A Demand-Driven Approach for a Multi-Agent System in Supply Chain Management

Yevgeniya Kovalchuk and Maria Fasli

School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom
{yvkova,mfasli}@essex.ac.uk

Abstract. This paper presents the architecture of a multi-agent decision support system for Supply Chain Management (SCM) which has been designed to compete in the TAC SCM game. The behaviour of the system is demand-driven and the agents plan, predict, and react dynamically to changes in the market. The main strength of the system lies in the ability of the Demand agent to predict customer winning bid prices – the highest prices the agent can offer customers and still obtain their orders. This paper investigates the effect of the ability to predict customer order prices on the overall performance of the system. Four strategies are proposed and compared for predicting such prices. The experimental results reveal which strategies are better and show that there is a correlation between the accuracy of the models' predictions and the overall system performance: the more accurate the prediction of customer order prices, the higher the profit.

Keywords: Multi-Agent Systems, Trading Agents, Supply Chain Management, Prediction, Neural Networks.

1 Introduction

Supply Chain Management (SCM) involves a number of activities from negotiating with suppliers to competing for customer orders and scheduling the manufacturing process and delivery of goods. The activities are different in their nature: they work with different data, have different tasks and constraints. At the same time, they are interrelated to ensure the achievement of the ultimate goal of maximizing the enterprise's profit. This makes the chain very difficult to manage: being successful in one area of the supply chain does not necessarily guarantee the improvement of the overall performance. Designing an effective decision-support system (DSS) for SCM has become crucial in recent years, especially nowadays, when enterprises can no longer rely on static strategies for operating their business. With the advent of e-Commerce and in a global economy, SCM systems have to be able to deal with uncertainty and volatility of modern markets.

This paper introduces an intelligent DSS for SCM. A multi-agent approach is applied for designing the system in order to deal with the complexity of the domain and to provide flexibility regarding the system architecture. This approach allows

E. David et al. (Eds.): AMEC/TADA 2009, LNBIP 59, pp. 88–101, 2010.

separating different tasks within the SCM and exploring them both independently and in relation to each other. The system can be broken down into separate building blocks, each concentrating on a particular part of the supply chain. By replacing one building block with another and by combining them in different ways, various versions of the system can be created. In this way, the influence of changes in behaviour in each link of the supply chain can be systematically studied. In addition, the concept of agents is used to facilitate industrial application of the system: by assigning an autonomous agent to a separate entity of the supply chain, the tasks can be distributed geographically as well as implemented using different platforms.

The architecture of the proposed system includes agents for each link of the supply chain: supply, inventory, production, selling, and delivery. While following their own goals, the agents work in cooperation in order to achieve the common ultimate goal – to maximize the overall profit. The Demand agent takes the leading role in the system: the performance of the other agents is organised in such a way so as to ensure execution of customer orders on time. The main task for the Demand agent is to provide the most profitable customer order bundle. It does this by predicting the highest prices it can offer customers for each of their requests for quotes (RFQs) and still win their orders. Different strategies for predicting customer order prices are considered in this work. The first strategy is to model competitors' behaviour, predict their offer prices and bid just below them. The second approach is to predict customer order prices based on the time-series of these prices. The third strategy is to predict the prices based on details of the customers' RFQs, market details and bidding history. Finally, the last strategy is to predict probabilities of the winning price to be in particular intervals and bid according to the most probable price. The Neural Networks learning technique is used in the predictors.

The system has been tested in the TAC SCM simulated environment [9], which is now probably the best vehicle for testing SCM agents. It encapsulates many of the tradeoffs that could be found in real SCM environments: time-constraints, network latency, unpredictable opponents, etc. The generalized problem competitors are faced with can be formulated as follows: "given a market situation with specific rules, how does one act to buy, sell, and produce goods to maximize expected profit?" [10].

Many research teams have dedicated their work to exploring various issues that arise within the TAC SCM environment. They offer different system architectures and explore various methods for dealing with uncertainty and the volatility of the environment. This paper contributes to the area by offering a new multi-agent demand-driven architecture for SCM systems. Moreover, the paper introduces a number of algorithms for predicting customer order prices, which have not been explored in the TAC community yet. We compare the algorithms in terms of their accuracy of prediction and influence on the overall system performance.

The rest of this paper is organized as follows. An overview of related work is provided first. The description of the behaviour of the internal agents in the system follows. Section 4 introduces the approaches for predicting customer order prices. The experiment settings and results are presented next. The paper closes with the conclusions and a discussion of future work.

