Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 35 (2008) 591–603
Expert Systems with Applications
www.elsevier.com/locate/eswa

Agent Mertacor: A robust design for dealing with uncertainty and variation in SCM environments ☆

Kyriakos C. Chatzidimitriou [a], Andreas L. Symeonidis [a,b,*], Ioannis Kontogounis [a], Pericles A. Mitkas [a,b]

[a] Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, GR541 24 Thessaloniki, Greece
[b] Intelligent Systems and Software Engineering Laboratory, Informatics and Telematics Institute – CERTH, GR570 01 Thessaloniki, Greece

Abstract

Supply Chain Management (SCM) has recently entered a new era, where the old-fashioned static, long-term relationships between involved actors are being replaced by new, dynamic negotiating schemas, established over virtual organizations and trading marketplaces. SCM environments now operate under strict policies that all interested parties (suppliers, manufacturers, customers) have to abide by, in order to participate. And, though such dynamic markets provide greater profit potential, they also conceal greater risks, since competition is tougher and request and demand may vary significantly in the quest for maximum benefit. The need for efficient SCM actors is thus implied, actors that may handle the deluge of (either complete or incomplete) information generated, perceive variations and exploit the full potential of the environments they inhabit. In this context, we introduce `Mertacor`, an agent that employs robust mechanisms for dealing with all SCM facets and for trading within dynamic and competitive SCM environments. Its efficiency has been extensively tested in one of the most challenging SCM environments, the *Trading Agent Competition* (TAC) SCM game. This paper provides an extensive analysis of `Mertacor` and its main architectural primitives, provides an overview of the TAC SCM environment, and thoroughly discusses `Mertacor`'s performance.

Keywords: Supply Chain Management; Machine learning; Autonomous trading agents; Electronic commerce; Agent intelligence

1. Introduction

Efficient management of the supply chain is becoming more and more critical in today's economy. The increasing possibilities of globalization, the facilitation of mediation between different and distributed parties through the Internet and the shift in production from *technology* and *product-driven* to *market* and *customer-driven*, among others, are factors that have caused a growth in the complexity of the supply chain. Thus, a transition has been observed from static and long-term relationships between partners to more dynamic trading strategies, increasing the potential of making profit in both ends of the supply chain.

In order to support such a transition, current trends in *Decision Support* (DS) *Supply Chain Management* (SCM) software tend to integrate *Supplier Relationship Management* (SRM), *Customer Relationship Management* (CRM), and *Enterprise Resource Planning* (ERP) primitives, in order to provide competitive business solutions. DS SCM software efficiently monitors and records all transactions, while supply chain strategies are applied at various stages of the process, in order to reduce cost and improve service levels (Levi, Kaminsky, & Levi, 2000).

Nevertheless, in such systems human expertise is imperative, usually leading to their deprecation, from advanced DS systems to mere transactional databases. In addition, the flourishing of virtual organizations and electronic

☆ Mertacor home page: http://issel.ee.auth.gr/MertacorWeb/.

* Corresponding author. Address: Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, GR541 24 Thessaloniki, Greece. Tel.: +30 2310 99 6349; fax: +30 2310 99 6398.
E-mail addresses: mertacor@olympus.ee.auth.gr, asymeon@iti.gr, asymeon@ee.auth.gr (A.L. Symeonidis).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.07.050

marketplaces, has led to the shift from traditional markets, to more dynamic SCM environments, where goods (raw material, end-products) are auctioned between interested parties (suppliers, manufacturers, customers), and advanced bidding strategies are employed in order to achieve optimal results. The structure of these auction environments requires computational strength and accurate timing, therefore implying the need for autonomous SCM solutions, which shall identify rapid market changes and handle them in a cost-effective manner, in order to profit from specific economical regimes. Nevertheless, these SCM solutions should also satisfy all security, safety and soundness issues that may rise in such uncertain environments.

Recent research literature acknowledges intelligent agents as the most appropriate technology for trading and auctioning in electronic markets (He, Jennings, & Leung, 2003). Additionally, the use of *Multi-Agent Systems* (MAS) has been proposed as a potential solution to handle the complex and distributed nature of the supply chain in an effective way (Chen, Peng, Finin, Labrou, & Cost, 1999; Sadeh, Hildum, Kjenstad, & Tseng, 1999). Equipped with smart strategies and efficient learning techniques, agents can provide robust solutions to deal with uncertainty and complexity. The more dynamic the SCM environment, the more intelligent the agent has to be to cope with the problem at hand.

In this context, we introduce `Mertacor`, an agent that employs robust mechanisms for trading within a dynamic and competitive SCM environment. `Mertacor` takes over all company activities, from component procurement and inventory management to factory scheduling and product sales among others, aiming to maximize company revenue. Through extensive analysis, a number of key points within the SCM process have been identified and incorporated into the agent's trading mechanism. By the use of heuristics, SCM business rules, scheduling algorithms, data mining techniques and fail-safe mechanisms, `Mertacor` proves extremely capable of trading with other entities, within a dynamic, multi-variate, uncertain, time pressuring and partially observable environment. Its efficiency has been extensively tested in one of the most challenging SCM environments, the *Trading Agent Competition* (TAC) SCM game (http://www.sics.se/tac), where research teams from all over the world compete against each other, in a real-world like economy, striving to maximize profit. In both 2005 and 2006 competitions agent `Mertacor` was one of the top-tier performing agents, while it has also been used by many other teams as a strawman in their controlled experiments, proving that its architecture is adequately robust to stand varying market pressure.

The rest of the paper is organized as follows: Section 2 provides an overview of the application domain, in order to familiarize the reader with the environment the agent was tested on. Section 3 gives a high-level sketch of `Mertacor`, along with a detailed description of the modules implemented and the techniques employed. Next, Section 4 illustrates the performance of `Mertacor` and points out the strengths and weaknesses of the current implementation, through both competition results and controlled experiments. Finally, Section 5 summarizes the work conducted, discusses ideas on future improvements, and concludes the paper.

