Electronic Commerce Research and Applications 8 (2009) 63–77

Contents lists available at ScienceDirect

Electronic Commerce Research and Applications

journal homepage: www.elsevier.com/locate/ecra

Forecasting market prices in a supply chain game ☆

Christopher Kiekintveld [a,*], Jason Miller [b], Patrick R. Jordan [a], Lee F. Callender [a], Michael P. Wellman [a]

[a] Computer Science and Engineering, University of Michigan, 2260 Hayward Avenue, Ann Arbor, MI 48109-2121, USA
[b] Department of Mathematics, Stanford University, Building 380, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Received 17 October 2007
Received in revised form 4 November 2008
Accepted 5 November 2008
Available online 28 November 2008

Keywords:
Forecasting
Markets
Price prediction
Trading agent competition
Supply chain management
Machine learning

ABSTRACT

Predicting the uncertain and dynamic future of market conditions on the supply chain, as reflected in prices, is an essential component of effective operational decision-making. We present and evaluate methods used by our agent, Deep Maize, to forecast market prices in the trading agent competition supply chain management game (TAC/SCM). We employ a variety of machine learning and representational techniques to exploit as many types of information as possible, integrating well-known methods in novel ways. We evaluate these techniques through controlled experiments as well as performance in both the main TAC/SCM tournament and supplementary Prediction Challenge. Our prediction methods demonstrate strong performance in controlled experiments and achieved the best overall score in the Prediction Challenge.

1. Introduction

The quality of economic decisions made now depend on market conditions that will prevail in the future. This is particularly true of supply chain environments, which evolve dynamically based on complex interactions among multiple players across several tiers. We present and evaluate methods of forecasting market conditions developed to predict prices in the Trading Agent Competition supply chain management game (TAC/SCM). In TAC/SCM, automated trading agents compete to maximize their profits in a simulated supply chain environment. They face many challenges, including strategic interactions and uncertainty about market conditions. Though complex, this domain offers a relatively controlled environment amenable to extensive experimentation. TAC also attracts a diverse pool of participants, which facilitates benchmarking and evaluation. This is particularly important for prediction tasks, where the quality of predictions depends in part on the actions of opponents. Research competitions such as TAC offer the opportunity to test predictions in environments where the opposing agents are designed and selected by others, mitigating an important source of potential bias relative to using opponents of one's own design (Wellman et al., 2007).

Our agent for the SCM game, Deep Maize, relies on price predictions in both the upstream and downstream markets to make purchasing, production, and sales decisions. Fig. 1 shows an overview of the daily decision process. Predictions about market conditions feed into an approximate optimization procedure that uses values to decompose the decision problem. The architecture is described in detail elsewhere (Kiekintveld et al., 2006). Since our predictions are integrated into a task-situated agent, we can evaluate the benefits of improved predictions on overall performance in addition to prediction error metrics.

We apply machine learning to forecast prices in two different markets with distinct negotiation mechanisms and information–revelation rules. Like many real markets, the available information on market conditions comes from a variety of heterogeneous sources. Integrating different learning techniques and representations to exploit all of the available information is a central theme of our approach. We find that representing the current market conditions well is particularly important for effective learning.

We begin with a discussion of related work and an overview of the TAC/SCM game, highlighting the benefits and drawbacks of the different forms of information available for making predictions. The first set of methods we describe are used for predicting prices in the downstream consumer market. Central to all of these methods is a nearest-neighbors learning algorithm that uses historical

☆ This is a revised and substantially extended version of a paper presented at the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (Kiekintveld et al., 2007).

* Corresponding author. Tel.: +1 734 818 0259.
E-mail addresses: ckiekint@umich.edu (C. Kiekintveld), jmiller@math.stanford.edu (J. Miller), prjordan@umich.edu (P.R. Jordan), asleep@umich.edu (L.F. Callender), wellman@umich.edu (M.P. Wellman).
URLs: http://teamcore.usc.edu/kiekintveld (C. Kiekintveld), http://www.eecs.umich.edu/~prjordan (P.R. Jordan), http://ai.eecs.umich.edu/people/wellman (M.P. Wellman).

1567-4223/$34.00 © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.elerap.2008.11.005

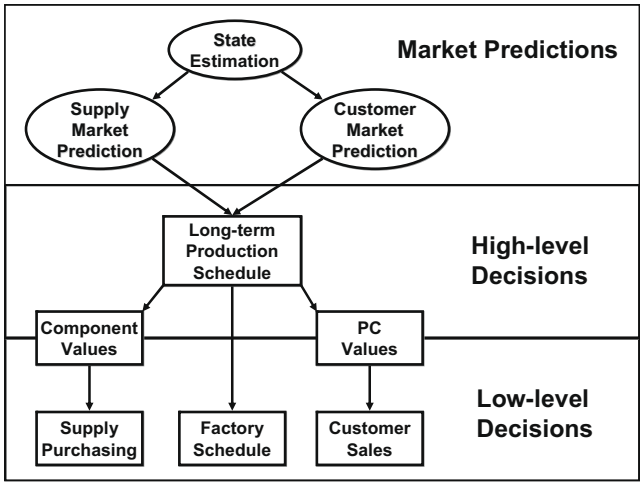

Fig. 1. Overview of Deep Maize's decision process on each TAC/SCM day.

game data to induce the relationship between market indicators and prices. We test variations that employ two different representations for price distributions: an absolute representation, and a relative representation that exploits local price stability. Finally, we develop an online learning method that procedure that combines and shifts predictions using additional observations within a game instance. Our evaluation of these methods includes both prediction error and supply chain performance. We show substantial improvements over a baseline predictor, with online adaptation exhibiting especially strong performance improvements.

We proceed to discuss methods for making predictions in the market for component supplies. There are two elements to our approach in this market. The first is a linear interpolation model that uses recent observations to make relatively accurate predictions about the current spot market. The second is a learning model that regresses prices against market indicators. This learning model can be used to make absolute price predictions, or to predict residuals for the linear interpolation model. We show that combining the linear interpolation model for current market prices with residual predictions improves overall performance, particularly for long-term predictions.

We close with discussion of the performance of both prediction algorithms based on the results of TAC/SCM tournaments and the inaugural TAC/SCM Prediction Challenge. Deep Maize has been a perennial finalist in the primary competition, and placed first in the Prediction Challenge. Our forecasting methods demonstrate strong performance on prediction error measures and contribute to strong overall performance in the context of decision-making.

2. Related work

Forecasting is studied across many disciplines, and many approaches have been developed for making predictions based on historical data. We do not attempt to survey all of the various methods here, but refer the reader to one of many introductory texts that describe exponential smoothing, GARCH, ARIMA, spectral analysis, neural networks, Kalman filters, and other popular methods (Brockwell and Davis, 1996; Chatfield, 2001; Hastie et al., 2001). Pindyck and Rubinfeld (1998) is a good introduction to many forecasting methods used in the context of economic predictions and modeling. The proliferation of techniques is in part an indication that the best method for any particular problem may depend on many factors, including the type and quantity of information available and the properties of the domain.

Many applications focus on comparing methods to identify the most effective approach for a particular problem (for example, see the application of forecasting to retail sales by Alon et al. or the application to predict electricity prices by Nogales et al. (2002)). Our approach for price prediction focuses on combining predictions made using difference sources of information and methods. Various methods have been developed for combining predictions, often motivated by the problem of combining the opinions of experts (Bates and Granger, 1969; Clemen and Winkler, 1999). Recently, predictions markets have become a popular method for aggregating information (Hanson, 2003). Our approach shares some features with these markets, including the use of logarithmic scoring rules to evaluate the quality of predictions.

Price forecasting is a very important element of decision-making in a wide variety of market settings, including financial markets (Kaufman, 1998) and energy markets (Nogales et al., 2002). The rise of internet auctions has led to an increasing interest in computational methods for price forecasting in auction environments (e.g., Ghani, 2005). Brooks et al. (2002) discusses many issues related to learning price models in the context of online markets. Price prediction methods have also been used as the basis for successful bidding strategies in many specific types of auctions, including simultaneous ascending auctions (Osepayshvili et al., 2005). It is not surprising then that price prediction has played an important role in the design of agents for the Trading Agent Competition since its inception. For the TAC travel game (Wellman et al., 2007), a crucial factor in agent performance was the ability to predict the closing price of auctions for hotel rooms. Participants explored many approaches for making these predictions, spanning statistical analysis, machine learning, and economic modeling (Cheng et al., 2005; Stone et al., 2003; Wellman et al., 2004). Post-tournament analysis revealed that the common thread among the most effective methods was not the particular technique applied, but rather the information taken into account by the predictor (Wellman et al., 2004); specifically, the best predictors accounted for the initial auction prices.

Most agents for the TAC/SCM game also employ some form of price prediction, but vary significantly both in what quantities are predicted and the methods used for making predictions. Many agents only estimate prices for the current simulation day, and do not attempt to forecast changes in prices over time (Benisch et al., 2004, 2006; Burke et al., 2006; He et al., 2006). One of the most successful agents in the tournament, TacTex, has applied a variety of sophisticated approaches for price prediction. The specific methods have evolved in successive tournaments, but generally rely on a combination of statistical machine learning applied to historical data and online adaptation (Pardoe and Stone, 2005, 2006, 2007a,b). The most recent version of TacTex specifically uses machine learning to predict both current and future prices. The MinneTAC team has also devoted significant effort to developing prediction methods for the TAC/SCM domain (Ketter, 2007; Ketter et al., 2007). Their approach is somewhat unique in that it focus on predicting changes in the economic regimes, which reflect over-supply, under-supply, or other characteristic market conditions. Our use of market state is similar in spirit to the idea of an economic regime, but we do not distinguish among a discrete set of possible regimes. The first version of Deep Maize used an analytic solution based on competitive equilibrium to predict market prices (Kiekintveld et al., 2004). This approach became infeasible when the game rules were modified due to the greater complexity of the supplier pricing model. Another unique approach was used by RedAgent, the winner of the first TAC/SCM competition (Keller et al., 2004). This agent used an internal market to coordinate decisions and make predictions; to our knowledge no current agents use this approach.