2 Related Work

The idea of applying a multi-agent approach to SCM systems has become very popular in recent years. We refer to [11] as one of the first attempts to organize the supply chain as a network of intelligent agents. The latest collection of papers on the applications of agent technology to SCM can be found in [7]. The book also discusses advantages and disadvantages of the agent-based approach for designing industrial software. The multi-agent system developed in [24] helps to reduce the total cost and bullwhip effect across the supply chain.

A significant contribution to the area has been made by the research teams that design trading agents to compete in the TAC SCM game. A survey of design approaches of these agents can be found in [14]. The survey is organized by the primary research agenda considered by the agents' developers: constraint optimization, machine learning, management of dynamic supply chains, scalable autonomous agents, architecture, empirical game theory, dealing with uncertainty, decision coordination, agent coordination mechanisms, predicted sales volume, future production schedule, inventory management, central strategy module, separate supply and demand models, and internal markets. Our paper contributes to this research by presenting an original multi-agent demand-driven architecture for the SCM system. In addition, the paper proposes four different strategies for sellers to follow when setting customer offer prices. The algorithms developed according to these strategies differ from the ones proposed by other TAC SCM participants. The methods used by other teams include fuzzy reasoning inference mechanisms [13], additive regression with decision stumps [21], linear regression [2], linear cumulative density function (CDF) [3], reverse CDF [16], continuous knapsack problem [1], dynamic pricing [5], and k-nearest neighbors [8, 17]. According to [8], the M5 algorithm outperforms multiple linear regression, neural networks, and support vector machines (SVM) when predicting customer wining bid prices. The M5 algorithm along with BoosTexter [25] have also been supported in [22], where the authors compared these algorithms with neural networks, decision stumps (single-level decision trees) boosted with additive regression, J48 decision trees, SVM, naïve Bayes, and k-nearest neighbours. According to [15], all the aforementioned methods do not take into consideration market conditions that are not directly observable. The authors use a Markov correction-prediction process and an exponential smoother to identify the market regimes and a Gaussian mixture model to determine the probability of receiving a customer order in different regimes for different prices.

The Neural Networks (NN) learning technique has not found much support within the TAC SCM community [22, 8]. However, it might be due to the fact that researchers have been using standard setup in their learning algorithms as implemented in tools such as WEKA (`http://www.cs.waikato.ac.nz/ml/weka/`) [28], Matlab (`http://www.mathworks.com/`), and Netlab (`http://www.ncrg.aston.ac.uk/netlab/`). We developed our own NN predicting tool and experimented with its settings, as we found strong evidence of successful application of NNs for solving forecasting tasks in the domains of finance and business other than TAC SCM. An overview of successful NN models applied to marketing, retail, banking and finance, insurance, telecommunication, and operations management is

provided in [26]. Empirical evidence of applicability of NN to the prediction of foreign exchange rates is reported in [29]. The authors of [6] discuss application of classical regression models, NN, fuzzy logic, and fractal theory for forecasting time series of dollar/peso exchange rate, U.S./Mexico exchange rates and prices of onions and tomatoes in the U.S. market. They conclude that the regression models show the poorest performance, and also that NN outperform fuzzy logic when forecasting in the short-term, while fuzzy logic outperforms NN when forecasting in the long term. In [12], the researchers propose several methods for predicting online auction prices using regression, decision trees (C5.0), and NNs. Their binary classifier based on NNs demonstrated the highest prediction accuracy (96%).

3 System Architecture

The system has a multi-agent architecture. Each agent within the system is responsible for a particular aspect of the supply chain. Although each agent focuses on specific tasks within its problem domain trying to achieve its own goals and having its own constraints, the agents do not act in isolation. They communicate with each other in order to achieve the main goal of generating profit. The system includes the following agents: Manager agent, Demand agent, Supply agent, Inventory agent, Production agent, and Delivery agent. The agents are described below in turn and Figure 1 illustrates the system architecture using UML notation [4].

The Manager agent is responsible for the communication with the TAC server as well as managing all other agents. It undertakes the following tasks: (1) Imports game settings, competitors' identities, Bill of Materials, and Component Catalog; (2)Updates inventory, factory and bank status; (3) Gets supplier offers, customer RFQs and orders; (4) Sends customer offers and supplier RFQs and orders; (5) Sends production and delivery schedules; (6) Gets market and price reports; (7) Keeps a record of RFQs, offers, orders, schedules, reports, and other information shared by all other internal agents; (8) Coordinates the agents' performance. While managing the whole SCM system, the agent aims to maximize the overall profit.

