Data-Mining-Enhanced Agents in Dynamic Supply-Chain-Management Environments

Kyriakos C. Chatzidimitriou and Andreas L. Symeonidis,
Aristotle University of Thessaloniki

Data mining analysis can benefit a well-established agent supply-chain-management network, both at a macro and micro level.

As agent technology matures over time, autonomous agents gain applicability and trust in trading and auctioning goods in real-world electronic markets as well as in managing more complex environments such as supply-chain networks.

A typical supply chain comprises various entities, such as suppliers, manufacturers, distributors, and customers, and its functionality is characterized by an ensemble of parallel and continuous operations at various levels (strategic, tactical, and operational). Supply-chain entities strive to optimally manage the flow of goods (downstream) from raw materials to end products, in accordance with the information flow (upstream) from demand to supply, with the overall goal to reduce costs, increase profits, and maximize service levels and customer satisfaction.[1]

Supply-chain management (SCM) is a challenging task because the environment is multiagent (cooperative and competitive), partially observable, dynamic, stochastic, and extremely complex. Changes in one link can cause ripple effects throughout the network. One such phenomenon is the well-known *bullwhip effect*, in which variation in demand increases up the chain, resulting in larger safety stocks and thus greater storage and product costs, with the list of causes and effects going on and on. This applies not only to the interorganizational domain but also to the intraorganizational range of operations; pricing and marketing mechanisms could also interfere with inventory management and procurement contracts.

In addition, efficient SCM implies successful handling of globalization, facilitation of mediation through the Internet, and shifts in production from technology and product-driven business processes to market and customer-driven manufacturing. Static and long-term relationships between partners have evolved to more dynamic trading schemes, in which interested parties auction goods and services among themselves. The more frequent the transactions, the more costly a policy (re)design. Thus, a policy that

1541-1672/09/$25.00 © 2009 IEEE
Published by the IEEE Computer Society

can semiautomatically adapt and respond to current market conditions is extremely appealing.[2]

Data mining (DM) offers an adaptation methodology[3] that can provide a predictive edge over competitors. Through DM, we can extract correctly evaluated models along various links in the supply chain, enabling us to generalize, detect abnormalities, and identify potential market opportunities. Coupled with agent technology (AT), which can provide real-time input and assessment of the DM-extracted models, decisions on the current state of the environment can be made in an automated manner. Specifically, in a supply-chain organization, experts can apply DM to various facets such as pricing, forecasting, and customer and supplier relationship management, always keeping in mind that these solutions should satisfy all security, safety, and soundness issues that might arise in such versatile environments.[4]

Data mining offers an adaptation methodology that can provide a predictive edge over competitors.

Trading Agent Competition

To investigate this AT–DM symbiosis in SCM environments, we used a generic testbed that simulates a supply-chain network in which we could apply and evaluate various methodologies. The Trading Agent Competition Supply Chain Management (TAC SCM) game allows for a huge strategy space, including time limitations and frequent transactions along with a competitive, dynamic, and stochastic environment that inhibits long-term policies, analytical studies, and the application of mathematical optimization techniques. In particular, we focused on auctioning, which, in general, defines pricing in B2B marketplaces.[2] We can easily extend our methodology to real-world SCM environments, given sufficient data are available.

Within the TAC SCM game,[5] agents act as personal computer (PC) manufacturers, competing with each other for supplier and customer contracts. A maximum number of six agents can connect to the TAC SCM game server, which simulates the suppliers and customers and provides banking, production, and warehousing services. Game length is 220 days, with each day lasting 15 seconds. Throughout the game, each agent must

- negotiate supply contracts,
- bid for customer orders,
- manage daily assembly activities, and
- ship completed orders to customers.

Agents run their own PC assembling unit, which has limited production capacity (2,000 factory cycles per day). They can assemble 16 predefined types of PCs, each one requiring a different component compilation, and procure the 10 different available components (CPUs, motherboards, memory, and hard disk drives) by sending requests for quotes (RFQs) and issuing orders to suppliers. The suppliers also have limited capacity, and because they simulate revenue-maximizing entities, component availability shouldn't be taken for granted. Each day, customers send RFQs, and agents bid on them, depending on their ability to satisfy delivery dates and prices. The bid price shouldn't exceed the reserve price the customer requires, which is between 75 and 125 percent of the PC components' nominal price. The next day, if an agent's quote is a winning offer (the lowest bid), the customer sends its order to the agent. To get paid, the agent must either assemble the ordered PCs on time or supply the customer with PCs already stocked in inventory. If an agent fails to deliver customers' orders, it's charged with a penalty. The agent with the greatest bank balance at the game's end wins. Figure 1 provides a schematic representation of the game,[6] while the game specifications offer a more detailed description.[7]

Data-Mining-Enabled SCM Agent Design

To accomplish its goal of maximizing profit, a generic SCM entity must tackle three major tasks, each closely related to the others: manage procurement, manage factory scheduling (for example, production and delivery scheduling), and manage sales by responding to customer RFQs.