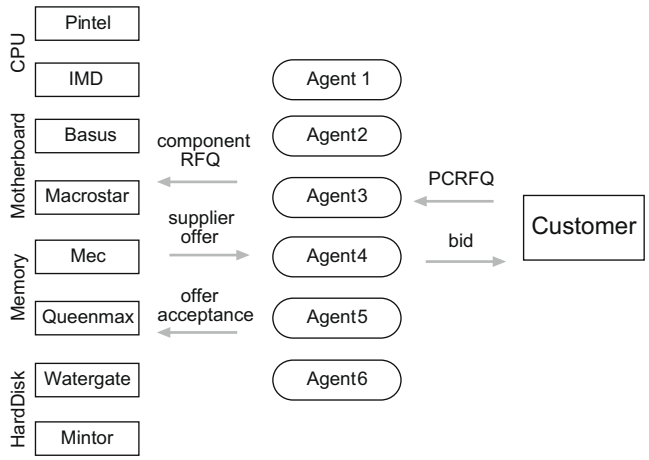

Fig. 2. TAC/SCM supply chain configuration.

3. The TAC supply chain game

Agents in TAC/SCM play the role of personal computer (PC) manufacturers (Collins et al., 2007; Eriksson et al., 2006).[1] The six automated agents compete to maximize their profits on a simulated supply chain, shown in Fig. 2. Agents assemble 16 different types of PCs for sale, defined by compatible combinations of CPU, motherboard, hard disk, and memory components. Each day agents negotiate with suppliers to purchase components, negotiate with customers to sell finished products, and create manufacturing and shipping schedules for the next day. Each agent has an identical factory with limited capacity for production. Each game has 220 simulated days, each of which takes 15 s of real time.

3.1. The personal computer market

The 16 PC types are separated into three market segments representing high-, mid-, and low-grade PCs. The average demand level for each segment varies separately according to a random walk with a stochastic trend. Each day, the customer demand process generates a set of RFQs (request-for-quotes) representing the aggregate demand. Each customer request is valid for a single day and specifies a PC type, due date, reserve price, and penalty for late delivery. The negotiations for these requests use a simultaneous sealed-bid auction mechanism. Agents submit price offers for a subset of the RFQs. The lowest bidder for each request is awarded the order, and must deliver the product by the due date or pay the penalty specified in the contract. Due to the configuration of the market, there are strong interactions among auctions both within and across days.

3.2. The component market

To procure components to build PCs, agents must negotiate with eight suppliers whose behavior is defined in the game specification. Each supplier sells two distinct grades of a single component type. CPU components have a unique supplier, whereas all other types are provided by two suppliers. Suppliers have a limited daily production capacity for each component they sell. These capacities vary according to a mean-reverting random walk, and are not directly observable by the manufacturers.

Manufacturers negotiate with suppliers via an RFQ mechanism. Each day, a manufacturer may send up to five RFQs per component to each supplier specifying type, due date, quantity, and reserve price. Any future day may be specified as the due date. The suppliers respond the following day with offers specifying a proposed quantity, price, and delivery date.[2] Suppliers may opt to accept a subset of the offers by responding with orders. Suppliers determine offer prices based on the ratio of their committed and available production capacity. This pricing function accounts for inventory, previous commitments, future capacity, and the demand represented in the current set of customer RFQs. In general, component prices rise as demand rises and suppliers become more constrained. In some cases, the supplier may be capacity constrained and unable to offer components at any price. Prices depend on both the requested due date and the day the request is made. Price predictions must be made for this entire space of possibilities to support important decisions about the optimal time to make component purchases.

3.3. Observable market data

Agents may utilize several complementary forms of evidence to make predictions about market prices. Each provides unique insights into current activity and how the market may evolve. The first category of evidence comprises observations of the markets during a game instance. This information is the most timely and specific to the current situation, but is of relatively low fidelity. More detailed observations of market activity are available in game logs after each game instance has completed.[3] Logs from previous rounds are publicly available, and additional data may be generated in private simulations.

During a game, agents receive daily observations from participating in the PC and component markets. In the PC market, agents observe the set of requests and whether or not each bid resulted in an order. They also receive a daily report for each PC type listing the maximum and minimum selling price from the previous day. The details of individual market transactions and opponents' bids are not revealed. In the component market, agents receive price quotes for each request they send, some of which may be price probes. Neither the supplier's state (inventory, capacity, and commitments) nor other agents' requests are revealed. Every 20 simulation days, manufacturers receive a report containing aggregate market data for the preceding 20-day period. This data includes the mean selling price and quantity sold for each PC type, and the mean selling price, quantity ordered, and quantity shipped for component type. The report also includes the mean production capacity for each supply line. The importance of this evidence is that it is specific to the current market conditions and current set of manufacturing agents. However, these observations say relatively little about how market conditions are likely to evolve, especially if similar conditions have not been observed earlier in the game.

The historical logs reveal the entire transaction history in each market (including all bids, offers, and orders), as well as supplier capacities and other state attributes hidden during a game instance. This precise information is potentially very useful for predicting the evolution of market prices over time. However, conditions in the current game are unlikely to be identical to conditions in historical games. A particularly difficult issue is that the pool of competing agents is constantly changing as teams update their agents, so historical data may not be representative of the current strategic environment. Selecting relevant historical data is an important challenge.

[1] We present an overview of the game rules here. The full rule specifications for each competition, tournament results, and general TAC information are available at http://www.sics.se/tac.

[2] Suppliers may offer either a partial quantity or a later delivery if they are unable to meet the initial request due to capacity constraints.

[3] A rule change in 2007 explicitly disallows agents from collecting and utilizing these logs during a tournament round, but logs from previous rounds or simulations may still be used.

Table 1
Four categories of predictions made in the Prediction Challenge.

Prediction	Description
Current PC prices	Lowest price offered by manufacturers for each current RFQ
Future PC prices	Median lowest price offered for each PC type in 20 days
Current component prices	Offer price for each component RFQ sent on current day
Future component prices	Offer price for each component RFQ sent in 20 days

3.4. The Prediction Challenge

The 2007 TAC event introduced two challenge competitions to complement the original TAC/SCM scenario. In the TAC/SCM Prediction Challenge (Pardoe, 2007), agents compete to make the most accurate price predictions in four categories: short- and long-term, for both component and PC markets (see Table 1). The challenge isolates the prediction component of the game by having participants make predictions on behalf of the same designated agent, PAgent. Simulations including a PAgent are run before the challenge to generate log files. During the challenge, the log files are used to simulate the experience of playing the game for the predictors by providing the exact sequence of messages observed by PAgent during the game. The predictions made based on this information by competitors in the challenge do not affect the behavior of the PAgent. This simulation framework allows all predictors to make an identical set of predictions, and also allows predictions to be made many times for the same set of game instances. The predictions are scored against actual prices using the root mean squared (RMS) error, normalizing by component and PC base prices.

Each round of the challenge used three sets of 16 game instances, with different agents competing with PAgent on the supply chain. The agents comprising the competitive environment were selected from those in the public agent repository.[4] Since tests can be run independently and repeated, the software framework for the challenge also provides a useful toolkit for conducting one's own prediction tests. We took advantage of this in developing and testing new prediction methods for the component market in preparation for the 2007 TAC/SCM tournament and Prediction Challenge.

4. Methods for PC market predictions

Each day, Deep Maize estimates the underlying demand state in the customer market and creates predictions of the price curves it faces.[5] For the current simulation day, the set of requests is known. Estimating the price curve in this case is equivalent to estimating the probability that a given bid on an RFQ will result in an order, denoted Pr(win|bid). To predict the price curve for future simulation days, it is also necessary to predict the set of request that will be generated. We combine predictions about the set of requests and the distribution of selling prices for each request (i.e., Pr(win|bid)) to estimate the effective demand curve for each future day. The effective demand curve represents the expected marginal selling prices for each additional PC sold. The agent uses these forecast demand curves to make customer bidding decisions and create a projected manufacturing schedule. We begin by discussion of our methods for estimating a key element of the game state, and proceed with a discussion of our methods for predicting customer market prices.

4.1. Customer demand projection

The customer demand processes that determine the number of RFQs to generate exert a great influence on market prices in the SCM game. There are three stochastic demand processes, corresponding to the respective market segments. Each process is captured by two state variables: a mean Q and trend τ. The number of RFQs generated in each market segment on day d is determined by a draw from a Poisson distribution with mean Q. The initial demand levels for each segment are drawn uniformly from known intervals. The mean demand evolves according to the daily update rule

$$Q_{d+1} = \min(Q_{\max}, \max(Q_{\min}, \tau_d Q_d)). \quad (1)$$

Both the minimum and maximum values of Q are given for each market segment in the specification. The trend τ is initially neutral, $\tau_0 = 1$, and is updated by the stochastic perturbation $\epsilon \sim U[-0.01, 0.01]$:

$$\tau_{d+1} = \max(0.95, \min(1/0.95, \tau_d + \epsilon)). \quad (2)$$

When the demand Q reaches a maximum or minimum value, the trend resets to the neutral level.

The state Q and τ are hidden from the manufacturers, but can be estimated based on the observed demand. We estimate the underlying demand process using a Bayesian model, illustrated in Fig. 3.[6] Note that the updated demand, Q_{d+1}, is a deterministic function of Q_d and τ_d, as specified in Eq. (1). This deterministic relationship simplifies the update of our beliefs about the demand state given a new observation. Let $\Pr(Q_d, \tau_d)$ denote the probability distribution over demand states given our observations through day d. We can incorporate a new observation $\widehat{Q}_{d+1}$ as follows:

$$\Pr(Q_d, \tau_d | \widehat{Q}_{d+1}) \propto \Pr(\widehat{Q}_{d+1} | Q_d, \tau_d) \Pr(Q_d, \tau_d). \quad (3)$$

Eq. (3) exploits the fact that the observation at $d+1$ is conditionally independent of prior observations given the demand state (Q_d, τ_d). We can elaborate the first term of the RHS using the demand update (1):

$$\Pr(\widehat{Q}_{d+1} | Q_d, \tau_d) = \text{Poisson}(\min(Q_{\max}, \max(Q_{\min}, \tau_d Q_d))).$$

After calculating the RHS of Eq. (3) for all possible demand states, we normalize to recover the updated probability distribution. We can then project forward our beliefs over (Q_d, τ_d) using the demand and trend update Eqs. (1) and (2), to obtain a probability distribution over (Q_{d+1}, τ_{d+1}).