The Demand agent deals with selling personal computers (PCs) to customers. Each day it gets customer RFQs and orders from the Manager. In addition to these, the agent generates RFQs that might arrive in the future. Due to the limited production capacity, future demand has to be taken into consideration when scheduling production: future orders might bring more profit than the current ones. It has been shown in [21] that predicting future demand level (number of RFQs in a Bundle) doesn't significantly improve the system's performance comparing to setting this level equal to the current level. According to this, we assume that the future RFQ bundle contains the same number of RFQs that arrived on the current day. The value of each parameter of a future RFQ is chosen uniformly in the interval between the minimum and maximum allowed values for this parameter according to the game specification. For every new and future RFQ, the agent decides on the bidding price to offer to the customer. This paper introduces several approaches for setting offer prices which are discussed in the next section. Having the bidding prices, this agent estimates the profit of both new and future RFQs based on the latest prices the Supply

agent paid for the components. It then sorts the RFQs in profit descending order and asks the Production agent to project production for 10 days in the future (i.e. create production drafts) using the details of the new and future RFQs as well as orders. Considering only the new RFQs allocated to production drafts, the Demand agent generates customer offers and returns the RFQ bundle to the Manager to be sent to customers. The goal of this agent is to maximize revenue from the customers' orders.

The remit of the Supply agent is the procurement of low cost components on time from suppliers. Considering the component demand, current level of component usage, and available stocks, the agent generates its supplier RFQs. The agent uses the strategy of sending RFQs with different due dates. Long-term RFQs to arrive in 20 days are sent according to the current level of component usage to benefit from lower prices. Short-term RFQs to arrive in 3-6 days are then sent to meet current production needs. The agent tracks the suppliers' deliveries and prices, and sends its RFQs to the suppliers with the lowest level of current prices and delays. The agent sets its RFQ prices based on the prices paid recently, current prices quoted by the suppliers for probe RFQs (RFQs with zero quantity), and prices provided in the latest market report. When the RFQ details are decided, the agent generates an RFQ bundle. After getting offers from suppliers, the Supply agent generates its order Bundle. It accepts all complete offers and earliest partial offers. The RFQ Bundle along with the Order Bundle are passed to the Manager who sends them to the corresponding suppliers.

The Inventory agent manages the arrival of components from suppliers and assembled PCs from production, as well as releases components for production and PCs for delivery to customers. It registers the component and PC demands of the Production and Delivery agents respectively, and tries not to let the inventories go below a certain threshold in order to satisfy these demands. To minimize inventory storage costs, the agent dynamically adjusts the threshold levels for each component. To avoid situations where the Production agent schedules the production of PCs that cannot perhaps be produced due to lack of components, the Inventory agent also manages the critical levels of each component below which the PC production cannot be scheduled.

The Production agent is responsible for scheduling current production and projecting production in the future. Having the details on customer RFQs and orders from the Demand agent and component inventory stocks from the Inventory agent, the agent schedules its production for 10 days in the future. Having a limited production capacity, it tries to maximize the production utility (the potential profit that the scheduled production might generate). For every day in the future, the agent schedules the current and late orders, depending on their due date, profit and availability of components, and then it allocates current and future RFQs, again considering their due dates, profit and availability of components.

The remit of the Delivery agent is to deliver PCs to customers according to their orders. To prevent penalties for late deliveries, it schedules the delivery of active orders as soon as the requested PCs are released from production. It sorts current active orders by their due date and allocates the delivery of these orders into the current delivery schedule until the corresponding PCs are available in store.

The UML sequence diagram Figure 2 summarizes the interaction between the agents.