In general, different approaches can lead to different architectures that take specific requirements into account and counterbalance task performance. Nevertheless, the cornerstone of an SCM agent's architecture is the design choice for the order fulfillment process. These are the most prominent:

- *Make-to-stock (MTS)*, in which agents build and stock products based on forecasts of customer demand. This practice is common in the retail industry.
- *Assemble-to-order (ATO)*, in which agents build the final product from a stock of components, but only after a customer has given the product configuration. This process assumes modular products, such as in the PC industry, where a customer might select the components online and then post the order.
- *Build-to-order (BTO)*, in which agents build each product only after a customer has placed a final order.

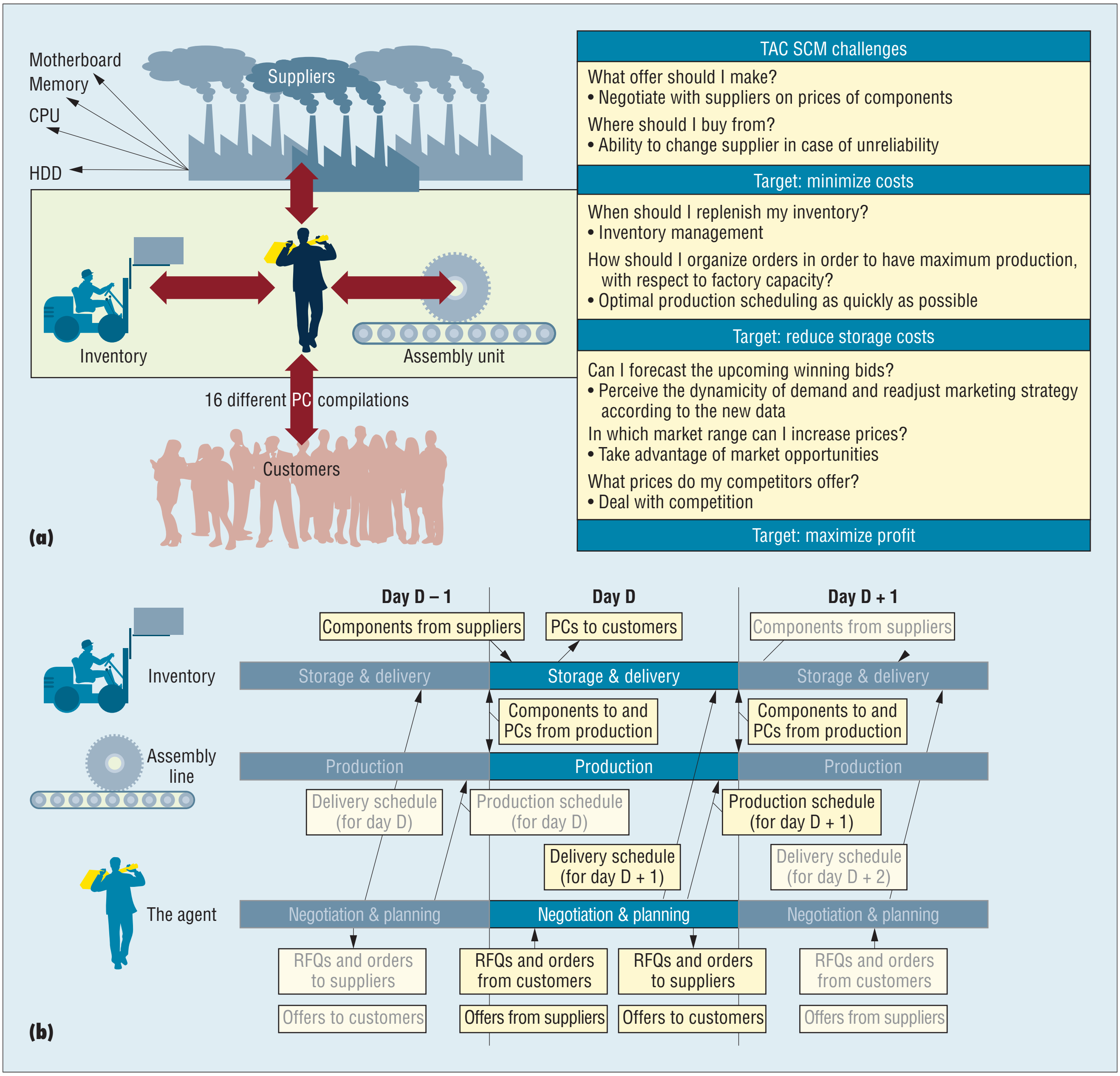

Figure 1. An overview of the Trading Agent Competition Supply Chain Management (TAC SCM) game. (a) A graphic summarizing the TAC SCM game, which involves agents acting as PC manufacturers. (b) Negotiations, assets, and messages exchanged daily during the competition.

- *Engineer-to-order (ETO)*, in which agents design and build a product according to customer requirements and specifications. Such an order fulfillment process is generally employed in large custom software and civil engineering projects.