A sequence of updates by Eq. (3) represents an exact posterior distribution over demand state given a series of observations. To maintain a finite encoding of this joint distribution, we consider only integer values for demand and divide the trend range into T discrete values, evenly spaced (Deep Maize uses $T = 9$ for tournament play). Given a distribution over demand states, we can project this distribution forward to estimate future demand.

4.2. Learning prices from historical data

We use a k-nearest-neighbors (k-NN) algorithm to induce the relationship between indicators of market conditions (features) and likely distributions of current and future prices. The idea is

[4] The TAC agent repository http://www.sics.se/tac/showagents.php stores binary implementations of many agents from past TAC tournaments, released by their developers.

[5] The methods described in this section were developed for the 2005 TAC/SCM tournament, and used with minor updates and new data sets in the 2006 competition. The version used in the 2007 tournament and Prediction Challenge were identical to the 2006 version (including the data sets).

[6] We have released source code for this estimation method. It is available in the TAC agent repository.

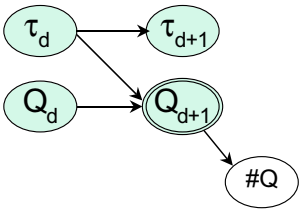

Fig. 3. Bayesian network model of demand evolution.

simple: given a set of games, find situations that most closely resemble the current situation and predict based on the observed outcomes. When prices are not changing much from day to day, learning prices from historical data has little additional benefit over simply estimating the prices in the current game instance. However, when prices are changing, historical data may help the agent to predict how the prices will move based on features of the current situation.

For each PC type and day, we define the state using a vector of features $F = (f_0, \ldots, f_n)$.[7] The difference between two states f^1 and f^2 is defined by weighted Euclidean distance:

$$d = \sqrt{\sum_{i=0}^{n} w_i \cdot (f_i^1 - f_i^2)^2},$$

where w_i is a weight associated with each feature. Additionally, we bound the maximum distance for any individual feature, so $d = \infty$ if $f_i^1 - f_i^2$ is greater than the bound. To predict the prices during a game, the agent first computes the current state features. It then finds the k state vectors in a database of historical game instances that are most similar to the current state, using d as the metric. To ensure diversity in the data set, we select at most one of the k vectors from each historical game instance. Associated with each state in the historical database are the observed price distributions in that game instance. We average these for the k observations to predict prices in the current game. Preliminary experiments varying the value of k showed the best results for values between 15 and 20; we use a value of 18 for all competitions and experimental results presented below.

The potential features we experimented with included the current simulation day, statistics about current prices, and estimates of underlying supply, demand, and inventory conditions. Many of these were motivated by analysis showing that related measures were strongly correlated with customer market prices in previous competitions (Kiekintveld et al., 2005). We tested a range of possible features, weights, and bounds in preliminary experiments, and selected those which had the lowest prediction error after extensive but ad hoc manual exploration. All of the following results use equal weights and bounds. For predicting the current days' prices, we selected the following features: average high and average low selling price from the previous five days, estimated mean customer demand (Q), the simulation day, and the linear trend of selling prices from the previous ten days. For predicting future prices, we used the same set of features with the addition of the estimated customer demand trend (τ).

4.2.1. Data sets

We employ two distinct data sets for making predictions, one based on tournament games and one based on internal simulations. The tournament data set contained games from earlier rounds in the tournament. As new rounds were played, we added the additional games to the training set. During the 2005 tournament we automatically added new games within a round to the data set; this was disallowed by the 2007 rule change mentioned above.

The second data set consisted of games run using six copies of Deep Maize. We played a sequence of roughly 100 games, initially using just the tournament data set to make predictions. After each game, we replaced the oldest log in the data set with the new one, so eventually the data set consisted entirely of games played by the six Deep Maize agents. The final 45 games of this sequence were selected as the 'self-play' data set. Ideally, as the data sets contain more data from this specific environment, the predictions will come to reflect the true market prices and agents' behavior will improve. In the limit, we can imagine agents converging to a competitive equilibrium, which may be closer to the environment of the final round than games from less competitive rounds early in the tournament. As it happened, our analysis indicates that predictions based on this data set were not very accurate.

4.3. Representing and estimating price distributions

We experiment with two different representations for the distribution of PC prices. One uses absolute price levels, and the other is relative to recent price observations. The advantage of absolute prices is that they translate readily across games and can easily represent changes in the overall level of prices. The relative distribution is centered on moving averages of the high and low selling price reports for the previous five days. Between these two averages we use 60 uniformly distributed points, above the average of the highs and below the average of the lows we use 20 points distributed uniformly. The strength of this representation is in its ability to exploit the fact that prices tend to be stable over short periods of time in the game (a few simulation days). It effectively shifts observations from historical games into the relevant range of prices for the current game, which is often small. When extracting data from historical games we adjust for the number of opponent agents by randomly selecting one agent and removing that agent's bid, if it exists. This accounts for the effects of the agent's own bid on the price; it is effectively learning the price that would be expected based on only the competitors' bids. We also smooth the observed distributions by considering all requests in the range $d - 10, \ldots, d + 10$.

To compute the estimated price distribution we take the average of the distributions stored for the k-nearest-neighbors, as described above. For each price point j, we take the predicted probabilities for this point in each of the historical instances and average them. For relative predictions, these points represent pricing statistics relative to current prices; for absolute, they represent absolute prices. After averaging, we enforce monotonicity in the distribution. Starting from the low end of the distribution, we take the maximum of the estimated probability for the price point and the estimated probability of the immediately preceding price point.

To forecast prices i days into the future, we project based on what happened i days into the future in each historical game. The agent requires predictions for all future days, so using this method requires that the historical instances have at least as many days remaining in the game as the current game does. We enforce the tighter constraint that the historical predictions must start on exactly the same simulation day. For relative price predictions, we also account for changes in the overall price levels over time by modifying the pricing statistics. For each price statistic, we compute the average change over the intervening i days in the historical data. Pricing statistics for the future days are modified by adding the average change. The absolute distributions do not need

[7] When initially computing the values of features, we normalize the values for each feature by the standard deviation of the feature in the training set.

to account for this, since the price points are based on absolute values.

4.4. Online learning

We employ an online learning procedure that optimizes predictions according to a logarithmic scoring rule.[8] First, we produce baseline predictions $P_{\text{tournament}}$ and $P_{\text{self-play}}$ based on the two data sets described in Section 4.2.1. We then construct adjusted predictions of the form

$$a(b \cdot P_{\text{tournament}} + (1-b) \cdot P_{\text{self-play}}) + c, \tag{4}$$

where b weights the two prediction sources based on their accuracy in the current environment, and the coefficients a and c define an affine transformation allowing us to correct for systematic errors.

We determine the values of a, b and c using a brute-force search to find the set of values (from a discretized range) that minimizes the scoring rule. The possible transformations are scored using the recent history of bids won and lost, a valuable but otherwise unused source of information. Each possible set of values is scored on the measure $-\sum_{\text{bids}} \log(\alpha_{\text{bid}})$ where α_{bid} is the magnitude of the difference between the predicted probability of winning and the result of the bid, which is 1 if the bid resulted in a win and 0 otherwise. To prevent very large changes in the prediction we loosely bound the possible values of a, b and c. This optimization is performed using the actual bids submitted by the agent in recent days, and modifies the distribution only within the range of prices where bids are made. This typically leaves the extremes of the distribution (very high or low probability bids) unchanged, since there is little bidding activity at these extremes.

5. Evaluation of PC market predictions

We evaluate our forecasting techniques using both prediction error and supply chain performance. We evaluate several variations of the predictor that employ subsets of the functionality described in Section 4 to assess the contributions of each. As a baseline we use a heuristic predictor described by Pardoe and Stone (2005). The baseline predictor uses only information contained in the current price level, and relies completely on local price stability for predictive power. This is similar to what our predictor might look like using the relative representation without any historical data or online transformations. This predictor produces surprisingly accurate predictions despite its simplicity. The reason is that prices in the TAC SCM game tend to be quite stable for long periods, especially if the start and end game conditions are excluded.[9] One of the primary reasons we use this as a baseline is that we believe it is reflective of the default heuristic methods used by many of the agents to make decisions (especially in the early years of the competition). In particular, it makes use of only local information about prices in the current game, and does not attempt to project changes in price levels over time.

The baseline method predicts the distribution of PC prices using a weighted average of uniform densities between the low and high prices from the previous five days.[10] We project this into the future by assuming that the distribution of prices does not change and adjusting for the predicted change in the number of RFQs generated, as given by the model of customer demand described in Section 4.1.

We present results for the baseline and four variations of our k-NN-based predictor. These variations use either the relative or absolute representation, and either apply online affine transformations or not. All results use both the tournament and self-play data sets, which contain 95 and 45 game instances respectively. Regardless of whether overall affine transformations are applied we use the online scoring method to weight the predictions from these two sources. At the end of this section we briefly discuss the effects of varying the data sets. All error results are derived from the 14 clean finals games from the 2005 tournament.[11]

5.1. Prediction error

We discuss two different types of prediction error. The first is for predictions made about the probability of winning the current day's auctions, and the second is for predictions about the effective demand curve on future days.