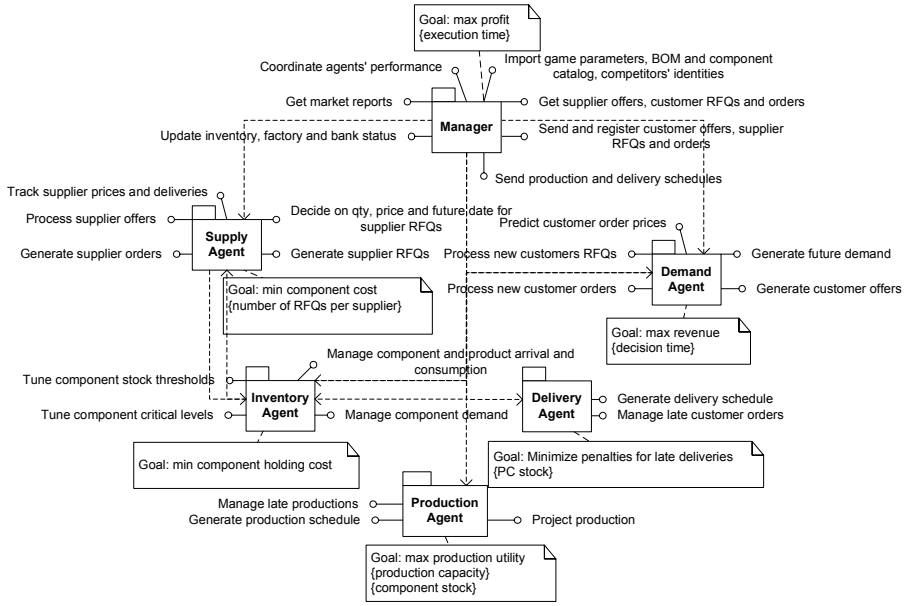

Fig. 1. The SCM system architecture

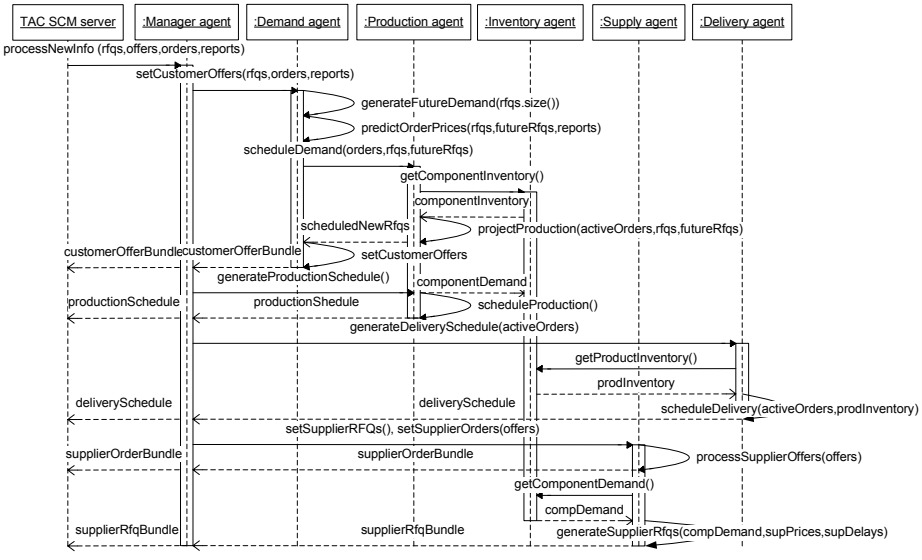

Fig. 2. Agent interactions in the SCM system

4 Strategies for Predicting Customer Order Prices

This paper investigates approaches for setting customer offer prices and how various algorithms for predicting winning bid prices influence the overall system's performance.

In the TAC SCM game, there are six agents who act as product manufacturers competing for supplier components and customer orders for finished PCs. Customers send RFQs to all agents for the 16 types of PCs that can be manufactured on a daily basis. Agents make offers and according to the game rules, customers accept the lowest offers proposed among all agents. Information on competitors' offer prices is not available to TAC agents. However, apart from RFQs details, the lowest and the highest order prices for each PC type from the previous day are available.

Four different strategies to determine which prices to offer customers are proposed. All of them are based on customer order price predictions. The first strategy is to predict competitors' offer prices and bid just below them. The second one is to predict the lowest and the highest customer order prices for each product based on the time-series of the prices and bid in between the predicted values. According to the third approach, order prices are predicted based on details of the customer RFQs, market details and bidding history. Finally, the last approach is to predict probabilities of an order price to be in particular intervals and bid according to the most probable price.

The Neural Networks learning technique (NN) is used to make predictions. Genetic Programming (GP) has been also applied for modelling competitors' behaviour and making time-series predictions [19]; however, we found that NN models outperform GP models in terms of accuracy of prediction, time execution, and complexity of implementation. Thus, only NN models are considered. NN architectures of the models differ to meet the requirement of each algorithm. The sigmoid activation function and Back-propagation training algorithm [20] are used in all NNs.