Based on the requirements of the TAC SCM scenario and the current real-world practices of PC manufacturers,[1,8] ATO seems like the best choice for this industry. A major advantage of this order fulfillment design choice is the ability to utilize risk-pooling to reduce variation in demand and aggregate it over ranges of products instead of individual products. A range of products is a set that utilizes a common group of components. Thus, if an agent observes a large increase in the demand for a certain product, the agent factory could use components from the common pool because of the decreased demand for another product in the same range. On the other hand, an MTS strategy with accurate forecasts could also prove viable in such SCM environments. Even better, a hybrid strategy with ATO and

MTS, if properly configured and evaluated, could boost the SCM agent's performance: the agent could work in a short-term ATO mode, and if it detects an increase in future market prices, employ an MTS strategy in parallel to produce additional stock for predicted highly profitable long-term orders.

Having selected an order fulfillment process, an agent should then reduce costs (component purchase and storage costs), increase profit margins from sales (sell as many products as possible at the highest possible prices), and maintain high customer service levels (minimize penalties and avoid missed revenue from canceled orders). For all these goals, DM techniques can provide an efficient paradigm for modeling the agent's decision mechanism. The agent designer might apply DM in these specific fields:

- *Predicting future demand*. Accurate predictions can indicate how many components to procure, when to expect prices to rise or fall, whether to save resources (components, products, and factory cycles) for future use, and so on. Because most operations have lead times in the supply chain, an agent must take action now in order to deal with events later. Thus, positioning the agent's decisions on the future-demand curve is vital for the agent's performance. Moreover, when deciding on the RFQs to bid on, the agent can simulate future auctions to choose whether to employ an eager (spare all resources on the current day's RFQs) or conservative (hold resources for forthcoming RFQs) bidding strategy.
- *Bidding*. The agent can use historical data (available in TAC SCM game logs after each game) to extract models for predicting an auction's closing price or calculating probability distribution functions of bid acceptance, given a requested price.[2]
- *Extracting useful patterns*. These patterns could comprise either economic regimes,[9] such as oversupply or scarcity of resources, or decision trees predicting unfruitful auctions. In the latter case, the agent could make bids equal to the reserve price and gain more profit, whereas in the former case, it could adapt its strategy according to market conditions.
- *Predicting game/environment state*. Based on the predicted demand curve, the agent can identify the values of certain important game variables such as demand for specific products or product ranges and daily minimum and maximum prices for each product. These variables are strongly correlated with customer demand and are used as predictor variables for other DM models.
- *Modeling suppliers*. Agents can also model suppliers through DM to predict procurement offer prices based on volume, due date, and reserve prices. These models can assist in identifying the optimal set of RFQs to send suppliers each day to get cheaper prices, while maintaining the appropriate inventory position (IP) equal to the safety stock (SS) levels.

On this last point, for each day x into the future, IP should equal SS to have the prespecified delivery performance,

$$IP_{d+x} = SS_{d+x} = OHI + \sum_{i=d}^{d+x}\left(OO_i\right) + \sum_{i=d}^{d+x}\left(BO_i\right), \quad (1)$$

where the IP for day $d + x$ into the future is equal to the on-hand inventory (OHI, components the agent currently has in stock), plus the components on order (OO, components ordered and that will be delivered from suppliers at day i) up to day $d + x$, minus back orders (BO, orders the agent has to deliver to customers at day i) up to day $d + x$. This is the standard equation for calculating the inventory position on any given day into the future. Based on the price-predicting models and Equation 1, we can optimize what orders we should place (volume and due date) to satisfy SS levels and minimize costs.

If we make these design choices, the agent's blueprint would look like the one in Figure 2.

Sales Modeling

Our analysis has focused on the most challenging aspect of SCM, the agent's bidding mechanism. It interacts with other entities (competitors and customers) and thus requires precise modeling of market conditions. Our main research goal was to identify the specifications for an efficient mechanism for handling the RFQs produced by customers each day, striving simultaneously for high service levels and high profitability, while always accounting for resource constraints.

In the TAC SCM auction environment, as well as in real-life situations,[2,10] it's common for the agent to employ a probability distribution function that predicts the probability of an offer becoming an order, given the bidding price. After maximizing the expected utility (expected profit, in our case), the agent can then calculate the optimal price:

$$ExpectedProfit = P\left(Win \middle| Price\right) \cdot \left(Price - Cost\right). \quad (2)$$

If an agent bids on a single auction or takes part in all auctions, the challenge is to identify the price that maximizes expected profit. TAC SCM is even more challenging because agents must decide on a subset of daily RFQs to bid on. Bidding the price that maximizes the expected profit for all daily RFQs for a consecutive number of days, although a legitimate strategy, will almost certainly lead to