2. Application domain analysis

In this section an overview of the TAC SCM game is provided. The official specifications of the game can be found at Collins et al. (2005). Within the TAC SCM environment (Arunachalam & Sadeh, 2005), agents act as *Personal Computer* (PC) manufacturers, willing to make profit in the simulated market, by competing with others on customer and supplier contracts. The TAC SCM game is played over the Internet and six agents take part in each game session. Game length is 220 "TAC" days, with each day lasting 15 s of real time (55 min/game). The entire simulation is controlled by a game server to which all participating agents connect.

Agents run their own factory unit, which has limited production capacity. Each day the game server sends *requests for quotes* (RFQs) to all agents on behalf of customers. Agents may make their offers to the customers based on their ability to satisfy delivery dates and reserve prices by sending a quote before the end of the day. The next day, if an agent's quote is a winning bid, the customer orders from that agent and, to get paid, the manufacturing agent must deliver the ordered PCs on-time. The manufacturing agent may either assemble the ordered PCs at that time or it can use PCs previously assembled and stocked in its inventory. Different types of PCs may be assembled, each requiring a different component compilation. Agents may procure components from eight (8) different suppliers by sending RFQs and issuing orders to the suppliers. If an agent defaults in delivering customer orders, it is billed with a penalty that is determined by the customer in its initial RFQ. Fig. 1 provides a schematic representation of the game. One may easily identify four (4) game phases: (i) component procurement, (ii) inventory management, (iii) production and delivery scheduling, and (iv) computer sales. Based on this observation, `Mertacor` follows the same architectural and functional partitioning, as presented in Section 3.

2.1. Component procurement

Agents sell computers that comprise four (4) components: CPUs, motherboards, memory and hard disk drives. In order to obtain these components at satisfactory prices, agents negotiate with suppliers, which are autonomous, revenue maximizing entities, willing to sell their entire capacity for the highest price. Their daily production capacity is calculated using a random walk function and the expected capacity for a future date on which they are to make an offer is the current capacity reduced by a portion reserved for future orders. A limited number of RFQs

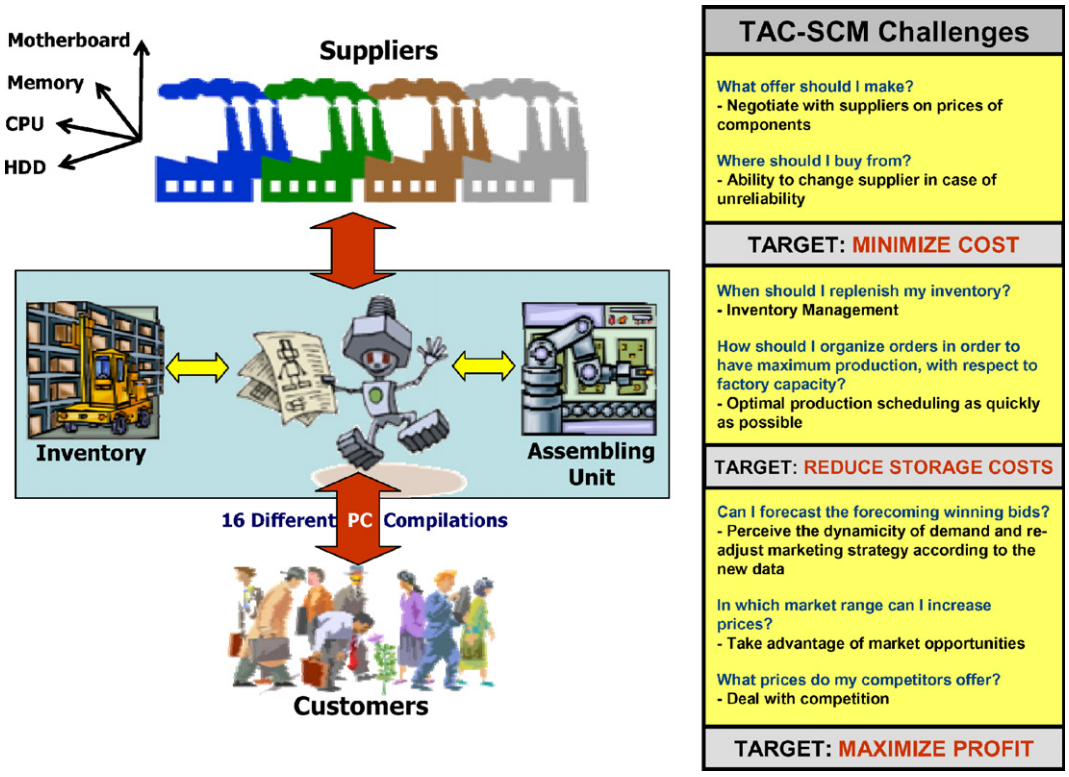

Fig. 1. An overview of the TAC SCM game.

is sent to suppliers, asking for specific components, at specific dates, along with the maximum price agents are willing to pay for them. Depending on their uncommitted capacity for that certain time period and on the reputation of the agent they doing business with (discussed below), suppliers may respond with offers, make no offer, propose alternative offers for a reduced quantity and/or a relaxed due date offer, or respond with a zero offer, in case they find the maximum price offered by the client as not satisfactory. A deal is then "closed" or broken.

In case a deal is established the agent has to send an order the following day, validating the deal reached. If not, an immediate impact in agent's *reputation*[1] against the specific supplier is applied. Agents rejecting supplier offers for large quantities will have a low reputation, thus being offered the highest prices and longest delivery times.

As price is directly dependent on committed production capacity and agent reputation, *it is obvious that component procurement is among the most challenging fields of the game*, where agents compete to ensure their constant feed with components.