4.1 Modelling the Competitors' Behaviour

According to the game specification, customers choose the lowest price among the ones offered by all sellers. Prediction of the competitors' prices for an RFQ allows to identify the lowest price which will be offered to a customer. Using GP, the trees have been evolved for each competitor, to represent which attributes a competitor is using when setting its offer prices [19]. According to these trees, an individual NN has been constructed for each competitor: only the attributes represented in the competitor's tree have been included as inputs to its NN. The full set of inputs consists of the following parameters: PC type, current date, lead time (due date minus current date), quantity, reserve price, penalty, the lowest and highest reported market price, and current demand level. Inputs are normalized to be in the interval [0.1; 0.9], using the minimum and maximum allowed values for each input according to the game specification (formula 5).

4.2 Time-Series Prediction

In the TAC SCM game, the lowest and highest customer order prices for each product type are available from the previous day. In the context of a highly competitive

market the difference between these prices tends to be very small. It has been experimentally established that setting offer prices in between these prices is a competitive strategy. According to this, the NN learning technique is applied to perform time-series forecasts of the lowest and highest customer order prices for the next day. Customer offer prices are then set in between the predicted values. Algorithms within this group vary in the following: data transformation methods; data normalisation methods; number of historical data points included in time-series. The following data transformation and normalisation methods are applied over NN inputs:

Differential transformation

$$x_d = x_t - x_{t-1} \tag{1}$$

Rational transformation

$$x_r = \ln(\frac{x_t}{x_{t-1}}) \tag{2}$$

Statistical transformation

$$x_s = \frac{x_t - \bar{x}}{\sigma}, \qquad \bar{x} = \frac{1}{N}\sum_{t=1}^{N} x_t, \quad \sigma^2 = \frac{1}{N-1}\sum_{t=1}^{N}(x_t - \bar{x})^2 \tag{3}$$

Linear varied normalization

$$x_i^{lv} = (\frac{x_i - x_{\min}}{x_{\max} - x_{\min}}) \tag{4}$$

Linear fixed normalization

$$x_i^{lf} = (\frac{x_i - x_{\min}}{x_{\max} - x_{\min}}) \cdot 0.8 + 0.1 \tag{5}$$

Non-linear normalization

$$x_i^{nl} = \frac{1}{1 + e^{-x_i}} \tag{6}$$

where x_t and x_{t-1} are consecutive data values in a series; $x_{\min}$ and $x_{\max}$ are the minimum and maximum allowed values for the corresponding data type; $\bar{x}$ is the mean of the series values and σ^2 is their variance.

The models take price values from six or eleven preceding days and predict the winning price for one day in the future.

On average, the most accurate model appears to be the one with the differential transformation method, linear varied normalisation method, and eleven data points in

input time-series. However, during the course of a game, accuracy of the models' predictions varies. A meta-model has been applied over the models to find the final predicted price according to the models' performance in runtime. The heuristics of the meta-model is based on the idea of reinforcement learning. The final predicted price is set to the weighted sum of the prices predicted by all time-series models. Weights are summed up to 1 and tuned on-line during the course of a game: the most currently accurate model is rewarded by increasing its weight, while the worst model is punished by decreasing its weight. The optimal step for tuning the weights is set experimentally to 0.01. Experiments demonstrated that inclusion of the meta-model doesn't improve the accuracy of prediction compared to when only the best performing on average model is applied. According to this, only this best time-series model (TB, "Time-series the Best") is tested in the experiments that follow.

4.3 Order Price Prediction Based on Bidding History

According to this approach (referred further as WP, "Winning Price"), customer order prices are predicted for each RFQ using RFQ details, current market information, and results from previous auctions. Using this information, the NN predicts the expected value of the order price. The inputs for the model include: product type, its quantity, current date, due date, penalty, customer reserve prices, the lowest and the highest customer order prices for the last three days, and the current demand level (ratio of the number of RFQs received from the customers to the maximum possible number according to the game specification). Records in the training set map these attributes to the actual order price. The number of hidden units is set to 5 and the learning rate is tuned during the training process according to the dynamics of the prediction error.

4.4 Order Price Probability Prediction

A set of ensembles of NNs, one set for each product type, is designed to predict order price probabilities. The possible price range is split into small intervals. Each NN in the ensemble is assigned to one such interval and predicts the probability of the order price to be in this interval. The final price is set to a random value from the interval with the greatest probability (the random element makes prices hard to predict by our opponents). The strategy for setting the upper limit of the possible price range varies. According to one algorithm (PF, "Probability Fixed"), the upper price limit is fixed according to the highest price observed in all previously played games. In another algorithm (PV, "Probability Varied"), the upper limit is set for each RFQ individually according to the customer reserve price (the highest price the customer is willing to pay). The inputs for both algorithms include RFQ details and current market information, such as: type of product requested, its quantity, current date, due date, penalty, customer reserve prices, the lowest and the highest customer order prices for the last three days, and the order level as calculated for the previous day (ratio of the number of orders received from customers to the number of offers sent to them). Along with these attributes, an offered price and the corresponding binary code showing if the offer with this price resulted in a customer order is recorded during the games for each RFQ. These records are used for training the models. The input units are normalised according to formula (5).