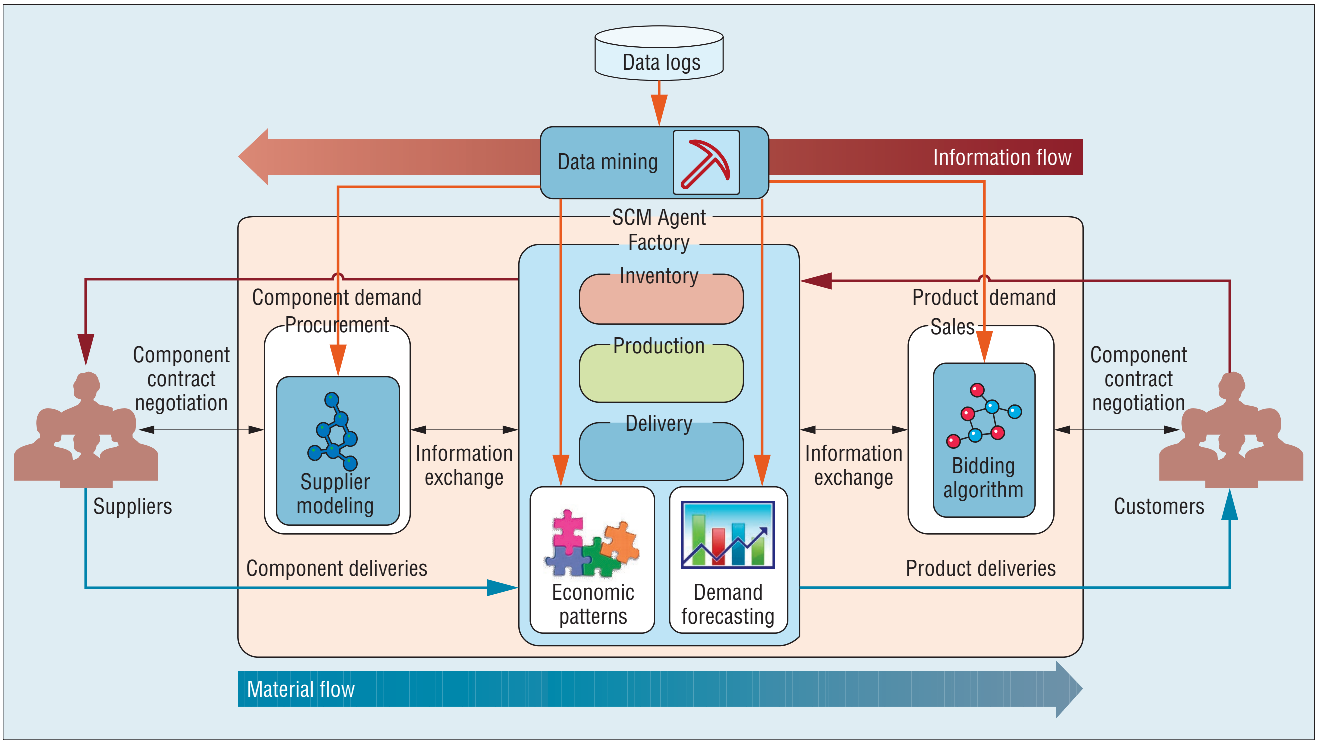

Figure 2. Agent blueprint. The basic architecture of a supply-chain-management agent uses data mining primitives (demand forecasting, economic patterns, customer, and supplier modeling).

large penalties, missed deliveries, and low revenue. Based on this hypothesis, the agent faces the problem of finding the optimal set of bids each day, and although it can apply mathematical optimization and local search algorithms, time limitations prohibit their use, so the agent must implement heuristic and greedy algorithms.[10]

We can quickly derive the probability distribution function by calculating the probabilities of acceptance at certain prices and then linearly interpolating between these specific points. More points, of course, mean higher accuracy. Preliminary statistical analysis indicates that bidding at the daily minimum price of a product observed the previous day results in winning 75 percent of the orders; this drops to 25 percent when bidding at the maximum price observed the previous day. A price of zero offers a percentage of 100 percent, bidding at the reserve price results in a low 2.5 percent, and bidding at any price x above the reserve price, according to the game specifications, has a 0 percent probability of acceptance. Figure 3 displays these percentages.

At this point, DM helps construct the distribution function. DM procedures generate the DM model based on historical transactions so it can predict the closing price for each auction by using order characteristics (reserve price, quantity, due date, and so on) and variables describing the state of the environment as predictors. Bidding at the predicted price results in a 50 percent chance of winning the order, because, intuitively, the error in the prediction will be a little higher or lower than the actual closing price. This also holds true experimentally. If we enforce a monotonicity constraint so that the inequalities between prices are $P_0 - price < P_{min} - price < P_{model} - prediction < P_{max} - price < P_{reserve} < P_x >_{reserve}$ all the time, we derive a probability distribution like the one in Figure 4. The probability distribution curve resembles a sigmoid function

$$\left(1 \big/ 1 + e^{(a+b\cdot z)} \right); \tag{3}$$

optimization software could fit it using iterative gradient methods. Adding one more point to the search of the probability distribution, especially in the linear part of the sigmoid function, results in a better modeling of that distribution. Additionally, having taken into account the 50 percent of order-winning probability when bidding at the price predicted by the model, an agent can skip calculating the probability distribution function and can bid up to two times its own resources. This approach isn't optimal, but it's much simpler and more time efficient.

