means that, in order for a shout to get accepted, it must be better than the global equilibrium computed. This works as an attraction to intra-marginal traders (buyers and sellers whose private values are, respectively, higher and lower than the global equilibrium), while keeping extra-marginal traders (buyers and sellers whose private values are, respectively, lower and higher than the global equilibrium) from trading goods.

4.4 Matching Policy

Mertacor uses the matching policy described by Wurman et al in [14]. Incoming unmatched bids and asks are sorted to two separate heaps. When a bid is matched with a shout, they are automatically moved to two separate "matched" heaps. When the market is closed, the highest bids are matched with the lowest asks. This method maximizes both social welfare and Mertacor's profit margin.

4.5 Clearing Condition

Mertacor's condition for clearing the market is again twofold. In the first phase of the game, before the global equilibrium point is computed, Mertacor behaves like NYSE, clearing the market at a given probability after each shout. This method, called "continuous clearing", has been proven to be adequately efficient, while keeping a high transaction throughput [6].

After the global equilibrium point calculation is completed, Mertacor switches to a modified round-clearing condition. This means that the market is closed at the end of each round. The variation used is that Mertacor switches again to a continuous clearing policy towards the end of the game, in order to maximize throughput.

4.6 Pricing Policy

Like the accepting policy and the clearing condition, Mertacor's pricing policy is also divided into two phases: before and after the global equilibrium point is computed.

During the first phase, Mertacor uses a variation of a discriminative k-pricing policy [10]. The k parameter is computed as the ratio of sellers in this game, resulting in a k of 0.5 for a balanced market. Our policy slightly favors sellers when there are more buyers and favors buyers when there are more sellers, giving them an incentive to balance the market.

In the second phase of the game, Mertacor switches to a global equilibrium pricing policy. This means that the price for all transactions is the same price as the global equilibrium. This way, Mertacor gives each trader the same profit that they would gain in an efficient global partitioning.

In both cases, though, Mertacor favors intra-marginal traders at the expense of extra-marginal ones. This means that, if a transaction is executed between an intra-marginal trader and an extra-marginal one, Mertacor will clear it at the price given by the intra-marginal trader.

4.7 Charging Policy

Choosing the right charging policy is the most challenging task in the design of a CAT agent. Setting the fees charged too low may not yield enough revenue, while values that are too high may discourage traders from joining the specialist's market.

As explained in Section 3, each specialist may impose five different fees: a) registration fee, b) information fee, c) shout fee, d) transaction fee, and e) profit fee.

Mertacor only imposes the profit fee, keeping the other ones down to zero. This is decided keeping in mind that only successful transactions must lead to a payment – in other words, an agent who does not earn anything should not pay anything either.

Heuristic experiments showed that the optimal value (i.e., the value which maximizes the specialist's score) is 0.2. This can go as low as 0.1, whenever the specialist's market share is too low, but as high as 0.3, when the specialist estimates that the market conditions are suitable.

5 Predicting the Bidding Strategy

In order to classify each trading agent as intra-marginal or extra-marginal, we needed to know its private value. The most crucial step in determining it was to find the agent's bidding strategy, which had to be predicted by observing its bidding history.

5.1 Modeling the Problem

The problem of predicting the bidding strategy of an agent can be seen as a classification problem, where each agent has to be assigned one of four labels (ZI-C, ZIP, GD or RE). However, it differs from conventional classification problems, in that the most characteristic input data is the past history of bids. With each bid having a specific timestamp and being correlated to its past bids, the problem is identified as a sequence mining one, where various time series have to be matched into labels.

Sequence mining is of particular interest because of various reasons. The first is the high dimensionality of the problem. More particularly, in our case, the history of past bids can comprise several hundreds or even thousands of samples, depending on the game's duration. One can easily understand that it is necessary to reduce the problem's dimensionality, allowing for better performance of the classification algorithms.

However, high dimensionality is but the tip of the iceberg. In many cases, including ours, the number of dimensions is not known a priori, but instead continuously grows according to the bids history. For example, during the first trading days of the game, we often have fewer than ten past samples. However, as the game advances and bids are constantly added to the history logs, the time series grows longer.