As we discuss the prediction error results, it is important to keep in mind that there is a substantial amount of uncertainty inherent in the domain model; we do not expect any prediction method to achieve anywhere near perfect predictions. To demonstrate this and provide some calibration, we consider the process that generates the overall level of customer demand in the SCM game. The level of customer demand (number of requests generated each day) is one of the most important factors influencing market prices. Recall that we use a Bayesian model to track the mean customer demand and trend; this model is the theoretically optimal predictor of the process. We ran tests comparing this predictor to a naive predictor that predicts that the mean demand on all future days will remain exactly the demand observed today. The optimal Bayesian model improved on the naive model, but the difference was relatively small. The naive model had an RMS error of approximately 35, and the Bayesian model achieved an RMS error of approximately 30—a reduction on the order of 15%. Whereas this may seem a relatively small improvement in prediction quality, these small improvements can translate into substantial differences in overall agent performance.

5.1.1. Current day prediction error

Predicting the conditional distribution $\Pr(\text{win}|\text{bid})$ for the current day's RFQ set is an important special case because these estimates are used for making bidding decisions. It is also unique because it is the only day for which we have the actual set of customer RFQs available; for future days, there is substantial additional uncertainty since the quantities, PC types, due dates, penalties, and reserve prices are all unknown.

We measure RMS error over a range of possible prices (i.e., potential bid values). The error may vary over the distribution, and this allows us to present a more detailed picture of the errors each predictor makes. The range of values we consider is relative to the recent high and low selling prices in the game. We distributed 125 price points over the range from 10% below the average low selling price from the previous five simulation days, to 10% above the average of the high selling price over the same window. Bids towards the high and low ends of this distribution are very rare, since they are either very high or low relative to the market. Deep Maize's bids are centered around price point 62, with a large majority falling between price points 50 and 80.

[8] The logarithmic scoring rule is strictly proper, in that the optimal score is obtained only for the true probability. Scoring according to rules that are not proper (including RMS error) can lead to biased estimates.

[9] A linear regression on the average selling prices from one day to the next results in an R^2 value on the order of 0.99, so current prices are extremely predictive of prices in the near future.

[10] We use weights of 0.3 for the most recent two days, 0.2 for the middle day, and 0.1 for the oldest two days.

[11] In two games (3718 and 4254) the University of Michigan network lost connectivity and this team's agent was unable to connect to the game server for a large part of the game. This distorts the results for uninteresting reasons, so we omit these games.

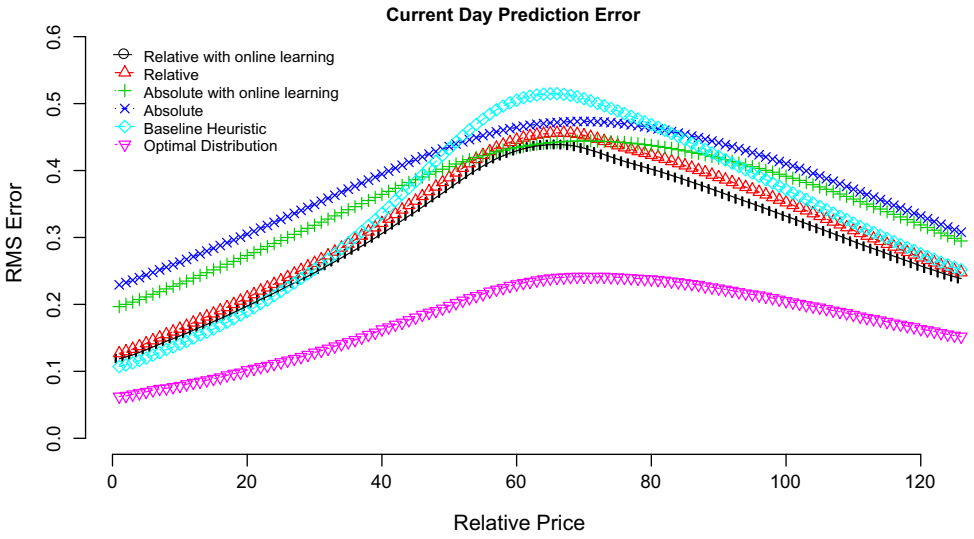

Fig. 4. RMS prediction error for current day predictions in the TAC/SCM final round games. The points on the horizontal axis represent a uniform distribution of points over a range defined by the recent high and low selling prices in the market.

To compute the error measure, we process each RFQ separately. For each of the 125 price points, we measure the error as the difference between the predicted probability and the actual outcome. The outcome is either 1 if the lowest bid was higher than the price point, or 0 if the lowest bid was lower than the price point (corresponding to whether a bid at that price would have won or lost the auction). A subtlety of measuring error based on individual RFQs in this way is that any probabilistic prediction that is not either 1 or 0 will, by definition, show error on this measure. To provide a benchmark for this effect, we also plot the error for the optimal distribution. This is a distribution derived from the actual prices for the PC type and day being measured, but without RFQ-specific features.

Fig. 4 shows the results of this error measure computed for each of the five candidate predictors, as well as the optimal distribution. We compute errors averaged over all games from the 2005 tournament final round, restricted to predictions made for the middle part of the game from simulation day 20 to 200. This restriction is primarily because the baseline predictor is not designed for border cases, such as when no recent price observations are available. The error of the baseline is artificially high in these cases, which would distort the overall error results in favor of our methods.

Of the candidate predictors, the relative predictor with online learning has the lowest overall prediction error. All four of our predictors have lower error than the baseline in the center part of the distribution. The two absolute predictors have greater error when making predictions for very high or very low prices. Online learning reduces prediction error across the entire distribution for both the absolute and relative representations. The relative predictors generally perform better than the absolute predictors. However, the absolute predictor with online learning has lower error than the relative predictor without online learning in the center of the price distribution. This is where online learning should have the greatest impact, since more bids are placed in this region and it has more data to make updates. The differences in error between our predictors and the baseline are quite substantial. In the center of the distribution, the difference between the baseline method and the optimal method is $\approx$0.3, and the difference between the optimal and the relative predictor with online learning is $\approx$0.2, an improvement of roughly 30%.

Conceptually, errors towards the center of the distribution are likely to be much more important because bids are centered there. However, is not clear how errors should be weighted, since bid densities may vary in different situations (and certainly for different agents). These predictions are also used for making non-bidding decisions, which may use the information in different ways. Using supply chain performance (as we do in Section 5.2 below) is one way to resolve this dilemma, since we can test the predictors as they are used by a real decision-making procedure.

5.1.2. Future forecasting error

A distinctive feature of Deep Maize is that it explicitly forecasts changes in future market conditions. For future days, the agent translates its predictions into an effective demand curve for use in production scheduling. This demand curve specifies the price the agent would need to bid to win a particular quantity of PCs, on average. These prices are given in an ordered list, with the first price representing the bid to win one PC, the second price the bid to win two PCs, and so on. The predictor gives the probability of winning a bid $\Pr(\text{win}_i|\text{bid})$, which we combine with the predicted overall demand to determine the expected quantity won for any given bid level. Inverting this function yields the desired prices.[12]

We measure the error for this effective demand curve, which combines errors from the price predictions and demand predictions. For each simulation day, we compute the actual demand curve by sorting the observed prices for each PC. We then measure the RMS error between the prices in this list and the predicted demand curve, summing the errors for each PC in the list up to a maximum of 200 values per day.[13] Fig. 5 compares the five predictors on this aggregate measure of future forecasting error out to a prediction horizon of 50 days.

The pattern of results is similar to the results for the current day. The two relative predictors score the best over all horizons, followed by the baseline and then the two absolute predictors. The online affine transformations are beneficial regardless of the representation used. We note that this measure averages error over the distribution (each data point represents a compression of the full distribution shown in Fig. 4). We present information in this way partially for easy visualization, but also because future

[12] Our functional form allows simple inversion of this function, but this inversion can also be computed using a simple binary search to find the necessary bid level.

[13] An agent typically projects selling 100 or fewer PCs of any type on a given day, so this is roughly twice the number of values the agent actually uses to make decisions.

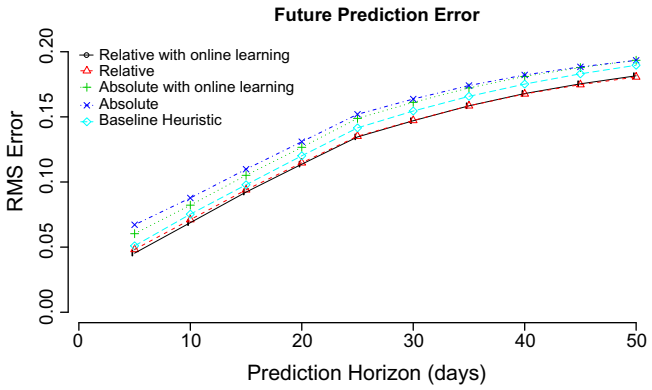

Fig. 5. RMS error between predicted and actual effective demand curves out to a horizon of 50 days, computed on the 2005 finals.

planning may be more dependent on errors over the entire distribution than bidding decisions. However, we should note that the absolute predictor shows higher overall error on this measure primarily because of errors at the extreme parts of the distribution, as in the short-term error results.

5.2. Supply chain performance

Since our forecasting methods were specifically designed to support decision-making in a fully implemented and automated agent, we have the opportunity to assess how forecasts impact agent performance. This is particularly interesting because of the complex interactions between prediction error and decisions. As discussed above, it can be difficult to determine how to weight different types of prediction errors. Another issue with online predictions is that the decisions made affect the information available for making future predictions.

We ran simulations using a profile of two MinneTAC-05 agents, two TacTex-05 agents, one static version of Deep Maize, and one modified version. MinneTAC-05 and TacTex-05 were finalists in 2005 that were among the first to release agent binaries. We ran four sets of games with a minimum of 35 samples. The static version of Deep Maize used the relative affine predictor. We present results as differences between the static and variable versions of Deep Maize. This pairing reduces variance but introduces bias to the extent that varying the sixth agent affects the performance of the static version. Another important caveat is that these results are for a single profile of strategies. More systematic approaches to selecting test environments may be better justified on game-theoretic grounds (Jordan et al., 2007), but we make do with a single convenient albeit ad hoc profile here.