Another DM model that could increase the agent's profit is to identify rules that encapsulate particular market states. Such states include

- the *start game period*, in which all agents try to build stock, thus buying components at higher prices;
- the *end game period*, in which all agents try to diminish their repositories, thus selling all of their stocked goods

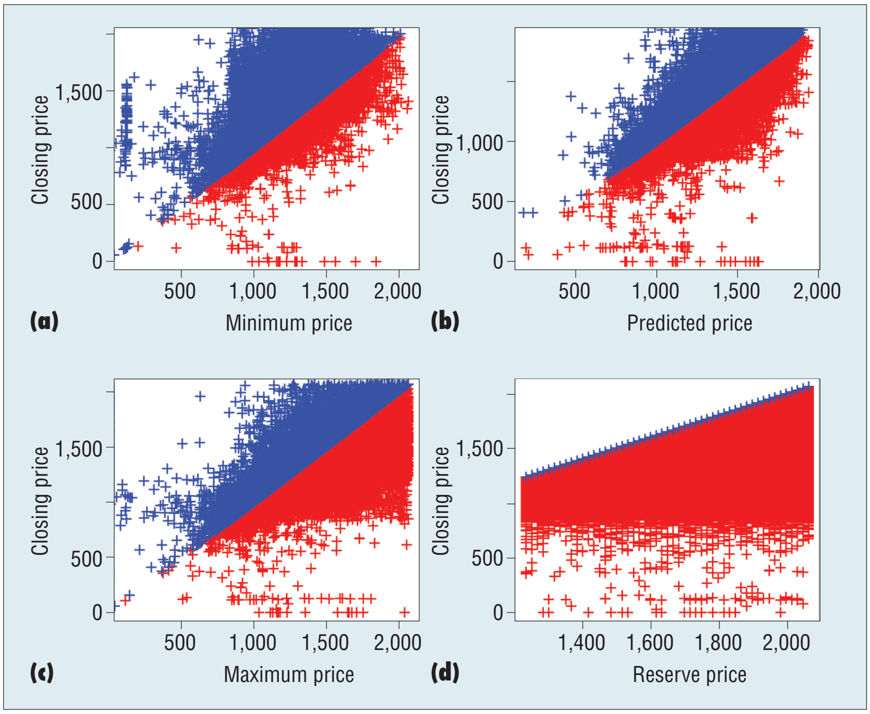

Figure 3. Prices. The *y*-axis represents the Trading Agent Competition closing price, and the *x*-axis the price under investigation: (a) the minimum price, (b) the predicted closing price, (c) the maximum price, and (d) the reserve price. Blue denotes auctions that would have been won by bidding the price shown. Red denotes those that would have been lost.

at lower prices;

- points in time *when certain components are depleted* and there's a low supply of certain products, thus affecting their price; and
- points in time *when there is an overdemand by customers* and certain RFQs don't receive any offers, leading to unfruitful auctions.

Algorithm 1 takes these seasonal phenomena into account, identifies market opportunities, and handles them accordingly.

Each day, the agent receives RFQs from customers and evaluates them with respect to the current game state. In addition, it makes a prediction of the RFQ price ranges for a time window of d days ahead. If it expects forthcoming RFQs to be more profitable, the agent doesn't use all its factory cycles on the current day's RFQs. After calculating its free capacity with respect to pending and future orders, the agent attempts to predict whether the current day's auctions will be fruitful. If not (because there will be no competition), the agent places a bid equal to the reserve (maximum) price. If the auction seems fruitful, the agent feeds the RFQ details and the current game state into a DM model responsible for predicting an auction's closing price. The agent sets the probability to 0.5 based on the already-performed statistical analysis of the model's behavior. It then calculates the expected profit for all offers and sorts them in descending utility order. For each offer, if resources are adequate, the agent makes a bid and discounts resources (factory cycles and components) on the probability of acceptance of any given offer.

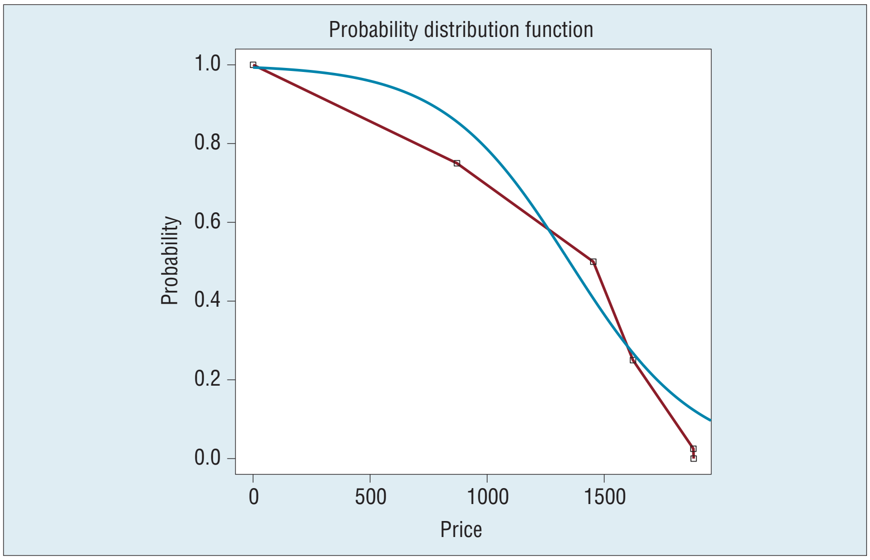

Figure 4. Probability distribution for a specific bid's acceptance. Square signs dictate the price-probability pairs from our data. The red line is the linear interpolation, and the blue line is the probability function's nonlinear least squares estimation (modeled as a sigmoid function).