2.2. Inventory management

Manufacturer agents in TAC SCM hold their individual components' and finished products' inventory. Since components are essential for PC production, the component inventory should always have the quantity required for on-time production of the customer orders. But component inventory is a shank cost at the end of each game, so agents should avoid buying more components than they can sell in the form of assembled products until the end of the game. Additionally, individual components and finished products in inventory are charged daily with a percent of the base price storage cost. It is, thus, obvious that keeping up a large inventory is not cost-effective, although useful for constant production. *A "happy medium" should be applied.*

2.3. Production and delivery scheduling

Manufacturer agents send two schedules to the game server every day, the production and the delivery schedule, in order to inform on their assembling status. The production schedule reports what types of PCs the factory unit has to assemble the following day. Having the required components, each PC requires from 4 to 7 factory cycles for production. Nevertheless, the factory unit has a limited production capacity (2000 cycles per day). A limited number of PCs can, thus, be assembled daily. All finished products are sent to the inventory, the day following.

The delivery schedule indicates which products to deliver to customers. Of course these products should either be in production or in the inventory. Deciding on which products to assemble and ship is highly related to penalties

[1] The trustworthiness of the specific agent. It is defined as a fraction of offered components that have been accepted by the agent.

and supplier defaults or large variations in lead times, and *strongly affects agent revenue*.

2.4. Computer sales

Customer demand in TAC SCM is also simulated by the game server. In a similar to supplier–manufacturer agents manner, customers negotiate with the latter on PC quantities and prices. Each TAC day, virtual customers send a bundle of RFQs to all the manufacturer agents, denoting the PC type, the quantity, the due date, the maximum price they are willing to pay for each unit and the penalty that would be paid for each day of late delivery. One of the most interesting features of TAC SCM is the variability of customer demand throughout a single game. Demand is split into three market ranges, (*low-mid-high*) with each of the 16 different PC types assigned to one of these ranges. Demand for each market range is determined each day with a different number of customer RFQs. It follows a Poisson distribution, having a mean parameter defined, multiplied by the current trend for each range, which is a random walk.

Manufacturer agents have no knowledge of the prices offered by competing agents; they only receive a daily price report indicating the minimum and maximum selling price for each PC type, as well as a market report every 20 days providing statistical information. Once an agent is assigned a customer order, it has to deliver on-time, or else it is charged with a penalty for each day the delivery is late. Orders that are five days late, are canceled.

The number of RFQs varies from 25 to 100/day for the *low* and *high* market ranges (Fig. 2) and from 30–120/day for the *mid* market range (Fig. 3). Keeping in mind that each RFQ may request up to 20 PCs, *one may identify that the number of PCs request each day may vary significantly* (165–4400 *PCs*).

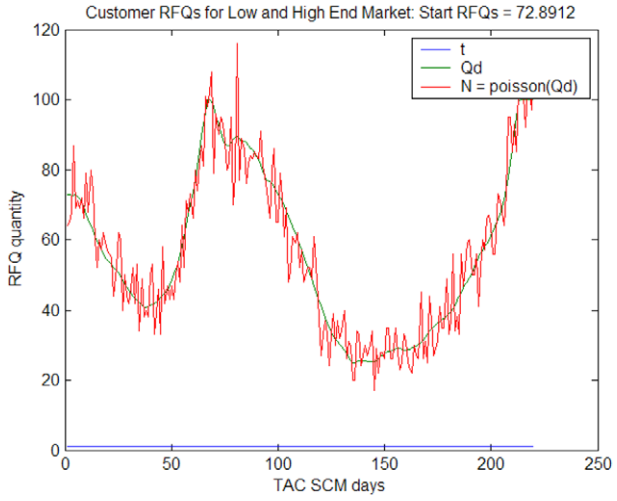

Fig. 2. Indicative customer RFQ demand for the *low* and *high* range markets.

Fig. 3. Indicative customer RFQ demand for the *mid* range markets.

2.5. Summary

From the discussion above, it is evident that the game is representative of a broad range of supply chain situations. It requires agents to compete concurrently in multiple markets, in order to procure components (supply side) and sell different products (sales side). The most intriguing feature of the game is that agents must act in a market with such large variations in demand and absence of information, making the need for a robust and adaptive bidding mechanism compulsory. Agents cannot get information about transactions made by other agents, including prices their opponents offered to customers or received by suppliers, others' inventories and bank balances. They do not have immediate or future knowledge of customer demand, nor supplier production capability.

To succeed, agents must slightly underbid on RFQs that seem profitable with respect to one's competitors, keeping in mind that very low bids are profitable only when cheap components are procured, which is not usually easy to accomplish. Another point in a successful selling strategy is to identify which market ranges or which PC types have high selling prices, resulting in high income. High selling prices for a specific product could be a result of reduced interest on behalf of competitors, or even a result of lack of components from the suppliers' side. Exploiting such economical regimes could lead to increased agent revenue. Nevertheless, in such adaptive strategies there is always the risk of being manipulated! Caution is, thus, imperative.