In addition to scores we give three measures of customer sales performance: average selling price (ASP), "selling efficiency", and "timing efficiency". Selling efficiency is the fraction of achieved revenue to the maximum possible revenue for identical daily sales, given perfect information about opponents' bids and the option to partially fill orders. Timing efficiency measures how well the agent distributed sales over time. Let the t-optimal same day selling price be the highest ASP the agent could achieve by selling each PC at most t days after it was actually delivered and no earlier than it was actually produced. PCs are labeled according to a FIFO queuing policy. Timing efficiency is the ratio of the t-optimal same day selling price to the selling efficiency. This factors out the effects of bidding policy and leaves the effects of deciding which days to sell on. The appeal of these additional metrics is that they allow us to compare a specific element of decision performance against an optimal value, though we must remain cognizant of the additional constraints when interpreting the results.

Table 2
Performance measures in simulated games. The numbers represent the difference between two versions of Deep Maize using the relative affine predictor and the indicated predictor.

Predictor	Score	ASP	Selling efficiency	20-Day timing
Relative, no affine	−2.3 M	0.004	−0.007	−0.015
Absolute, affine	0.3 M	0.001	−0.008	0.000
Absolute, no affine	−1.2 M	0.007	−0.014	−0.009
Baseline	−9.1 M	−0.032	0.002	−0.032

The simulation results are in Table 2. The most striking point is that all of our k-NN-based predictors comfortably outperform the same agent using the baseline predictor. To calibrate, the difference between the top and bottom scores in the 2005 finals was ≈6.5 M, less than the advantage of any of our predictors over the baseline. Clearly, effective prediction can have a sizable impact on agent performance when the decision-making architecture is capable of exploiting the information. The affine transformations again showed benefits for both representations.

The absolute predictors scored better than the relative predictors, albeit by fairly small margins. This is somewhat unexpected, given the error results. Earlier results using a smaller data set for the predictors actually showed a slight advantage for the relative predictor. We expect a larger data set to benefit the absolute representation disproportionately because some of the features are based on current prices. With more data more similar instances are available, and the absolute representation should predict more like the relative representation. We conjecture that the relative representation may have greater advantages for smaller data sets, but have not verified this experimentally. Another likely explanation for this discrepancy is that the absolute representation better reflects the actual uncertainty present because it is able to spread predictions over a broader range of prices. This may cause the agent to delay making purchasing decisions until more information is available, potentially improving performance.

In general, the differences between the agent variations on the sales metrics are relatively small, but we note some interesting features. The timing measure correlates quite well with the overall scores. The baseline predictor performs very poorly on this measure as well as having a low ASP. The selling efficiency numbers do not show a strong pattern, and the ASP numbers for the k-NN variations are also ambiguous. This suggest that one of the more consistent benefits of better forecasts for Deep Maize is the usefulness of these forecasts in timing sales activity.

5.3. Tournament and self-play data

We also generated error scores for the four predictor variants using only the tournament data set and only the self-play data set (a total of 12 different variations). We do not present full data here, but can describe the pattern of results fairly easily. Using only the tournament data resulted in almost exactly equivalent performance to using the combined data sets. The results using only the self-play data were almost always worse than either the tournament only or combined settings. One conclusion we draw from this is that the process we used to generate the self-play data did not produce the most relevant experience for making predictions. Another is that the online scoring procedure was effective at detecting the source of more salient information to use in the combined predictor. It does this by placing a very high value on the variable b in Eq. (4), weighting the tournament data set much more heavily than the self-play data set. As the agent plays a game, we often observe values set to the maximum possible value, within the range of allowable values.

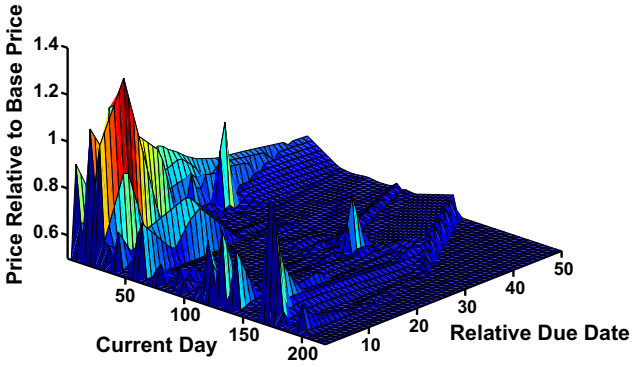

Fig. 6. Sample price predictions $p_{i,i+d}(1)$ for a single component where i is the current day and d is the relative due date.

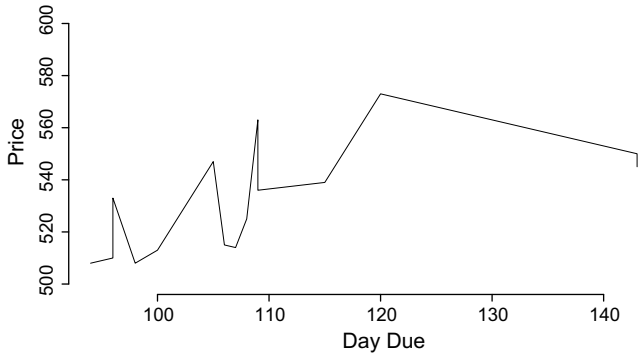

Fig. 7. A sample linear interpolation of recent price quotes.

6. Methods for component market predictions

As for the PC market, Deep Maize uses price forecasts to determine how to procure components in the supply market. Component price forecasts take the form $p_{i,j}^{(s,c)}(q)$, denoting the predicted offer price for a quantity q of component type c, requested from supplier s on day i, with due date j. The price predictions for each possible pair of request date and due date form a surface, as depicted in Fig. 6. These prediction are made each day for each supplier-component pair (s, c). This contrasts with the PC market, where we model a price curve with a single time parameter (the request date).[14]

Each day, manufacturers receive offers from suppliers in response to the RFQs sent the previous day. Information from these offers may be used as input to price predictions. Note that it is generally not possible to get a direct price quote for each future date, due to the limitation of five RFQs per day for each supplier and component combination. Thus, there is significant uncertainty regarding even the current price curve.

We consider three classes of prediction methods for the component market. The first creates an estimate of the current price curve using the observed spot price quotes and linear interpolation. This method does not attempt to predict changes in prices over time. The second approach augments the first by using market indicators to predict residual changes from the estimated price curve over time. This method exploits local price stability in the short term by using the linear interpolation model as a base, but still allows for modeling price variability over time. The final method applies regression to predict prices directly based on market indicators. This method allows for the greatest flexibility in predicting price changes.

6.1. Linear interpolation price predictor

As noted in many accounts of TAC/SCM agents, market prices in the game tend to exhibit local price stability over short horizons. The degree of volatility may vary depending on the particular agents playing and random variations in the underlying game state. However, there are structural reasons to expect such stability. One is that the underlying supplier capacity and customer demand levels change over time, but rarely exhibit drastic changes over short time periods. In addition, the outstanding commitments and inventory of suppliers and manufacturers tend to exert a damping effect on component price changes.

[14] We have found empirically that the due dates for PC requests have minor effects on the price, so we do not model these directly.

Deep Maize estimates the current market state based on price quotes from recent days. We employ linear interpolation to estimate prices for due dates with no recent observations. Fig. 7 shows an example price curve estimated using this method, which we term the linear interpolation price predictor (LIPP). Due to local price stability, we expect LIPP to be a reasonably good predictor for prices in the near future. Empirical results below support this proposition.

6.2. Relative component price predictor

In analyzing the performance of LIPP, we note a tendency for predictions over the entire price curve to systematically overestimate or underestimate prices. There are several plausible reasons for this. Recall that suppliers' current capacity varies according to a random walk during the game. Suppliers construct prices in part based on projecting their current production capacity into the future to determine their available capacity. Since available capacity is used in pricing for all request dates, fluctuations in capacity cause prices for all of these dates to rise or fall in unison. This pattern could also result from manufacturing agents increasing or decreasing order quantities, but maintaining approximately the same timing in their purchase schedules. This might result, for instance, from rising or falling customer demand levels.

LIPP does not take into account any information beyond recent price quotes that might help to predict these sorts of price fluctuations. We introduce the relative component price predictor (RCPP) as a way of exploiting these additional forms of information. RCPP predicts the residual error of LIPP by applying regression to technical attributes summarizing the observable market state. A list of the attributes we used is given in Table 3. The predictions made by RCPP are formed by calculating the summation of the residual errors and LIPP predictions.

The intuition for this approach is that the RCPP algorithm identifies game states that are likely to result in predictable price shifts, leading to inaccurate LIPP estimates. The residuals modify the current price estimates to account for these shifts. This approach exploits the short-term accuracy of the linear price predictor, while exploiting historical data and market indicators to adapt future forecasts. We note that there is a parallel between this approach and the relative representation used in predicting prices in the PC market.

6.3. Absolute component price predictor

The final class of predictors we consider is the absolute component price predictor (ACPP). ACPP applies regression to directly learn prices from the technical indicators. Unlike RCPP, this approach does not bootstrap the short-term accuracy of LIPP, so it

Table 3
Attributes considered by RCPP and their RReliefF values for current and future days. The first five attributes correspond to parameters of an individual RFQ and the remainder of market state.

Attribute	Current	Future	Description
Day	−0.000603	0.000842	Current day
Sent-day	−0.000603	0.000842	Day the RFQ will be sent
Due-day	0.000124	0.001038	Day the RFQ will be due
Duration	0.000209	0.000603	Due-day minus sent-day
Quantity	0.011297	0.013593	RFQ quantity
Linear-price	0.019816	0.010123	LIPP price prediction
Nearest-price-point-diff	−0.001689	−0.002899	Nearest known price point distance (days from due date)
Days-since-data	−0.006886	0.006228	Days since due date had a price quote
Market-capacity	−0.000496	0.00023	Last supplier capacity from market report
Estimated-capacity	−0.000317	0.001029	Estimate of supplier capacity using market reports and trend
Surplus	−0.000514	0.000157	Available capacity minus agent demand in recent period
Aggregate-surplus	−0.000618	0.000295	Available capacity minus agent demand in total
Mean-demand	−0.001663	0.000741	Mean customer demand for component
Mean-low-demand	−0.002198	0.001023	Mean customer demand for low-end components
Mean-mid-demand	−0.0014	0.0009	Mean customer demand for mid-range components
Mean-high-demand	−0.002	−0.000787	Mean customer demand for high-end components
5-Day-demand	−0.001258	0.000852	Customer demand for last 5 days
20-Day-demand	−0.000687	0.00164	Customer demand for last 20 days
40-Day-demand	−0.000833	0.0014	Customer demand for last 40 days

is not relative to the estimated current prices. The potential advantage of this approach is that it offers greater flexibility for modeling price shifts, since it is not constrained by the LIPP prediction. If there are prevailing long-term price trends not captured in the RCPP residuals, ACPP may yield more accurate predictions at long horizons. We note that there is a parallel between this approach and the absolute representation used in predicting prices in the PC market.