5 Experimental Setup

Having a number of learning algorithms for predicting customer order prices, the task is to compare their predictive abilities and to identify the strategy which is better for the Demand agent to follow so as to ensure a better overall system performance.

When learning from data, we are interested in which data are perceived from the environment, how these data are preprocessed, and in what way the output is used for making decisions. Our time-series predictors use price values only. The other algorithms require more information from the market environment. The input set is the same for these algorithms with the only exception that probability predictors use order level instead of demand level. The algorithms differ in NN settings, methods for preparing inputs, and in the way the outputs are used to set customer offer prices.

First, a number of experiments have been run in the TAC SCM simulated environment to indentify the most accurate predictive model. In order to do this, all the models have been tested simultaneously and the prices predicted by them were recorded for further analysis. To provide a fair evaluation benchmark, the customer offer prices have been set using a random element according to the following formula:

$$Offer\ price = (p_{highest} + p_{lowest})/2 + a_1 - a_2 \tag{7}$$

where p_{lowest} and $p_{highest}$ are the lowest and highest customer order prices reported on the previous day; a_1 and a_2 are coefficients set to random values within the interval [0; 20] (the upper limit of the interval is set according to the average gap between the lowest and highest customer order prices observed in the games).

The second set of experiments has been run to explore how predictive models affect the overall system's performance. The experiments have the aim to identify which model helps to get the best score and if there is a correlation between the accuracy of the models' predictions and the score achieved in the game. The models have been tested in pairs: two versions of the system with different predictors have been playing in the same game against each other and four other competitors. All other settings in both versions of the system have been kept the same.

The following TAC SCM agents have been chosen as competitors: TacTex2007 [21], PhantAgent2006 [27], Maxon2006, SouthamptonSCM 2006 [13], and CrocodileAgent2005 [23] (the agents' binary code is publicly available at http://www.sics.se/tac/). For the second set of experiments, the second version of the system replaced the TacTex2007 agent. For both experiments, 30 games have been played to collect the data for training the models and then another 40 games – to estimate their performance.

6 Results

For the first set of experiments, where the accuracy of models predictions has been estimated, the models are compared in terms of their average relative error (ARE):

$$ARE = \frac{\frac{1}{N}\sum_{i=1}^{N} abs(x_i^{actual} - x_i^{pred})}{\frac{1}{N}\sum_{i=1}^{N} x_i^{actual}} \tag{8}$$

Table 1. Summary of models' performance

Model Name	Abbreviation	Section	ARE (st. dev.)	Rank
Competitor Individual	CI	4.1	0.0437 (0.017)	3
Time-series the Best	TB	4.2	0.0320 (0.016)	1
Winning Price	WP	4.3	0.0353 (0.016)	2
Probability Fixed	PF	4.4	0.1080 (0.034)	-
Probability Varied	PV	4.4	0.1080 (0.028)	4

Table 2. Models' pair-comparison

Experiment	% of wining games		% of winning bids	
	Model 1	Model 2	Model 1	Model 2
CI vs. TB	0	100	62,7	50,2
CI vs. WP	0	100	52,0	45,4
CI vs. PV	100	0	62,9	58,3
TB vs. PV	100	0	58,3	56,8
WP vs. PV	100	0	61,6	56,1
TB vs. WP	60	40	53,8	52,7

where x_{actual} and x_{pred} are actual and predicted customer order prices observed in a case; N is the number of cases recorded in all games.

The detailed discussion of the results from the first set of experiments can be found in [18]. In summary, the algorithms cope with the dynamics of the environment very well: accuracy of their predictions remains the same throughout a game considering that some opponents also learn. According to Table 1, the time-series model gives the highest accuracy of prediction (ARE=3,2%) followed by WP model which achieves ARE=3,5%. The strategy of applying competitors' price predictors gives ARE=4,4%, while both probability price predictors provide the lowest accuracy with ARE=10,8%.

In the second set of experiments, the effect of applying the predictive models on the overall system performance has been estimated. Two different versions of the system have played against each other and the percentage of winning games as well as the number of orders won compared to the number of offers sent have been estimated for each of them. According to the results (Table 2), the systems with the TB and WP models perform similarly good, outperforming other versions of the system which use the CI or PV predictive models (as PF and PV models predict with the same accuracy and their architectures are similar, only the PV model has been tested in the second set of experiments). The system with the PV models achieves the lowest score. The ranking order for the models is provided in Table 1.