3. Agent Mertacor

In order to better manage and efficiently act on each one of the four TAC SCM facets (*Component Procurement*, *Inventory Management*, *Production and Delivery Scheduling*, and *Customer Bidding*), `Mertacor` employs a modular architecture, where each task is delegated to a specific mod-

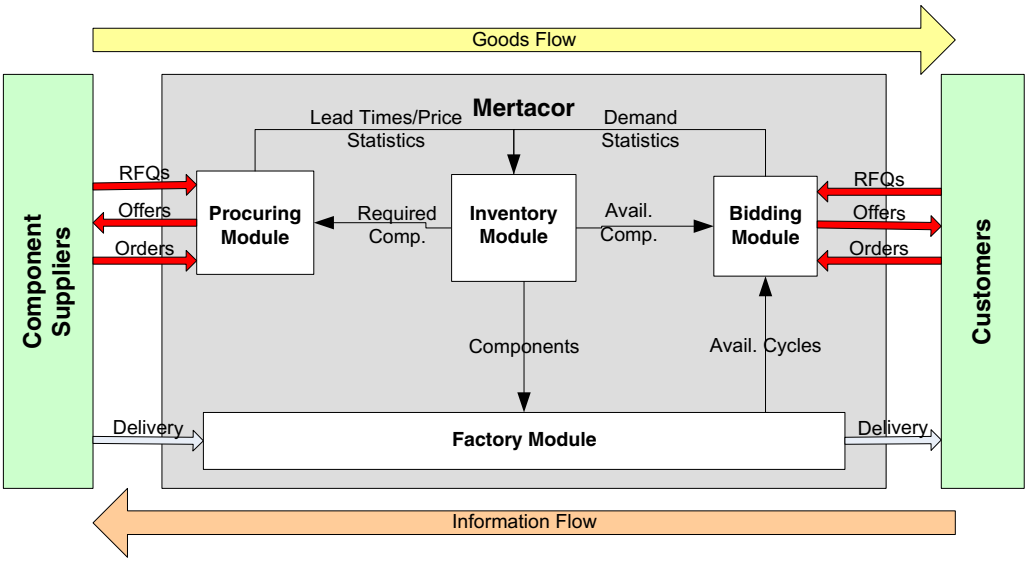

Fig. 4. The architecture of agent Mertacor.

ule, while all modules act in close collaboration. `Mertacor`, being a wrapper around the modules, ensures communication with suppliers and customers. Following other successful paradigms (He, Rogers, David, & Jennings, 2005; Pardoe & Stone, 2004), `Mertacor` exploits the integration of techniques from the *Operations Research* (OR) literature, heuristics and adaptive algorithms, as well as statistical modeling. The overall `Mertacor` architecture is illustrated in Fig. 4, where four core modules can be identified:

1. The Inventory Module (*IM*),
2. The Procuring Module (*PM*),
3. The Factory Module (*FM*), and
4. The Bidding Module (*BM*).

It should be denoted that such a modular architecture is not restrictive to the requirements of the game, *but can be easily applied to other SCM environments also*.

3.1. Agent modules

3.1.1. Inventory management

IM constitutes the cornerstone of one's supply chain structure, since all modules are highly dependent on it. SCM literature provides many paradigms of inventory management techniques, make-to-stock and make-to-order among others. `Mertacor` realizes an *assemble-to-order* system (ATO), a hybrid combination of the two aforementioned paradigms, which proves suitable in environments where assembly times are significantly smaller than replenishment times (Cheng, Ettl, Lin, & March, 2001), like in the case of the TAC SCM game. Here, suppliers cannot guarantee on-time delivery due to dynamic production capacity, while manufacturer agents are able to assemble products in just one "TAC" day. Taking also into account that an ATO system eliminates end-product inventory, reduces storage costs, improves forecast accuracy through demand aggregation and provides quicker response time for order fulfillments through risk pooling, ATO seems like the optimal choice.

An ATO system works as follows: The main goal of the system is to calculate certain inventory levels (thresholds) that need to be satisfied and below which replenishment is needed. The thresholds for each component are calculated in real-time using the following equation (Levi et al., 2000; Cheng et al., 2001):

$$R = D_{\mathrm{AVG}} L_{\mathrm{AVG}} + z\sqrt{L_{\mathrm{AVG}} D_{\mathrm{STD}}^2 + D_{\mathrm{AVG}}^2 L_{\mathrm{STD}}^2} \quad (1)$$

where D is the demand for a specific component,[2] L is the component's supplier lead time and z is a safety factor denoting the service level supplied by the user. Table 1 presents several service levels along with the corresponding safety factor.

Demand D is expressed in terms of products from the orders made by the customers. Average values and standard deviations are calculated over a period of 10 days from historic transactions. Statistics of product demand may vary from day to day, making the thresholds more unstable. This is both due to the variability observed each day in customer demand, as well as to the existence of the other competitive agents. Nevertheless, smaller variations can be achieved by approximating demand in terms of product ranges (*low-mid-high*).

[2] AVG is the average value, STD is the standard deviation of an attribute.

Table 1
Service level and the corresponding z value

Service level	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%	99.9%
z	1.29	1.34	1.41	1.48	1.56	1.65	1.75	1.88	2.05	2.33	3.08

Additionally, in ATO strategy no assembled product is stored in the inventory and, since components are shared along many products: (a) the levels of the component thresholds are lowered due to the aggregation of demands (b) internal component exchanges are applicable, in order to avoid late orders and penalties, and (c) lower storage costs are observed. Component demand is calculated on the range demand by applying the scheme in Fig. 5, which has been adjusted to the TAC SCM specs. Minimum and maximum levels are also coded as fail-safes. These levels are not static, but they are redefined throughout the game using a heuristic scaling, capturing mainly the start- and end-game effects.

Comp. ID	Low	Medium	High
1	0.4	0.33	0.0
2	0.0	0.17	0.6
3	0.6	0.17	0.0
4	0.0	0.33	0.4
5	0.4	0.5	0.6
6	0.6	0.5	0.4
7	0.8	0.5	0.4
8	0.2	0.5	0.8
9	0.6	0.5	0.4
10	0.4	0.5	0.6