7. Evaluation of component market predictions

We evaluate the relative performance of the three classes of predictors on the basis of prediction error using the Prediction Challenge framework. We present the results of experiments run after the competition using the data set from the final round of the 2007 Prediction Challenge.[15] The data consists of a total of 48 games containing the designated PAgent, divided into three sets of 16 games played against the same set of opponents. All opponents were selected from the agent repository. The predictors are scored using normalized RMS error on all required component price predictions.

7.1. Setting expiration intervals for LIPP

We begin by exploring an important parameter of the LIPP method, which we then fix for subsequent analysis. Recall that manufacturers are limited in the number of RFQs they may submit each day. We can increase the number of observations available for estimating the price curve by including RFQ responses from previous days, with the caveat that these quotes may represent stale information. We represent this tradeoff using an expiration interval that determines how many days of previous observations are used to construct the LIPP estimate. Using a longer interval increases the amount of data available, along with the risk that the data no longer accurately reflects the current situation. To determine the best setting of the expiration interval, we tested intervals from one to five days. The results are shown in Fig. 8.

The mean accuracy of the LIPP predictions decreases monotonically as the size of the length of the expiration interval is increased. This result implies that the day-to-day price volatility in TAC/SCM is high enough that the benefit of having data points for additional due dates is outweighed by the reduced accuracy of the older data points. It is possible that adapting this expiration interval dynamically during a game based on observed price volatility or other factors could yield more accurate predictions, but we have not yet fully explored this possibility.

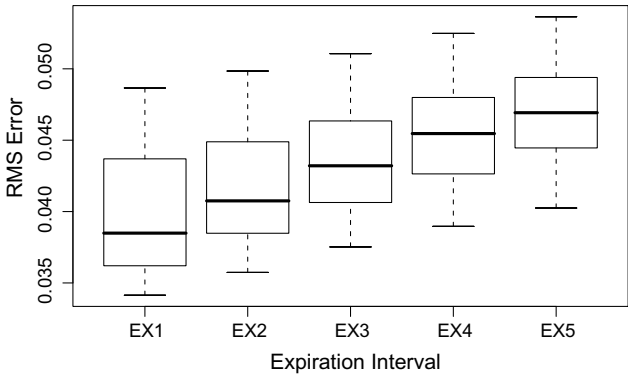

Fig. 8. RMS error for LIPP with various expiration intervals.

7.2. Estimating attribute quality for regression models

As a preliminary step to regression, we seek to evaluate the quality of various attributes that may be used to predict current and future component prices. Unlike classification problems, there are few measures of attribute quality available from the current literature. For example, information gain, *j*-measure, and χ^2 cannot be used for estimating attribute quality in a regression. Three measures that have been proposed for regression with numerical attributes are mean absolute and mean squared error (Breiman et al., 1984), and RReliefF (Robnik-Šikonja and Kononenko, 1997). Because the error measures assume independence among attributes, they are less appropriate given strong interdependencies like those introduced in our analysis. In contrast, RReliefF has be shown empirically to perform well in situations where there are strong interdependencies, such as parity problems. A short explanation of the RReliefF values is rather elusive (see related work Robnik-Šikonja and Kononenko, 2003 for a thorough exposition), but we attempt here to give some intuition as to what the values represent.

[15] Many of these analyses were also performed before the Challenge using other test sets; we repeat all of the experiments using this data set for consistency.

RReliefF is an extension of the ReliefF algorithm to regression. In classification, ReliefF measures an attribute's quality according to how well a given attribute distinguishes instances that are similar in the attribute space. The value itself is the approximated difference of two probabilities with respect to a given attribute value. The first is the probability that a different value is observed given near instances from different classes. The second is the probability that a different value is observed given near instances from the same class. Because price predictions and, in general, regressions are continuous, we cannot calculate these probabilities. RReliefF attempts to solve this problem by introducing a probability distribution which determines whether two instances are different. The values calculated by RReliefF lie in the range $[-1, 1]$ with random attributes tending to be slightly negative. At the extremes, a value of near one signifies that a different attribute value implies a different regression value for nearby instances. A RReliefF value greater than zero indicates that the first probability subsumes the second and thus the attribute is likely to discriminate nearby instances to a useful extent. We adopt this simple threshold to determine whether attributes are included in our regression models. Our experience is consistent with the empirical findings that RReliefF attribute selection improves our regression models over the default model that includes all available attributes.

A complete list of all attributes considered for regression analysis as well as the RReliefF values for each attribute on both current and future day predictions is given in Table 3. The attributes are a mix of RFQ parameters and estimated market state. Note that RReliefF values often differ significantly with respect to current and future predictions. In particular, the table suggests that current predictions are best determined by the attribute subset quantity, due-day, duration, and linear-price, all but the last of which are RFQ parameter attributes. This differs notably from future predictions, for which the majority of attributes are still of threshold quality. Component prices vary loosely according to a few market state parameters that are hidden from direct observation. Our attribute set was chosen in an attempt to extract useful observable parameters from the market in lieu of the hidden state. For current predictions, it appears that the market state attributes are not very useful in regression. This is likely explained by the observation that the internal market state changes gradually and that useful portions of the internal state are already expressed in the LIPP prices from the prior day. Thus, current prices can best be determined by attributes from the RFQ and LIPP prices, whereas including the estimates of market state is not beneficial. This is not to say that estimating internal market state is without predictive power in all cases. As the RRefliefF values in the future column suggest, the future market state is no longer reflected in LIPP prices and including these attributes is likely beneficial.

7.3. Comparing regression methods for RCPP

There are many machine learning algorithms that could be applied to learn the residual errors for RCPP. We tested a variety of regression algorithms from the Weka machine learning package (Witten and Frank, 2005) for this task, including linear regression, *k*-NN, support vector machines, and various decision-tree regressions. We generated training data from approximately 50 game instances simulated on a local computing cluster. Each game of training data had one PAgent playing, along with a collection of other agents taken from the agent repository (the same set of potential opponents used for the Prediction Challenge). Each game contains approximately 1800 sample price predictions for each CPU component and twice as many for non-CPU components. We train separate predictors for each component type, so we have approximately 90,000 data points for CPU components and 180,000 for non-CPU components. We take samples from the viewpoint of the PAgent in the training games, so the market indicators are computed with respect to this agent's observations.

After some preliminary experimentation, we selected additive regression[16] and reduced-error pruning trees (REP trees) as the most promising candidates. Many of the other algorithms had similar results in terms of prediction error, but required much longer to train or generated very large models. For some of the algorithms, overfitting was a severe problem. We tested several parameter settings for both additive regression and REP trees using the data set from the final round of the Prediction Challenge. Additive regression was tested both with default settings and with a shrinkage rate of 10%.[17] REP Trees were tested with maximum tree depths of four through seven. All variations learned residuals from LIPP predictions with an expiration interval of one (the best setting from the previous section).

Fig. 9 shows the prediction error results for these methods on both current and future predictions. The different learning algorithms produced very similar results; each would have placed first in both component prediction categories. For the actual challenge, we used the REP tree algorithm with maximum depth five to learn residuals for RCPP, since this algorithm was consistently accurate for both current and future predictions.

7.4. Comparison of prediction methods

In the previous sections, we explored variations of the LIPP and RCPP methods to identify good parameter settings for these algorithms. We now directly compare the performance of the three different classes of methods introduced in Section 6. We use the LIPP method with expiration interval one, the RCPP method learning residuals using REP trees of max depth five, and an ACPP predictor that uses additive regression to make price predictions directly from the technical indicators.[18] Prediction error results for all three algorithms on both current and future predictions are given in Fig. 10. All current comparisons in the figure are statistically significant beyond the 0.005 level and all future comparisons in the figure are statistically significant beyond the 0.05 level using a standard *t*-test.

For the current price predictions, the performance for LIPP and RCPP is very close, with both outperforming the absolute predictor by a wide margin. This is strong evidence that the LIPP method effectively summarizes the current price state, and that this is a very good predictor of prices in the near future due to local price stability. Interestingly, none of the methods we tried for learning residuals were able to improve LIPP predictions by a significant margin. This suggests that in the near term, most of the information about market conditions is effectively summarized by the current prices.

The benefits of learning prices from historical data using technical indicators is evident in the results for predicting future prices. Both ACPP and RCPP perform better than the LIPP method in this case, demonstrating the value of representing changes in prices over time. RCPP performs better than ACPP on these future predictions, indicating that the current prices still have significant predictive value, even at a 20-day horizon. Overall, the RCPP method, which combines the price estimates from linear interpolation with learned residuals, offers the best of both worlds; it retains short-term accuracy while significantly improving long-term forecasts.

[16] Additive regression is a meta-learning algorithm wherein each decision stump is fit to the residual of the previous step every iteration.

[17] The shrinkage rate considers how much of the trained decision stump is used each iteration. Normally, the shrinkage rate is set to 100%, however, a smaller shrinkage rate induces a smoothing effect and reduces overfitting.

[18] Based on the results in the previous section, we would expect a REP-tree version of ACPP to have very similar performance.