The third and most important issue in sequence mining is the notion of "sequence". This means that it is not merely a set of incoming value-timestamp pairs, but each value and each timestamp is related to its history. In fact, most information can be deduced by looking at the bid history and not at a bid itself, since, for example, ZI-C could give practically any bid value observed separately. On the other hand, conventional classification or clustering algorithms treat each feature as independent of the other ones and do not look for relationships between features. This means that it is not easy to examine each bid's history.

The most common approach in bibliography so far, for example by Martinez-Alvarez et al [9], is to use a moving window to split the time series and feed the latest N samples to the classifier. The authors take this approach one step further by normalizing input data. However, this still does not provide a satisfactory solution to the third issue described above, since each sample is still treated independently in the algorithms.

5.2 Our Approach

In our approach, we take this philosophy one step further by introducing deltas to the classifier. More specifically, let us consider a set of prices p_0... p_n, with p_0 representing the most recent one, and their corresponding set of timestamps, t_0... t_n. Notice that increasing subscript values in the sequence denote earlier points in the timeline. We define a sliding window of size 6, but only the first value p_0 is fed as-is to the classifier's input. For each of the rest of the values p_1 to p_5, we subtract the previous one in the series (more recent in the bid history). The results, d_0 to d_4, give the classifier the notion of sequence, therefore facing the corresponding issue.

Furthermore, we have four additional features as the classifier's inputs. These are calculated by subtracting, for each one of d_1 to d_5, the value of the second previous sample, d_0 to d_4 accordingly.

Since the difference in time is more independent statistically than the difference in bid value, we only define a simple sliding window of size 3 when it comes to time differences. This means that we have three time-related attributes: t_0, t_0 - t_1, and t_1 - t_2.

The input dataset is completed by adding two more attributes independent from the time series, namely the type of the trader (buyer or seller), as well as whether this particular bid led to a successful transaction.

Since agents have limited time available for calculations while the game is running, the model was built offline. Classification was continuous, with the traders being assigned a label multiple times during the course of the game. In the case of a wrong prediction, traders' private value would be estimated wrongly, which might eventually lead to their misclassification as intra-marginal or extra-marginal.

Training data was initially taken from our own experiments with the CAT platform. This was used in the first version, designed for the qualifying rounds of the game. However, for the final games, we retrained the model using values from the qualifying games only. Only qualifying games 2 and 4 were used. The reason behind this is that games 1 and 3 were not run at full-length, but instead lasted only 100 trading days, which is only barely enough for the traders to finish exploring the markets and settle to their preferred one. While the number of trading days is unknown due to the game specifications, it was hinted that they would last much longer. In any case, since our model's inputs only depended on the data so far, the algorithm was designed to perform well in full-length as well as slightly shorter games.

Algorithm Selection. The next step was the selection of the most suitable algorithm. This is done in a manner similar to [12], but skipping the data preprocessing part. We compared the following three algorithms: Neural Networks, J48 and Support Vector Machines. We used the WEKA platform [13] to train the model, using 10-fold cross validation.

We eventually decided to discard Support Vector Machines because of its slow response and inadequate performance for the specific input data. Time was an important issue, since the trading rounds are of fixed duration. Additionally, the classification accuracy did not exceed 30%. Meta-classification might have slightly improved it, but it might have been at the expense of complexity.

Of the remaining two algorithms, J48 outperformed Neural Networks, giving a classification accuracy of 56.5 percent. Confirming our findings in [12], meta classification, in the form of Bagging, further enhanced the model's performance, boosting it to 65.5 percent. Final results are presented in Table 1.

Table 1. Summarized classification results

Algorithm	Correctly Classified	Incorrectly Classified	Kappa Siatistic	Mean abs. error	RMS error
Bagging - J48	65.55%	34.45%	0.54	0.24	0.34
J48	56.47%	43.53%	0.42	0.23	0.44
SMO	29.22%	70.78%	0.03	0.36	0.46
Perceptron	45.45%	54.55%	0.26	0.33	0.41

Tables 2 amd 3 presents the detailed accuracy by class, as well as the confusion matrix, for the winning algorithm, the combination of Bagging and J48.