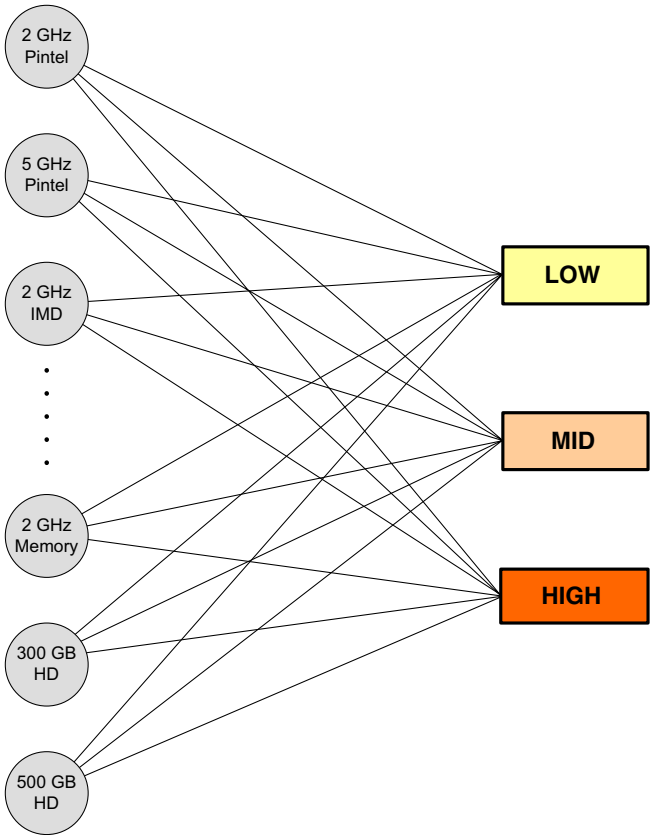

Fig. 5. Mapping range demand to component demand. The demand in each range is multiplied by the corresponding number to give the final component demand.

The aforementioned system has the ability to handle unique components for the product families and can be extended to become a *configure-to-order* system (CTO), where no pre-specified end-products exist and the customer can personally select the set of components (Cheng et al., 2001). *IM*, which is in charge of calculating reorder level values for each component, informs the *PM* to start negotiating with the suppliers for replenishment.

3.1.2. Component procurement

`Mertacor`'s performance is highly dependent on two factors: (a) having an inventory filled up with cheap components and (b) satisfying the inventory levels, since each delayed order implies a penalty and after five days, order cancellation. In order to cope with these requirements, we have developed a simple strategy, which is divided into two distinct phases: initial and normal-state.

`Mertacor` *initial-state procurement strategy* is followed for the first two days. The RFQs sent to the suppliers on Day-0 and Day-1 aim to build an initial inventory so that `Mertacor` can start production immediately and therefore start bidding for orders from the very first day of the game. The components procured are predictably expensive, nevertheless their usability is increased, since at the beginning of the game competition is not that strong, allowing for higher product prices to be achieved. On Day-1, another bundle of RFQs is sent, aiming to acquire relatively large quantities of cheap components for the forthcoming days.

On Day-2 `Mertacor` switches to a *normal-state procurement strategy*, designed to satisfy the aforementioned goals. It exploits all five available RFQs per component and per supplier each day, to maximize knowledge of the supplier's selling prices (probing). Based on this probing, the agent selects the most beneficiary offer to order the largest quantities. In general, RFQs sent by our agent can be divided into three categories:

- *Normal procurement RFQs*, aiming to satisfy inventory reorder levels and allow bidding for customer orders

in the near future. Quantities for these RFQs are calculated based on the reorder levels of the *IM*.

- *Critical procurement RFQs*, a special state in which our agent tries to procure components in order to satisfy critical customer orders.
- *Early procurement RFQs*, in an effort to obtain low-price components several days before they may be needed. These RFQs indicate quantities that if ordered, they would cause inventory to exceed reorder levels, so that there is no need for normal procurement after some posterior point in the game.

PM keeps track of the delays that may arise in supplier deliveries and maintains a real-time calculated average and standard deviation for every supplier, in order to inform *IM* on possible supplier delays or defaults.

3.1.3. Production and delivery scheduling

FM is responsible for: (a) generating production and delivery schedules, (b) adjusting inventory levels, and (c) adjusting available factory cycles. This module is also assigned the task of integrating the procuring, inventory and sales parts. The factory module is the most resource-demanding module of the agent. It implements a *factory simulator*, which simulates the operation of the factory for several days in the future, attempting to produce a draft of what will follow, based on knowledge on future (known or expected) supplier deliveries, customer deliveries, customer orders, production and delivery schemes. When the schedules are produced, they are communicated to the interested parties. The number of future days `Mertacor` is simulating is defined as *look-ahead time*. For the current implementation the look-ahead time has been specified to fifteen (15) days.

One of the most important tasks of the factory simulator is the projection of future component and product inventory. *FM* estimates future inventory needs by employing the iterative algorithm illustrated in Algorithm Block 1. This way `Mertacor` defines the expected inventory levels and factory cycles for the forecoming two weeks, and alerts the bidder not to exceed these levels.

Algorithm 1. Calculation of forecoming inventory

For each day of the look-ahead period starting from today:

(1) Update current inventory with supplier deliveries expected today (add components to be delivered on the specific date)
(2) Update inventory with products assembled from factory on the specific date according to the last production schedule submitted
(3) Remove components that are needed for next day's production
(4) Remove products that are to be delivered the next day to clients
(5) Remaining inventory is the starting inventory for next day

FM also releases the daily production schedule, which includes the orders that fit within the daily factory capacity (2000 cycles). The greedy scheduling procedure (Algorithm Block 2) is then employed, ranking all orders with respect to their added-value. In case too many orders are undertaken and the available capacity is exhausted, specific actions are taken.

Algorithm 2. Greedy scheduling algorithm

(1) Customer orders are sorted based on due date
(2) Orders with the same due date are sorted based on penalty
(3) Orders with similar due dates and penalties are sorted based on profit, that is unit price x quantity

After the orders are sorted, their production is arranged in the time-frame, based on the greedy recursive procedure described in Algorithm Block 3.