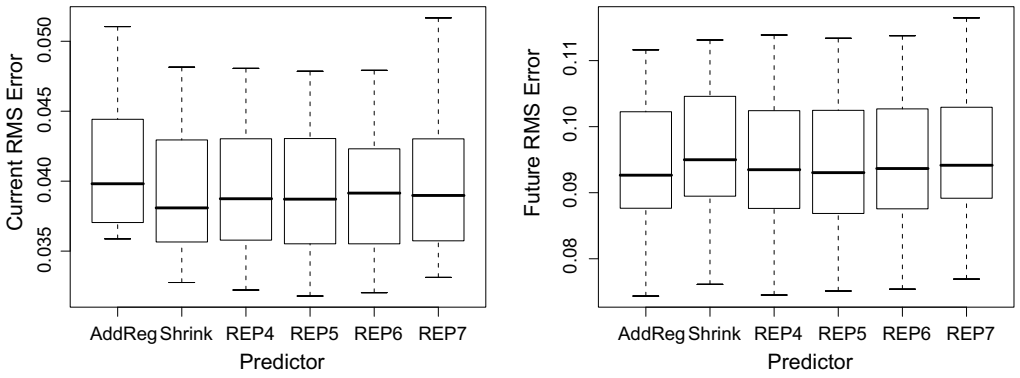

Fig. 9. Prediction error results for various candidate learning algorithms on the data set from the final round of the Prediction Challenge. Future corresponds to a 20-day horizon.

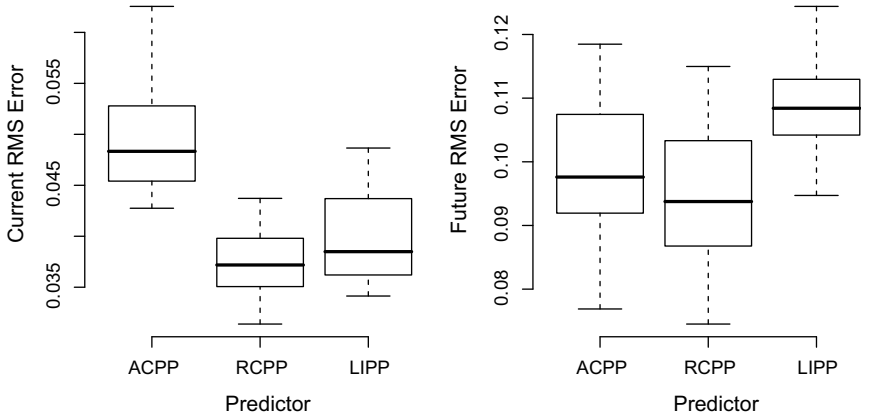

Fig. 10. Prediction errors for the absolute (ACPP), relative (RCPP), and spot (LIPP) predictors on the Prediction Challenge final round data set. Future corresponds to a 20-day horizon.

We selected RCPP for use in both the Prediction Challenge and the main tournament.

7.5. Supply chain performance

In this section, we analyze the effects of component prediction methods on aggregate agent performance, similar to the analysis performed on the customer prediction effects described in Section 5.2. We ran simulations using a profile of TacTex-07-Finals, Phant-Agent-07, Mertacor-05, MinneTAC-06, and two versions of Deep Maize. One version of Deep Maize stayed static and was used as the baseline agent. In this case, the baseline agent was Deep-Maize-07-Finals, which used RCPP with LIPP5 for its component predictions. The second Deep Maize agent competing in the simulations used either LIPP1, ACPP, or RCPP with LIPP1 for its component predictions. To evaluate each regression method, we ran three sets of games, involving no less than 35 games in each set. We report the difference in score between the variation of the baseline using the indicated predictor and the baseline itself, as well as the *p*-value of the corresponding paired *t*-test. We found all pairwise comparisons to be significantly different at a level beyond 0.01. The results are presented in Table 4.

We expect that the relative performance of each variation of Deep Maize will correspond to the accuracy of the prediction method used by the agent. The results support this for current price predictions, but show less sensitivity to the accuracy of long-term predictions. The predictor which fared the best in the prior section, RCPP with LIPP1, was an $0.8 M improvement over the baseline agent, with the only difference being an expiration interval of one versus five in LIPP. Averaged over a large number of games, this is a substantial difference in a profile of competitive agents. Surprisingly, LIPP1 without RCPP was also an improvement over the baseline agent. This is interesting because LIPP1 predicts future prices simply using its current prediction with no adjustment. Thus, Deep Maize was able to perform quite well without making an effort to adjust for future component prices. This effect is also evident in the result for the ACPP version.

Table 4
Relative score with respect to the baseline agent in simulated games. The numbers represent the difference between two versions of Deep Maize, one using the indicated predictor and the other using the version played in the 2007 tournament finals (Deep-Maize-07-Finals). In addition the *p*-value of each paired *t*-test is shown.

Predictor	Relative score	*p*-value
RCPP with LIPP1	0.80 M	0.0013
ACPP	−13.36 M	<2.2e-16
LIPP1	0.79 M	0.0032

In the prior section, we noted that although ACPP performed poorly in current predictions, it fared well in future predictions when compared to LIPP. The poor performance in the current prices had drastic consequences for the overall agent score. The effect of the large errors in current prices caused Deep Maize to procure far fewer components and thus claim less PC market share, which exacerbated the relative score difference by opening up more profitable market share to the other agents. All three tests support the proposition that prediction accuracy exerts a noticeable effect on an agent's performance, especially when evaluating current price predictions.

8. TAC/SCM tournament and prediction challenge

The TAC/SCM tournament and Prediction Challenge offer distinct opportunities for evaluation of our prediction methods in comparison with alternatives outside our control. The Prediction Challenge facilitates direct comparisons with competing prediction approaches, isolating this aspect of the game. The general tournament results allow us to evaluate the general competency of our agent relative to the other competitors. Predictions are an important component of Deep Maize's decision-making, so the overall performance of the agent reflects in part the quality of these predictions. An important aspect of both the SCM tournament and the Prediction Challenge is that each new game instance may present novel strategic environments not be reflected in any historical data. Thus, this exercise serves as an important indication of the robustness of our prediction results.

8.1. TAC/SCM tournaments

Tournament performance is an important indicator of the overall competence of TAC/SCM agents. Each TAC/SCM tournament comprises a sequence of rounds: qualifying, seeding, quarter-finals, semi-finals and finals. The qualifying and seeding rounds span several days and are used primarily for development and testing. The quarter-finals, semi-finals and finals are held on successive days. In 2005 and 2006, 24 agents were selected to begin the quarter-finals, with half eliminated in each round. In 2007, 18 agents started the quarter-finals, with six eliminated in each round. Results from the 2005, 2006 and 2007 tournament semi-final and final rounds including Deep Maize are presented in Tables 5–7.[19]

Deep Maize has performed very well in each of the TAC/SCM tournaments to date. Our agent reached the final round each year, placing fourth in 2005, third in 2006 and third in 2007. Deep Maize also demonstrated strong performance in the semi-final rounds, placing first in each heat. This record of overall performance provides evidence that Deep Maize is effective in a wide variety of competitive environments. The version of the customer prediction module presented here is very similar to the modules that played in all the 2005–2007 tournaments. This aspect of the agent was largely unmodified between 2005 and 2007, though other elements of the agent underwent significant revisions.

In 2005 and 2006, Deep Maize used only LIPP5 for component market predictions. RCPP was added to the predictions for the 2007 tournament (using a REP Tree with a maximum depth five), however we still maintained an expiration interval of five for LIPP. Unfortunately, we did not discover the superior accuracy of lower expiration intervals until shortly before the finals, such that we could not make the modification and completely test the strategy. The training data used for 2007 tournament play was derived from approximately 100 game instances, including games from the late rounds of the 2006 tournament and additional simulations using the best available agents from the agent repository.

8.2. TAC/SCM 2007 Prediction Challenge

The introduction of the Prediction Challenge allows us to directly compare our prediction approaches to other competitors, supplementing our own analysis based on controlled experiments. The Prediction Challenge was played in two rounds, a semi-final and a final. The semi-final round took place during the same time-frame as the TAC/SCM seeding round, before the main tournament. Like the qualifiers and seeding rounds in the main TAC/SCM tournament, this round was meant as a preliminary test of the overall Prediction Challenge for both the organizers and competitors. The final round took place the day before the main TAC/SCM finals. In both rounds, every competitor was given eight hours to make predictions about the 48 games defining the round. Competitors were allowed to produce their predictions any time during this period, independently of the other competitors.

The results from the Challenge semi-finals and finals are shown in Table 8. Deep Maize placed first overall in the finals, achieving the top score in both component prediction categories and the second-best scores in both PC prediction categories. These results provide

[19] More details about the tournament structure and full results are available at the TAC web site, http://www.sics.se/tac.

Table 5
Scores and rankings from the rounds Deep Maize played in during the 2005 SCM tournament. All scores are in millions of dollars. The adjusted column eliminates two game instances (3718 and 4259) where the University of Michigan network lost connectivity during the game, distorting the results substantially.

Semi-Finals 2		Finals		Finals (adjusted)	
Deep Maize	3.68	TacTex-05	4.74	TacTex-05	5.18
TacTex-05	3.57	SouthamptonSCM	1.60	Deep Maize	2.06
MinneTAC	2.27	Mertacor	0.55	SouthamptonSCM	1.55
RationalSCM	−2.28	Deep Maize	−0.22	Mertacor	0.53
CMieux	−2.33	MinneTAC	−0.31	MinneTAC	0.33
PhantAgent	−6.64	Maxon	−1.99	Maxon	−1.74

Table 6
Scores and rankings from the rounds Deep Maize played in during the 2006 SCM tournament. All scores are in millions of dollars.

Semi-finals 2		Finals	
Deep Maize	6.45	TacTex-06	5.85
Maxon	4.08	PhantAgent	4.15
Botticelli	1.95	Deep Maize	3.58
CMieux	1.81	Maxon	1.75
SouthamptonSCM	1.63	Botticelli	0.48
Mertacor	1.44	MinneTAC	−2.70

Table 7
Scores and rankings from the rounds Deep Maize played in during the 2007 SCM tournament. All scores are in millions of dollars.