The Effect of Difference Order. In our dataset, we used differences of first and second order, which are a rough representation of the first and second derivative. In order to further illustrate the effect of difference order in the results, we constructed three datasets: price1, price2 and price3, which contain the differences of first, first and second, and first to third order, respectively. The chosen algorithms were run on all three datasets and the comparative results are depicted in Table 4.

Table 2. Detailed accuracy by class for winning algorithm

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.65	10.3%	68.0%	0.66	0.67	0.87	GD
0.72	13.7%	65.7%	0.73	0.69	0.89	ZIC
0.61	16.2%	58.8%	0.61	0.60	0.83	ZIP
0.61	6.2%	72.8%	0.62	0.67	0.88	RE

Table 3. Confusion matrix for the winning algorithm

GD	ZIC	ZIP	RE	< −− classified as
4936	536	1296	739	GD
326	**5787**	1533	107	ZIC
1210	1562	**5034**	417	ZIP
782	926	700	**3909**	RE

Table 4. Classification accuracy by algorithm and difference order

	price1	price2	price3
Perceptron	43.37	45.45	45.00
SMO	29.26	29.22	29.25
J48	56.66	56.47	55.76
Bagging – J48	65.21	65.55	65.67
Average	48.63	49.17	48.92

We observe that the dataset which contains the second-order difference has the highest average classification accuracy on this dataset. For the highest-performing algorithm, namely the combination of Bagging and J48, we observe a slightly better performance for the price3 dataset. However, since all differences are computed on the fly during the course of the game in order to obtain a correct classification, we decided to use the price2 dataset, which also gives the highest average performance and needs less computational resources than price3.

6 Related Work

The most relevant work in bibliography is the one of Gruman and Narayana in [7]. They compared the performance of Support Vector Machines (SVMs) and Hidden Markov Models (HMMs). Their experiments were made in benchmark-like tests, using traders and specialists of controlled variations. They obtained a classification accuracy ranging from 52 to 62 percent using HMMs, according to fine-tuning of the HMM's parameters. Our approach not only outperforms it, but it is also implemented in "real-game" instead of controlled conditions. Furthermore, in contrast to HMMs, the decision-tree-based J48 hardly requires any fine-tuning, making it more robust when tested in the real world.

Bapna et al in [1] attempt to predict trading agents' Willingness-To-Pay in auction environments. When it comes to classifying agents' bidding strategies, they assume a single bidding strategy for all traders, namely the MBR (Myopic Best Response) strategy, and classify traders as Evaluators, Participators and Opportunists, as well as MBR and non-MBR traders. Details on the classification method, the input data, as well as classification accuracy, are not given, since the work's focus is mainly on Willingness-To-Pay and trader classification is merely used as an intermediate step.

When it comes to sequence classification, one example is the work by Lesh et al in [8]. They present an algorithm which examines all features of the given sequence and selects the ones to be used as input to traditional classification algorithms, such as Na?ve Bayes or Winnow. Zaki enhances this work in [15] by taking into account issues, such as length or width limitations, gap constraints and window size. This approach has increased complexity, but still does not take into account the continuity of the time series.

The sliding window approach prevails in more recent works, such as Martinez-Alvarez et al in [9]. They have a sequence clustering problem, in which they normalize the input data and subsequently determine the optimal window size. Taking into account the similarities between classification and clustering, this approach further supports the novelty of our work taking into account deltas instead of a sliding window, even normalized.

7 Conclusions and Future Work

We have used sequence mining, a data mining technique, to improve the decision mechanism of an agent. Using this technique, a specialist agent can successfully predict the bidding strategy of a set of trading agents. Our model was designed as part of a widely accepted, general-purpose trading agent competition game. Nevertheless, it remains generic enough to allow its adaptation to a multitude of generic auction environments with little or no modification.