Algorithm 3. Greedy scheduling algorithm

(1) A multiple-unit order is split into smaller assembly processes, each for a single unit of the product type specified in the order
(2) A search into the future available capacity is submitted for each of these elementary assembly processes, starting from the following day until the last possible date for the on-time production of the order
(3) f all small assembly schedules can be arranged within the available future capacity, a delivery request is placed for the date that follows the production. This way the production of a single order may be spread along multiple dates of assembly scheduling

Potential orders that should be scheduled are also taken into account. The simulator runs a process to estimate the most profitable set of RFQs that should be bid upon, in order to maximize revenue. This is the most resource-demanding process employed by the agent and is divided in two parts: (a) the initial selection of the most profitable RFQs and (b) the *overbidding* algorithm.

When identifying profitable RFQs (which is the set of RFQs that in case they result in customer orders, they can be assembled and delivered on-time with the largest profit margin possible), not all available factory capacity for future days is committed to current RFQs, but a constantly decreasing fraction of the factory's nominal capacity is used. Thus, it is possible to save cycles for profitable RFQs, expected in the next days. *FM*, in cooperation with *BM*, apply the mechanism described in Algorithm Block 4, in order to determine if an offer should be submitted for a specific customer RFQ or not.

Algorithm 4. Customer RFQ processing

- *BM* finds an optimal bid price for each RFQ sent by the customers and sorts RFQs depending on their expected profit (the process followed by *BM* is thoroughly described on the next section)
- For each RFQ, the simulator places a production and delivery search-request like the request placed in the case of a customer order. The production process may be broken into multiple smaller unit assembly processes, only this time the request is marked as RFQ
- The simulator repeats the search process described in the previous algorithm for every elementary assembly process of the RFQ. If a production and delivery request can be established successfully, an acceptance signal is sent to the bidder, and the process continues with the next RFQ. In the opposite case, intermediate schedules are deleted and a rejection signal is sent

Following the initial RFQ selection, the *overbidding* algorithm takes place in order to maximize factory performance. The procedure implemented is based on the assumption that not all offers for customer RFQs would result in customer orders, thus some of the factory capacity would remain unexploited. The maximization of factory performance requires more offers to be submitted than the factory and the inventory can hold, in order to retain a stable production rate. But the extra capacity offered beyond the factory limit should not result in late deliveries and customer penalties. The mechanism implemented requires that the possibility of an offer submitted to result in a customer order is known with a relative certainty.[3] Given this possibility is known, the overbidding algorithm described in Algorithm block 5 is executed:

Algorithm 5. Overbidding algorithm

- For each RFQ initially selected by the selection algorithm, *BM* provides the estimated acceptance probability p
- According to the capacity committed to this RFQ, *FM* releases the amount that corresponds to the rejection probability $1 - p$
- The released capacity is returned to the capacity handler and is now available for the re-evaluation of the first RFQ that was rejected by the initial selection algorithm
- If the capacity released is sufficient for the assembly of the re-evaluated RFQ, an offer is submitted and the appropriate capacity is committed to the potential order. In the opposite case, the evaluation process continues with the next previously rejected RFQ

[3] The integrated mechanism for the possibility estimation is described in detail at the *BM* section.

As already specified, not all available factory capacity for future days is committed to current RFQs but a constantly decreasing fraction of factory nominal capacity is available for future production. The fraction changes dynamically and is calculated for each RFQ in the row, depending on the RFQ expected profit and the successive bids' ratio. Through this mechanism some extra factory capacity may be committed for a specific day, when an RFQ deems extremely profitable. Estimation is based on a linear expression, where minimum factory capacity reservation factor corresponds to maximum expected profit or minimum successive bids' ratio, while maximum factory capacity reservation corresponds to minimum RFQ expected profit. This feature of *FM* has proven quite useful, having the scheduler perform from 2000 (at the beginning of the game) to 20,000 simulation cycles each TAC day, when customer demand is high and more executions are required for order and RFQ processing.

3.1.4. Bidding

`Mertacor's` bidding strategy is focused on finding the optimal bidding price for each RFQ received and then deciding on which of these RFQs to claim, sorted on anticipated profit. Our initial hypothesis is that every bid `Mertacor` places will be successful, and in order to realize it, a bidding mechanism based on machine learning techniques has been implemented. Certain customer RFQ properties and the running SCM environment state are used to predict the winning quote for each customer RFQ and the probability of the customer accepting the quote. Training data are derived from logs of previously played games, while some simple, but effective fail-safe mechanisms were employed, in case of unexpected market changes, where the extracted predictive models became invalid. The developed bidding mechanism proved to be fairly accurate, abiding by the rules of a varying market and, thus, realistic and applicable in real-life SCM domains.

The problem is formulated as a regression problem, where the output of the knowledge model is the price the agent shall propose to a customer RFQ, and the inputs are several environment variables known to the agent. We started modeling by fitting a linear regression model, since our goal was to generate a simple but interpretable model on how inputs affect the output. Having identified such a model and taking into account only the most significant of inputs, we then explored more elaborate learning algorithms to achieve optimal performance.

Training datasets

The training data for predicting the winning price of a bid came from the two 2005 TAC SCM semi-finals and the test data from the two 2005 TAC SCM finals. In order to simulate steady-state behavior samples from Day-50 to Day-200 were gathered and bid prices below 0.5 of base price were removed, since no profit can be made out of such orders. Prior to any analysis, data were normalized

by subtracting the mean of the training data and dividing by the standard deviation of each attribute. Both the training and the test datasets were randomly split into 10 subsets, with approximately 75,000 samples each. Model efficiency is determined through the average *root mean square error* (RMSE) over the ten test sets.

Feature selection

Thirty eight (38) attributes were initially considered, both order-specific and game-specific. The basic set of attributes is based on the work of Pardoe and Stone (2005), but thirteen (13) additional inputs were also included for a more complete analysis. The most important of them are:

Order-specific	Game-specific
• due date in days • quantity • reserve price • penalty • base price of product, and • CPU brand	• maximum prices of the product for the past five days • minimum prices of the product for the past five days • current date • current demand

Additionally, the average product price, the total demand and orders for a specific product, the total number of supplies delivered for the components of a product, and the average supplier's price for the components of a product were also included. All the additional features are calculated over the last twenty days and are available through the market reports of the game server.