Semi-Finals 1		Finals	
Deep Maize	9.76	PhantAgent	8.67
Tinhorn	6.94	TacTex	6.31
Maxon	5.63	DeepMaize	5.45
MasterBham	5.40	Maxon	1.79
CrocodileAgent	5.12	Tinhorn	1.34
Sorter95	−2.65	CMieux	1.24

Table 8
TAC/SCM Prediction Challenge 2007 Results (RMS error normalized by base price).

Prediction	Agent	Semi-finals	Finals
Current PCs	TacTex	0.04812	0.04554
	Deep Maize	0.04632	0.04682
	Botticelli	0.09449	0.04714
	Kshitij	0.04882	0.04873
Future PCs	TacTex	0.09318	0.09156
	Deep Maize	0.09473	0.09586
	Botticelli	0.13984	0.10238
	Kshitij	0.15428	0.11094
Current components	Deep Maize	0.04810	0.03919
	Botticelli	0.04163	0.04168
	TacTex	0.04239	0.04284
	Kshitij	0.10654	0.13333
Future components	Deep Maize	0.09925	0.09427
	Botticelli	0.10067	0.09700
	TacTex	0.10714	0.10338
	Kshitij	0.16324	0.13886

additional support for the overall efficacy of our methods. Deep Maize used a variation of RCPP for the semi-final round of the Prediction Challenge. Among other differences, this version used an expiration interval of five for the linear price interpolations. The final round version used an expiration interval of one, which largely account for the significant improvement in current component price predictions between the semi-final and final round. The data set for component price predictions was based on games specifically simulated for the Prediction Challenge, as discussed in Section 7.3. The customer price predictor used in the Challenge was identical to the version used by Deep Maize in the 2006 TAC/SCM tournament, including the data set.

9. Conclusions

We document and analyze successful approaches for forecasting prices in a challenging market domain. The general principle of our design is to support exploitation of all possible sources of information. To this end, we integrate several well-known ideas in an novel way, resulting in better overall predictions.

In the PC market, we evaluate our ideas against a baseline predictor similar to those used by many competing agents, applying both error measures and simulation performance as benchmarks. We show that improved predictions can dramatically improve agent performance in TAC/SCM. This is particularly interesting in light of the substantial uncertainties inherent in the domain and challenges of predicting the behavior of other agents.

The technique we present for applying transformations to predictions using online scoring is particularly interesting, and should be applicable in many domains. This method provides a way to perform principled combinations of predictions based on different sources of data, as well as a way to reduce systematic errors. It does these by making use of additional information derived from online performance. This method showed consistent benefits in our experiments across a range of conditions on both prediction error and simulation performance metrics.

We also investigated the use of absolute and relative representations for predictions. It appears that each of these representations has advantages under certain conditions, particularly related to how the information is used in decision-making.

In the component market we develop methods to integrate volatile spot price predictions with a long-term predictor using multiple sources of inter-market information. This relative predictor is more accurate than either the spot predictions or absolute price predictions from historical data in isolation.

Our prediction methods for both markets demonstrate strong overall performance, demonstrated in internal experiments and in our first-place finish in the inaugural TAC/SCM Prediction Challenge. These predictions have also contributed to the strong overall performance of Deep Maize in the TAC/SCM tournament in recent years.

Acknowledgements

Thanks to the participants and organizers of the TAC/SCM competition for providing such a fruitful research environment. We owe particular gratitude to David Pardoe for designing and operating the Prediction Challenge. Kevin O'Malley contributed to the Deep Maize infrastructure. This work was supported in part by NSF grant IIS-0205435, and the STIET program under NSF IGERT grant 0114368.

References

Bates JM, Granger CWJ. The combination of forecasts. Oper Res 1969;20(4):451–68.
Benisch M, Greenwald A, Naroditskiy V, Tschantz M. A stochastic programming approach to scheduling in TAC SCM. In: Fifth ACM conference on electronic commerce. New York; 2004. p. 152–9.
Benisch M, Sardinha A, Andrews J, Sadeh N. CMieux: Adaptive strategies for competitive supply chain trading. In: Eighth international conference on electronic commerce. Ann Arbor; 2006. p. 47–58.
Breiman L, Friedman J, Stone CJ, Olshen R. Classification and regression trees. Chapman & Hall/CRC; 1984.
Brockwell PJ, Davis RA. Introduction to time series and forecasting. Springer-Verlag; 1996.
Brooks CH, Gazzale R, Das R, Kephart JO, MacKie-Mason JK, Durfee EH. Model selection in an information economy: Choosing what to learn. Comput Intell 2002.
Burke DA, Brown KN, Hnich B, Tarim A. Learning market prices for a real-time supply chain management trading agent. In: AAMAS-06 workshop on trading agent design and analysis. Hakodate; 2006.
Chatfield C. Time-series forecasting. Chapman & Hall/CRC; 2001.
Cheng S-F, Leung E, Lochner KM, O'Malley K, Reeves DM, Wellman MP. Walverine: A Walrasian trading agent. Decis Support Syst 2005;39:169–84.
Clemen RT, Winkler RL. Combining probability distributions from experts in risk analysis. Risk Anal 1999;19(2):187–203.
Collins J, Arunachalam R, Sadeh N, Eriksson J, Finne N, Janson S. The supply chain management game for the 2007 Trading Agent Competition. In: Technical report CMU-ISRI-07-100. Carnegie Mellon University; 2007.
Eriksson J, Finne N, Janson S. Evolution of a supply chain management game for the Trading Agent Competition. AI Commun 2006;9:1–12.
Ghani R. Price prediction and insurance for online auctions. In: International conference on knowledge discovery in data mining; 2005. p. 411–8.
Hanson R. Combinatorial information market design. Inform Syst Front 2003;5(1):107–19.
Hastie T, Tibshirani R, Friedman J. Elements of statistical learning. Springer-Verlag; 2001.
He M, Rogers A, Luo X, Jennings NR. Designing a successful trading agent for supply chain management. In: Fifth international joint conference on autonomous agents and multiagent systems. Hakodate; 2006. p. 1159–66.
Jordan PR, Kiekintveld C, Wellman MP. Empirical game-theoretic analysis of the TAC supply chain game. In: Sixth international joint conference on autonomous agents and multiagent systems; 2007. p. 1188–95.
Kaufman PJ. Trading systems and methods. third ed. John Wiley and Sons; 1998.
Keller PW, Duguay F-O, Precup D. RedAgent-2003: an autonomous market-based supply-chain management agent. In: Third international conference on autonomous agents and multiagent systems; 2004. p. 1182–9.
Ketter W. Identification and prediction of economic regimes to guide decision making in multi-agent marketplaces. PhD thesis, University of Minnesota; January 2007.
Ketter W, Collins J, Gini ML, Gupta A, Schrater P. A predictive empirical model for pricing and resource allocation decisions. In: Ninth international conference on electronic commerce. Minneapolis; 2007. p. 449–58.
Kiekintveld C, Wellman MP, Singh S, Estelle J, Vorobeychik Y, Soni V, Rudary M. Distributed feedback control for decision making on supply chains. In: Fourteenth international conference on automated planning and scheduling. Whistler, BC; 2004. p. 384–92.
Kiekintveld C, Vorobeychik Y, Wellman MP. An analysis of the 2004 supply chain management trading agent competition. In: IJCAI-05 workshop on trading agent design and analysis. Edinburgh; 2005.
Kiekintveld C, Miller J, Jordan PR, Wellman MP. Controlling a supply chain agent using value-based decomposition. In: Seventh ACM conference on electronic commerce. Ann Arbor; 2006. p. 208–17.
Kiekintveld C, Miller J, Jordan PR, Wellman MP. Forecasting market prices in a supply chain game. In: Sixth international joint conference on autonomous agents and multiagent systems; 2007. p. 1318–25.
Nogales FJ, Contreras J, Conejo AJ, Espinola R. Forecasting next-day electricity prices by time series models. IEEE Trans Power Syst 2002;17:342–8.
Osepayshvili A, Wellman MP, Reeves DM, MacKie-Mason JK. Self-confirming price prediction for bidding in simultaneous ascending auctions. In: Twenty-first conference on uncertainty in artificial intelligence. Edinburgh; 2005; p. 441–9.
Pardoe D. The TAC SCM prediction challenge. 2007; <http://www.cs.utexas.edu/TacTex/PredictionChallenge/>.
Pardoe D, Stone P. Bidding for customer orders in TAC SCM. In: Agent mediated electronic commerce VI, lecture notes in artificial intelligence, vol. 3435. Springer; 2005.
Pardoe D, Stone P. TacTex-05: A champion supply chain management agent. In: Twenty-first national conference on artificial intelligence. Boston; 2006. p. 1489–94.
Pardoe D, Stone P. Adapting price predictions in TAC SCM. In: AAMAS-07 workshop on agent mediated electronic commerce IX. Honolulu; 2007a.
Pardoe D, Stone P. An autonomous agent for supply chain management. In: Adomavicius G, Gupta A, editors. Handbooks in information systems series: business computing. Elsevier; 2007b.
Pindyck RS, Rubinfeld DL. Econometric models and econometric forecasts. 4th ed. Irwin/McGraw-Hill; 1998.
Robnik-Šikonja M, Kononenko I. An adaptation of relief for attribute estimation in regression. In: Fourteenth international conference on machine learning. Nashville; 1997. p. 296–304.
Robnik-Šikonja M, Kononenko I. Theoretical and empirical analysis of ReliefF and RReliefF. Mach Learn 2003;53:23–69.

Stone P, Schapire RE, Littman ML, Csirik JA, McAllester D. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. J Artif Intell Res 2003;19:209–42.
Wellman MP, Reeves DM, Lochner KM, Vorobeychik Y. Price prediction in a trading agent competition. J Artif Intell Res 2004;21:19–36.
Wellman MP, Greenwald A, Stone P. Autonomous bidding agents: strategies and lessons from the trading agent competition. MIT Press; 2007.
Witten IH, Frank E. Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann; 2005.