Deriving DM Models

SCM environments often provide a deluge of data from different sources, aimed at serving different stakeholders. To turn this data into exploitable information, an agent must carefully collect and preprocess them. The quality and quantity of the data set is of vital importance because it practically determines the final algorithm performance (as DM experts say, "Garbage in, garbage out"). SCM environments such as TAC are much more complex because the market's dynamics and the interdependencies among all entities' performances dictate thorough

Algorithm 1. The data-mining-enabled bidding algorithm. An agent decides on the proposed bidding price for each request for quote (RFQ), with the help of the data mining models and then submits its offers based on profitability and factory resources available.

```
RFQs ← current day RFQs
current ← current game state
for i = 1 to i = d do
   demand predict demand in RFQs for day i
   RFQs.add(generateRFQs(demand))
   RFQs.creationDate ← i
end for
for all RFQs do
   state ← dm.predictState(current,rfq.creationDate)
   if dm.isUnfruitful(rfq,state) then
      rfq.bidPrice = rfq.reservePrice
   else
      rfq.bidPrice = dm.predictPrice(rfq,state)
   end if
   rfq.prob = 0.5
   rfq.profit = rfq.prob x rfq.bidPrice
end for
sort in descending expected profit order
while resources > 0 do
   agent.makeOffer(rfq)
   resources -= rfq.prob x rfq.resources
end while
```

analysis. For example, deviation in RFQ prices could imply a market opportunity (that is, profit) or market manipulation (that is, major loss).

The process we propose for building efficient models consists of the following facets:

- *Data aggregation and filtering*. Our first concern was to construct the optimal set of predictor variables. We filtered game instances to remove transient phenomena caused by start and end game effects, which produced data sets containing instances from day 50 to day 200. Additionally, we removed winning bids below 50 percent of the product base price as outliers (no profit is possible from bidding below such a price, according to the game specifications for supplier pricing).
- *Attribute selection*. We decided on a subset of attributes that best describes the inquired model, delivers the best generalization performance, and reduces data dimensionality.
- *Model development*. Finally, we extracted and evaluated different models using regression algorithms.

We performed our analysis on game logs from four consecutive tournaments that occurred from 2005 to 2008. We didn't consider logs from previous years (2003 and 2004) because a major change in game rules occurred in 2005. We chose the initial set of attributes based on their observability during the course of the game, allowing the agent to make online predictions. We derived one model for each of the 16 products; the initial set contained 38 attributes that attempt to predict the RFQ's winning price (see Table 1).

Because there was too much available data, we reduced the training data set to be time and memory efficient, without lowering the models' quality. For each year of competition, we created two data sets: one with the auctions on the product with the lowest base price, such as product ID 1, and one with the auctions on the product with the highest base price, such as product ID 8. Thus, we generated eight data sets. We performed random sampling to construct the final data sets and chose a 5 percent portion of the initial data, having observed that we could reach concrete conclusions by using learning curves. The final data sets contained 60,000 to 70,000 instances.

Our ultimate goal was to discover the subset of attributes that best describes an auction's closing price and has high predictive power despite the tournament, game, round, or year. This would make the agent more robust for different game states. We applied the *correlation feature selection* (CFS) algorithm with different search procedures such as *best first forward selection*, *greedy forward selection*, and *genetic search*. We kept all attributes that the feature selection algorithm picked more than 80 percent of the time

Table 1. The initial set of 38 predictor variables. Percentages refer to the product's base price.

Predictor variables	
1. Request for quote (RFQ) reserve price (75% to 120%)	35. Average of the maximum prices of the product for the previous 5 days
2. RFQ due date (3 to 12 days ahead)	36. Average of the total demand for the previous 5 days
3. RFQ penalty (5% to 15%)	37. Average of the range demand for the previous 5 days
4. RFQ quantity (1 to 20 PCs)	38. Average of the product demand for the previous 5 days
5. Current date	
6. Current total demand	
7. Current range demand in RFQs	
8. Current product demand in RFQs	
9–14. Minimum price of the product for the previous 5 days	
15–19. Maximum price of the product for the previous 5 days	
20–23. Total demand for the previous 5 days	
24–27. Range demand for the previous 5 days	
28–33. Product demand for the previous 5 days	
34. Average of the minimum prices of the product for the previous 5 days	

during attribute selection (tenfold cross-validation experiments); in all, we kept eight data sets. Consequently, to decide on the most useful attributes, we created a subset that contains all attributes represented in at least 75 percent of the cases in which the signs of the coefficients didn't change from year to year. After performing this procedure, all data sets contained the following attributes:

- reserve price of the RFQ under investigation,
- current total demand,
- current product demand,
- minimum and maximum prices the previous day,
- total and product demand the previous day,
- maximum price two days before,
- average minimum price over the last five days, and
- average maximum price over the last five days.