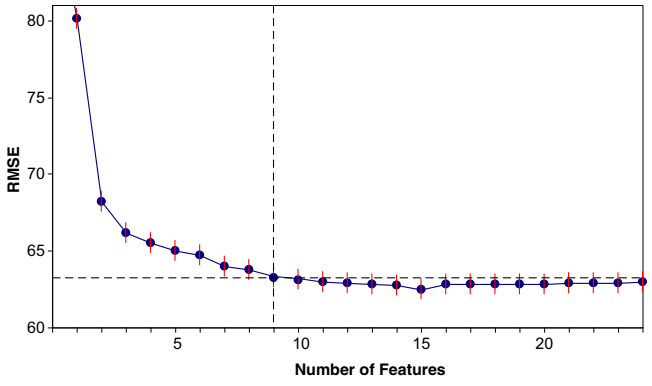

Fig. 6. The least complex model was chosen that was within one standard error from the minimum.

Table 2
The final nine attributes used to predict winning prices. These are: the RFQ's `due date`, its `reserve price`, the `highest` and `lowest` prices for the previous `two` and `four` days respectively, and the `current demand` of PCs

Feature	β value	β Value normalized
Due date	−8.066	−0.053
Reserve price	14.978	0.0985
High-1	151.902	1.00
High-2	58.665	0.386
Low-1	59.511	0.392
Low-2	30.834	0.203
Low-3	19.177	0.126
Low-4	14.505	0.095
Current demand	10.769	0.071

Feature selection is performed using *multiple linear regression* (MLR) and *backward elimination* (BE) based on the *F*-statistic measure defined as

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N - p_1 - 1)} \quad (2)$$

where RSS_1 is the residual sum of squares of the larger model with p_1 variables and RSS_0 the previous model with p_0 variables.

Applying this technique, BE sets off with the full set of attributes and removes the one with the lowest *F*-statistic value. The process is re-applied until no attribute is left.

The most parsimonious model within "one standard-error" (the one with the minimum average RMSE) was selected (Hastie, Tibshirani, & Friedman, 2001), leaving a model with only nine (9) out of the 38 initial predictors. The above described feature selection procedure is depicted in Fig. 6, while Table 2 denotes the optimal β coefficients for the MLR model.

The third column of Table 2 illustrates the normalized β values of the inputs, which are directly comparable, and their signs indicate the correlation between the input and the output. Interesting rules may be derived by observing the β values, i.e.

- The higher the highest and lowest prices for the past two days, the higher the current price.
- The later an order is due, the lower its price.
- The bigger the reserve price, the higher the offer to the customer.
- The higher the demand from the customer-side, the higher the prices, since there is less competition.

One may also identify that as we go further into the past, the lesser significance the high and low prices have.

Model selection

Besides MLR we also applied three other classification (regression) schemas, in order to decide on the one that optimally meets the problem of predicting the winning bid of an order: a Neural Networks, b SMOreg (Support Vector Machines) and, finally, c the M5′ algorithm (Quinlan, 1992). An indicative comparison between the four models on one of the testing datasets is provided in Table 3.

Table 3
Comparison of the linear model and M5′

Model	Average RMSE	Standard deviation
MLR	63.66	0.63
NN	67.31	0.72
SMOreg	62.37	0.58
M5′	57.11	0.48

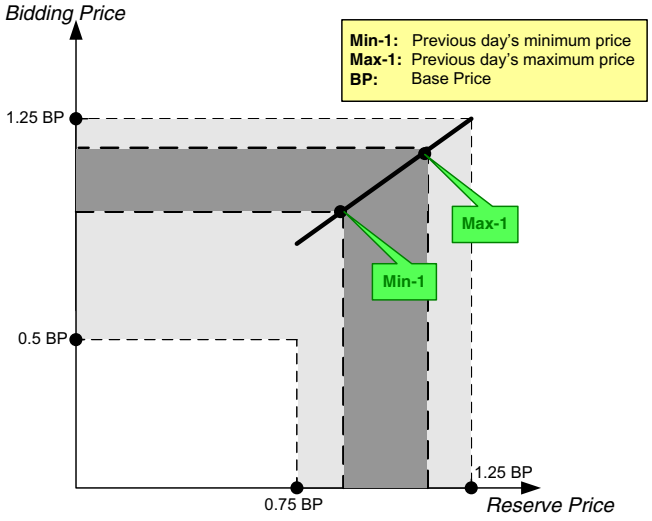

Fig. 7. The linear interpolation performed between the minimum and maximum prices of the product as given the previous day.

After extensive testing the M5′ model was selected. Further analysis can be found at Symeonidis, Nikolaidou, and Mitkas (2006).

Apart from the off-line mechanism for predicting the winning price of a bid, *BM* also incorporates two on-line modeling mechanisms that may be considered either as fail-safe or up-to-date mechanisms: (i) an ad hoc mechanism designed to function complementary to the prediction models, handling unexpected circumstances of selling prices, and (ii) an overbidding mechanism to help with filling the capacity levels provided by the scheduler.