Combining the results from both sets of experiments the conclusion can be drawn that there is a strong correlation between the models' accuracy of prediction and total score achieved in games. At the same time, the algorithms leading to the better overall performance do not necessarily provide higher percentage of winning orders (the ratio of the number of offers sent, to the number of orders received). For example, the strategy of predicting competitors' prices (CI), which comes third, provides the highest percentage of winning bids comparing to all other strategies. Therefore, a more extensive analysis of the algorithms' performance is required. In particular, the

ratio of prices predicted lower than the actual prices to those of higher predicted, as well as the relation between the predicted prices and the ones set by competitors have been investigated, however due to the limited space are not included to this paper.

7 Conclusions and Future Work

SCM is a very complex and dynamic process. It includes a number activities, which, on the one hand, have their particular individual tasks to perform and goals to achieve, but on the other hand, they are connected and interdependent. Being successful in one area of the supply chain does not necessarily guarantee the improvement of the overall performance. Thus, there is the need for a mechanism to separate different tasks and explore them both independently and in relation to each other. We implemented such a mechanism in our multi-agent decision support system for SCM. The multi-agent approach allows to change the behaviour of each agent at a time and identify how the changes affect the overall system's performance.

The proposed system consists of six agents: one for each link in the supply chain (supply, inventory, production, demand and delivery) and also the Manager agent that coordinates and integrates the performance of all other internal agents, as well as provides interaction with the external environment. The agents plan, predict and collaborate in order to achieve the goal of maximizing profit. The Demand agent plays a central role in the system. Its main goal is to provide the most profitable customer order bundles taking into consideration changes in the customer demand, limited production capacity, limited inventory stocks, and unstable supply. The agent predicts customer order prices. Different methods for performing forecasts and approaches for setting customer offer prices are investigated and their influence on the overall system performance is studied. The experiments in the TAC SCM environment demonstrated that time-series forecasts and price predictions based on RFQ details and bidding history provide the best performance. The systems with these algorithms achieve similar scores when competing against each other. The system with the competitor price predictors comes next, and the approach of predicting price probabilities gives the lowest result. The same ranking order is observed when comparing the accuracy of the models' predictions. Thus, there is a strong correlation between the accuracy of price predictions and the total profit made: the higher the accuracy, the better the overall system performance.

Although the multi-agent approach has been applied by other researchers for designing their SCM systems, this paper offers an original demand-driven system architecture and scenario of its behaviour. The major contribution of our work is the development and comparison of the algorithms for predicting customer order prices that have not yet been applied in this domain. The algorithms demonstrated good performance in the TAC SCM game. What is more important, the models are designed in such a way that they are not associated with the game rules and thus can be used in other dynamic and competitive environments. Applying the algorithms in other domains is one of the next steps in our research. We also want to test our most accurate algorithms against the algorithms developed by other researchers. As 42 different predictive algorithms have been developed, it has been hard at this stage to compare them all against existing methods proposed in the literature. Another task for

the future work is to explore possibilities of applying other learning techniques to perform forecasts of customer order prices according to the strategies proposed in this paper. We also want to investigate how the behaviour of internal agents in the system can be further developed in order to improve the overall system's performance.