All of these attributes' coefficients stayed positive during regression testing, implying that the higher the predictors' values, the higher the winning price of the auction. Table 2 summarizes the results of our approach for different years, algorithms, and attribute sets.

We can easily identify that the final subset of attributes (26 percent of the original data set's variables) performs very well compared to the original. And although different agents with different strategies participate in the game over the years, the attributes we selected are capable of capturing the market. From the major regression algorithms we applied, M5´ outperformed the rest of the data sets, probably because splitting the space with a decision tree and consequently employing different linear models allows M5´ to identify different market states.[11]

For example, given the same game state, the target output (predicted price) can vary when three agents participate in an auction instead of the usual six. Although we didn't consider start and end game effects, the models generalized quite well and predicted the closing prices of such periods with great accuracy. The same holds true when we used data from different years to train and test the models (see Table 3). Last but not least, the errors in probability estimation for various probabilities (0.7, 0.6, 0.5, 0.4) were very small, ranging from 0.001 to 0.05 (we used the WEKA data analysis software for performing DM[11]).

We also used DM to predict whether an auction would prove unfruitful. We used the same initial set of attributes as in the previous section and performed classification to identify whether an auction will be met (classified as 1) or not (classified as 0). Again, we employed attribute selection methodologies to find the most prominent attributes. We performed experiments using the data sets from the 2005 finals (more competitive) and second finals (less competitive) and generated six data sets: two tiny (6,245 and 5,806 instances), two medium (124,223 and 118,831 instances), and two huge (621,577 and 592,084 instances, respectively).

We applied the CFS algorithm (as in the previous set of experiments) as well as ReliefF to rank the tiny data sets and obtained 14 potential candidate subsets of attributes. We applied several algorithms to measure accuracy (C4.5, REPtrees, DecTables, NaiveBayes, RBFNets, IBk, JRip, BayesNet, and PART) and chose the most accurate for a tenfold cross-validation evaluation:

- current simulation date,
- reserve price,
- minimum and maximum price of product the previous day,

Table 2. Predicting an auction's closing price.*

			2005		2006		2007		2008	
			Product 1	Product 8	Product 1	Product 8	Product 1	Product 8	Product 1	Product 8
Full set (FS)	LR	CC	0.866	0.859	0.92	0.916	0.904	0.897	0.827	0.827
		MAE	74	118	62.5	96	66	104	98	141
		RMSE	113	176	97	148	93	146	141	200
	M5´	CC	0.886	0.883	0.934	0.931	0.924	0.915	0.865	0.877
		MAE	65	101	55	83.5	57	91	83	114
		RMSE	105	161	88	135	83	133	126	173
Yearly FS	LR	CC	0.865	0.857	0.932	0.915	0.902	0.896	0.826	0.832
		MAE	74	119	63	97	67	104	98.5	141.5
		RMSE	114	177	98	149	94	147	142	200.5
	M5´	CC	0.887	0.881	0.933	0.929	0.922	0.915	0.865	0.877
		MAE	64	101.5	55	84	58	92	82.5	114
		RMSE	104.5	163	89	137	84	134	126	174
Final FS	LR	CC	0.865	0.858	0.918	0.915	0.901	0.895	0.824	0.832
		MAE	74	119	63	97	67	105	99	142
		RMSE	114	177	98	149	95	148	142	201
	M5´	CC	0.889	0.882	0.933	0.93	0.922	0.913	0.865	0.876
		MAE	64	101	55	84	58	92	83	114
		RMSE	104	162	89	136	84	135	126	174
	NN	CC	0.886	0.881	0.932	0.928	0.919	0.913	0.864	0.875
		MAE	66	104	56	86	60	94	84	117
		RMSE	105	162	90	137.5	86	136	126	175

* Correlation coefficient (CC), mean absolute error (MAE), and root mean squared error (RMSE) are provided for linear regression (LR), trees (M5´), and neural networks (NN) modeling using tenfold cross-validation. The range of the target attribute is [825, 2062.5] for product 1 and [1175, 2937.5] for product 8.

Table 3. Establishing the validity and robustness of the models by training and testing using different years.*

Train:2005 Test:2006			
	CC	MAE	RMSE
Product 1	0.927	58.15	92.07
Product 8	0.924	88.23	141.46
Train:2006 Test:2007			
	CC	MAE	RMSE
Product 1	0.916	59.02	87.10
Product 8	0.909	94.09	138.35
Train:2007 Test:2008			
	CC	MAE	RMSE
Product 1	0.853	83.80	130.80
Product 8	0.869	116.81	178.69

* The errors are in the same range when using tenfold cross-validation with only one year as input using correlation coefficient (CC), mean absolute error (MAE), and root mean squared error (RMSE).

- current demand, and
- previous day demand.

Although accuracy might not be the best evaluation measure, it provides a qualitative overview of the problem at hand. We performed more elaborate testing using the selected attributes to evaluate whether they're related to start and end effects (current date), demand (customer demand the last two days), and strong competition between agents (minimum and maximum prices' distance from the reserve price). We selected C4.5 as the best performing algorithm and then experimented on the optimal configuration, to improve recall and precision.