The classification accuracy proved to be satisfactory, since our agent was able to correctly classify a large majority of the traders. Using J48, an algorithm based on decision trees, also had the advantage of quick performance, allowing Mertacor more time for other calculations.

Mertacor proved competitive by ranking fifth in the final games out of a total of 14 participants. Games were often augmented by the eventual presence of one or more test agents.

Future work in this direction will most likely be headed towards the direction of further introducing the notion of continuity into the models. The main current problem is that, while there is only one correct class for each time series, our model makes several predictions, one for each bid placed. Results could probably be enhanced by a meta-layer, which would take into account previous predictions of the same model for each agent and decide accordingly.

Acknowledgements

This work is part of the 03ED735 research project, implemented within the framework of the Reinforcement Programme of Human Research Manpower (PENED) and cofinanced by National and Community Funds (25% from the Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.E.-European Social Funding).

References

[1] Bapna, R., Goes, P., Gupta, A., Karuga, G.: Predicting Bidders' Willingness to Pay in Online Multiunit Ascending Auctions: Analytical and Empirical Insights. Informs Journal on Computing 20(3), 345–355, INFORMs (2008)

[2] Cliff, D.: Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical Report HP-97-91. Hewlett-Packard Research Laboratories, Bristol, England (1997)

[3] Erev, I., Roth, A.E.: Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. American Economic Review 88(4), 848–881 (1998)

[4] Gerding, E., McBurney, P., Niu, J., Parsons, S., Phelps, S.: Overview of CAT: A market design competition. Technical Report ULCS-07-006. Department of Computer Science, University of Liverpool, Liverpool, UK, Version 1.1 (2007)

[5] Gjerstad, S., Dickhaut, J.: Price formation in double auctions. Games and Economic Behaviour 22, 1–29 (1998)

[6] Gode, D.K., Sunder, S.: Allocative efficiency of markets with zero-intelligence traders: Market as a partial sustitute for individual rationality. Journal of Political Economy 101(1), 119–137 (1993)

[7] Gruman, M.L., Narayana, M.: Applications of classifying bidding strategies for the CAT Tournament. In: Proceedings of the International Trading Agent Design and Analysis Workshop, TADA 2008, Chicago, IL, USA, July 14, pp. 11–18. AAAI, Menlo Park (2008)

[8] Lesh, N., Zaki, M.J., Ogihara, M.: Mining features for sequence classification. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge Discovery and Data Mining, San Diego, CA, United States, August 15-18, pp. 342–346. ACM, New York (1999)

[9] Martinez – Alvarez, F., Troncoso, A., Riquelme, J.C., Aguilar – Ruiz, J.S.: LBF: A Labeled-Based Forecasting Algorithm and Its Application to Electricity Price Time Series. In: Proceedings of the Eighth IEEE International Conference on Data Mining, ICDM 2008, Pisa, Italy, December 15-19, pp. 453–461. IEEE, Los Alamitos (2008)

[10] Sattherthwaite, M.A., Williams, S.R.: The Bayesian Theory of the k-Double Auction. In: Friedman, D., Rust, J. (eds.) The Double Auction Market – Institutions, Theories, and Evidence, ch. 4, pp. 99–123. Addison-Wesley, Reading (1993)

[11] Symeonidis, A.L., Mitkas, P.A.: Agent Intelligence through Data Mining. Springer Science and Business Media (2005)

[12] Symeonidis, A.L., Nikolaidou, V., Mitkas, P.A.: Sketching a methodology for efficient Supply Chain Management agents enhanced through Data Mining. International Journal of Intelligent Information and Database Systems 2(1), 49–68 (2008)

[13] Witten, I.H., Eibe, F.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, New Zealand (1999)
[14] Wurman, P.R., Walsh, W.E., Wellman, M.P.: Flexible double auctions for electronic commerce: theory and implementation. Decision Support Systems 24(1), 17–27 (1998)
[15] Zaki, M.J.: Sequence mining in categorical domains: incorporating constraints. In: Proceedings of the ninth international conference on Information and knowledge management, McLean, Virginia, United States, November 6-11, pp. 422–439. ACM, New York (2000)