References

1. Benish, M., Andrews, J., Sadeh, N.: Pricing for Customers with Probabilistic Valuations as a Continuous Knapsack Problem. In: 8th International Conference on Electronic Commerce, Fredericton, Canada, pp. 38–46 (2006)
2. Benish, M., Andrews, J., Sardinha, A., Sadeh, N.: CMieux: Adaptive Strategies for Competitive Supply Chain Trading. ACM SIGecom Exchanges 6(1), 1–10 (2006)
3. Benisch, M., Greenwald, A., Grypari, I., Lederman, R., Naroditskiy, V., Tschantz, M.: Botticelli: A supply chain management agent. In: 3rd International Conference on Autonomous Agents and Multi-Agent Systems, New York, NY, pp. 1174–1181 (2004)
4. Booch, G., Jacobson, I., Rumbaugh, J.: OMG Unified Modeling Language Specification, Version 1.3., 1st edn. (2000)
5. Burke, D.A., Brown, K.N., Tarim, S.A., Hnich, B.: Learning market prices in real-time supply chain management. Computers and Operations Research 35(11), 3465–3478 (2008)
6. Castillo, O., Melin, P.: Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. IEEE Transactions on Neural Networks 13(6), 1395–1408 (2002)
7. Chaib-draa, B., Muller, J.: Multiagent based Supply Chain Management. Springer, New York (2006)
8. Chatzidimitriou, K.C., Symeonidis, A.L., Kontogounis, I., Mitkas, P.A.: Agent Mertacor: A robust design for dealing with uncertainty and variation in SCM environments. Expert Systems with Applications 35(3), 591–603 (2008)
9. Collins, J., Arunachalam, R., Sadeh, N., Eriksson, J., Finne, N., Janson, S.: The Supply Chain Management Game for the 2007 Trading Agent Competition. Technical Report CMU-ISRI-07-100, Carnegie Mellon University (2006)
10. Dong, R., Tai, T., Yeung, W., Parkes, D.C.: HarTAC – the Harvard TAC SCM '03 Agent. In: Trading Agent Design and Analysis Workshop, TADA-03, New York, NY, USA, pp. 1–8 (2004)
11. Fox, M.S., Chionglo, J.F., Barbuceanu, M.: The integrated supply chain management. Internal report of the Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto, Ontario, Canada (1993)
12. Ghani, R.: Price prediction and insurance for online auctions. In: 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 411–418. ACM Press, NY (2005)
13. He, M., Rogers, A., Luo, X., Jennings, N.R.: Designing a Successful Trading Agent for Supply Chain Management. In: 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems, Hakodate, Japan, pp. 1159–1166 (2006)
14. Ketter, W., Collins, J., Gini, M.: A Survey of Agent Designs for TAC SCM. In: Workshop for Trading Agent Design and Analysis, Chicago, USA (2008)
15. Ketter, W., Collins, J., Gini, M., Gupta, A., Schrater, P.: Identifying and Forecasting Economic Regimes in TAC SCM. In: 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, pp. 53–60 (2005)

16. Ketter, W., Kryznaya, E., Damer, S., McMillen, C., Agovic, A., Collins, J., Gini, M.: MinneTAC sales strategies for supply chain TAC. In: 3rd International Conference on Autonomous Agents and Multi-Agent Systems, New York, NY, pp. 1372–1373 (2004)
17. Kiekintveld, C., Miller, J., Jordan, P., Wellman, M.: Forecasting market prices in a supply chain game. In: 6th International Conference on Autonomous Agents and Multi-Agent Systems, Honolulu, Hawaii, USA, pp. 1318–1325 (2007)
18. Kovalchuk, Y.: Seller's Strategies for Predicting Winning Bid Prices in Online Auctions. In: International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, pp. 1–6 (2008)
19. Kovalchuk, Y., Fasli, M.: Adaptive Strategies for Predicting Bidding Prices in Supply Chain Management. In: 10th International Conference on Electronic Commerce (ICEC'08), Innsbruck, Austria, pp. 19–22 (2008)
20. Mitchell, T.M.: Machine Learning, International edn. MIT Press and The McGraw-Hill Companies, Inc. (1997)
21. Pardoe, D., Stone, P.: Adapting in agent-based markets: A study from TAC SCM. In: 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems, Honolulu, Hawaii, USA, pp. 677–679 (2007)
22. Pardoe, D., Stone, P.: Bidding for Customer Orders in TAC SCM: A Learning Approach. In: 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems, New York, NY, pp. 52–58 (2004)
23. Petric, A., Podobnik, V., Jezic, G.: The CrocodileAgent: Designing a robust trading agent for volatile e-market conditions. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2007. LNCS (LNAI), vol. 4496, pp. 597–606. Springer, Heidelberg (2007)
24. Saberi, S., Makatsoris, C.: Multi agent system for negotiation in Supply Chain Management. In: 6th International Conference on Manufacturing Research, Brunel University, UK, pp. 311–317 (2008)
25. Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)
26. Smith, K.A., Gupta, J.N.D.: Neural networks in business: techniques and applications for the operations researcher. Computers and Operations Research 27(11-12), 1023–1044 (2000)
27. Stan, M., Stan, B., Florea, A.M.: A Dynamic Strategy Agent for Supply Chain Management. In: 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Washington, DC, USA, pp. 227–232. IEEE Computer Society, Los Alamitos (2006)
28. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo (1999)
29. Yao, J.T., Tan, C.L.: A Case Study on Using Neural Networks to Perform Technical Forecasting of Forex. Neurocomputing 34(1-4), 79–98 (2000)