In almost all games, the competing agents met most of the auctions (more than 80 percent), thus the model we extracted would be biased toward predicting fruitful auctions. We re-adjusted the cost matrix for the tiny data set and tested it on the medium and huge data sets. We pruned the final C4.5 model, with a confidence of 0.0005 and a cost matrix of (0.2, 3.0) for true negative (TN), false positive (FP), false negative (FN), and true positive (TP). We assigned higher cost to FNs because, by predicting an auction as unfruitful and setting the offer price as the reserve price, it's very likely that the offer won't be accepted due to competition. On the other hand, predicting that the manufacturers will meet an auction when in fact the auction isn't met will make the agent lose profit but not the entire order. Table 4 summarizes the results for the two data sets.

Table 4. Predicting unfruitful auctions in the finals of TAC SCM 2005.

	Medium		Huge	
Measure	**Finals**	**2nd finals**	**Finals**	**2nd finals**
Accuracy	97.2	88.1	96.9	89.5
Precision	0.89	0.78	0.93	0.83
Recall	0.26	0.48	0.59	0.53
F-measure	0.40	0.59	0.72	0.65
Mean absolute error	0.04	0.16	0.02	0.15
ROC*	0.89	0.86	0.94	0.88

* Receiver operating characteristic

By using DM primitives, our goal was twofold: understand SCM market behavior and identify potential, and produce robust models induced from facts (data logs) that are viable and economically efficient in various market states. To validate our theoretical analysis, we built Mertacor, an agent that we benchmarked in the TAC SCM competition from 2005 on. Elaborate experimentation in the sales facet has induced DM models that exhibit stable behavior in a variety of situations and are able to adapt without remodeling. Once the domain expert prepares the training sets and decides on the optimal model, agent performance proves efficient and requires only fine-tuning. At least, that's what our Mertacor placements in the TAC SCM competition indicate (for more information, visit the official Web site at www.sics.se/tac/page.php?id=13). We, thus, strongly argue in favor of the utilization of DM techniques in SCM environments, especially in pricing decisions. DM is a strong candidate for enhancing the intelligence of autonomous agents in multivariable domains and should be considered as such.

THE AUTHORS

Kyriakos C. Chatzidimitriou is a PhD candidate in the Department of Electrical and Computer Engineering of the Aristotle University of Thessaloniki. His research interests include machine learning and trading agents. Chatzidimitriou has an MS in computer science from Colorado State University. He is a student member of the IEEE and the IEEE Computer Society. Contact him at kyrcha@issel.ee.auth.gr.

Andreas L. Symeonidis is a lecturer with the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki and a faculty affiliate of the Informatics and Telematics Institute in Thessaloniki. His research interests include software agents, data mining and knowledge extraction, intelligent systems, Web semantics, social networks, and evolutionary computing. Symeonidis has a PhD in electrical and computer engineering from the Aristotle University of Thessaloniki. He is a member of the IEEE and IEEE Computer Society. Contact him at asymeon@issel.ee.auth.gr.

References

1. D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi, *Designing and Managing the Supply Chain*, McGraw-Hill, 2003.
2. W. Elmaghraby, "Auctions and Pricing in E-Marketplaces," *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, D. Simchi-Levi, S.D. Wu, and Z.J. Shen, eds., Kluwer, 2004, pp. 213–246.
3. C. Zhang et al., "Agents and Data Mining: Mutual Enhancement by Integration," *Proc. Int'l Workshop Autonomous Intelligent Systems: Agents and Data Mining* (AIS-ADM 05), LNCS 3505, Springer, 2005, pp. 50–61.
4. A.L. Symeonidis et al., "Data Mining for Agent Reasoning: A Synergy for Training Intelligent Agents," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 8, 2007, pp. 1097–1111.
5. R. Arunachalam and N. Sadeh, "The Supply Chain Trading Agent Competition," *Electronic Commerce Research and Applications*, vol. 4, no. 1, 2005, pp. 63–81.
6. Swedish Inst. Computer Science, "Trading Agent Competition," Sept. 2008; www.sics.se/tac.
7. J. Collins et al., *The Supply Chain Management Game for the 2005 Trading Agent Competition*, tech. report CMU-ISRI-07-100, Carnegie Mellon Univ., 2006.
8. F. Cheng, M. Ettl, and G. Lin, *Inventory-Service Optimization in Configure-to-Order Systems*, tech. report, RC 21781, IBM, 2001.
9. W. Ketter et al., "Identifying and Forecasting Economic Regimes in TAC SCM," *Proc. Int'l Joint Conf. Artifical Intelligence* (IJCAI 05) *Workshop on Trading Agent Design and Analysis* (TADA 05), LNCS 3937, Springer, 2005, pp. 113–125.
10. D. Pardoe and P. Stone, "Bidding for Customer Orders for TAC SCM," *Proc. 2nd Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems* (AAMAS 04) *Workshop on Agent-Mediated Electronic Commerce* (AMEC 04), Springer, 2005, pp. 143–157.
11. I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools with Java Implementations*, Morgan Kaufmann, 2000.