Fail-safe mechanism

The former, named the `Follower` for its ability to follow prices on-line, evaluates the minimum and maximum prices for PCs ordered the previous day as stated in the daily price reports, and provides a prediction of the approximate level of winning bid price for each RFQ. The `Follower` deploys linear interpolation (see Fig. 7), based on the assumption that the maximum price paid corresponds to the maximum customer RFQ reserve price, while the minimum price paid corresponds to the minimum RFQ reserve price. Let P_M be the model price and P_F the follower price. Then, the final bid price placed is calculated as follows:

$$\text{if } (|P_F - P_M|/P_M)\% < \text{threshold}(10\%),\ P_F,\ \text{else},\ P_M \quad (3)$$

Experimental results showed a 20% improvement of the fail-safe mechanism in RMSE accuracy compared to other on-line naive mechanisms (Dahlgren & Wurman, 2004). The `Follower` has significantly helped `Mertacor` through sudden market changes, especially at the start- and end-game periods, when the game unfolds in unpredictable manners. Additionally, an extra increase of 13% in RMSE accuracy was measured, when applying the off-line/on-line mechanism combination.

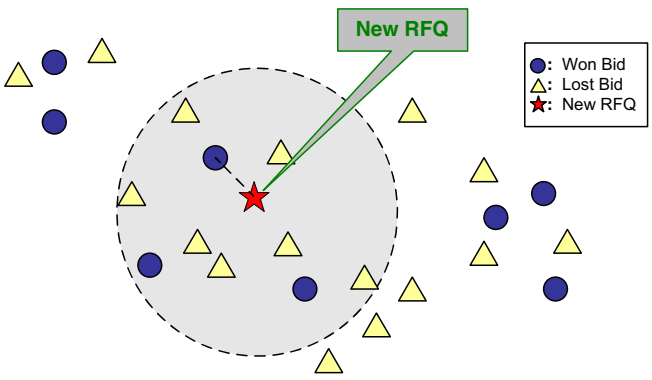

Fig. 8. Clustering RFQs with respect to their probability of becoming an order.

Overbidding

As far as overbidding is concerned, a scheme using the *k*-Nearest Neighbors (*k*-NN) algorithm (McQueen, 1967) was developed to produce a probability of acceptance for each bid placed (Fig. 8). Having identified the probability of an RFQ becoming an order, the bidding module signals the scheduler to commit only the fraction of the capacity that corresponds to that probability, letting the remaining capacity for the next RFQ in the row. The probability is calculated as the fraction of the n neighbors that became orders versus the total number of neighbors $k(n/k)$. After extensive testing, a value of $k = 10$ was selected for the TAC SCM game. The neighbors/exemplars are RFQs sent to the customers the previous day(s) tagged either as accepted or rejected. The set of attributes used are the attributes selected for the off-line model.

4. Results

4.1. Competition results

The agent presented in this paper participated in the TAC SCM 2005 and 2006 competitions and performed well in the different stages of both competitions. The results of the preliminary rounds of the 2005 game (qualifyings and seedings) indicated that `Mertacor` illustrated improved efficiency when competing against "tough players". This hypothesis was validated during the final rounds, where the games played were much more competitive. `Mertacor` placed 1st in the quarter-finals and 3rd in the semi-finals; at the finals `Mertacor` came up against the other 5 best scoring agents and finished 3rd, with a positive bank balance (see Table 4).

Even though numerous games have to be played in order to assess the "true" value of an agent and its gaming capacity with respect to other players, interesting results can come up by going through Table 4, helping us identify some of the strong points and drawbacks of our implementation. First of all, the ATO system employed along with the procurement strategy followed, resulted to the *lowest storage costs*, *high delivery performance rates*, and *low mate-*

Table 4
Mean skills of the agents in the 2005 finals for a total of 16 games

Agent	Score ($)	Revenue ($)	Material ($)	Storage ($)	Del. (%)	Util. (%)
TacTex-05	4.741M	108.586M	100.614M	2.013M	97.75	87.81
SouthSCM	1.604M	108.246M	102.375M	2.843M	98.06	87.75
Mertacor	546272	75.582M	72.639M	1.730M	98.88	60.63
Deep Maize	−220503	107.681M	103.309M	2.645M	97.31	85.13
MinneTAC	−311844	81.903M	79.728M	1.887M	99.88	65.00
Maxon	−1.985M	71.105M	68.588M	3.520M	100	56.19

The skills are: final bank balance (Score), revenue, cost of components, storage costs, delivery performance and factory utilization.

Table 5
Mean skills of the agents in the 2006 second finals for a total of 16 games

Agent	Score ($)	Revenue ($)	Material ($)	Storage ($)	Del.(%)	Util. (%)
Mertacor	10.818M	102.451M	89.760M	1.667M	99.88	82.13
CMieux	7.799M	107.191M	98.635M	1.600M	100	87.75
SouthSCM	5.216M	99.484M	92.252M	1.681M	99.94	80.44
UMTac-06	3.727M	80.885M	75.508M	1.249M	98.56	64.69
jackaroo	180.145	92.196M	89.204M	2.277M	100	73.88

The skills are: final bank balance (Score), revenue, cost of components, storage costs, delivery performance and factory utilization. The sixth agent did not compete.

rial costs. One may argue that the last metric also accounts for the inability to compete for orders, but once put into perspective of `Mertacor's` performance, satisfactory inventory management is implied. In addition, *BM* ensures a *high Average Selling Price* (*ASP*) for our agent (2nd with a 0.776 normalized *ASP* – 0.780 for agent Maxon). `Mertacor 2005` major drawback was the *expensive contracts with suppliers*, placing it 6th, with 0.726 average normalized CPU buying price (the most expensive component – 0.694 for agent SouthamptonSCM). Another bottleneck identified was the *low factory utilization* (equivalent to low revenue) that can be interpreted to *low throughput* (rate of products out of the factory versus components in the factory), causing additional reduction to profit. *A balance between high selling prices and high throughput was, thus, imperative*.

Table 6
RMSE of the follower and the M5′ model used in the bidding process

Round	Game ID	Follower	M5′	Mertacor's mechanism
Second finals 2006	6215	108.99	*86.63*	97.17
	6220	123.80	*95.99*	112.39
	6222	122.66	*99.31*	11.54
	8634	100.57	*84.93*	92.52
	8641	154.17	*117.04*	138.01
Average		113.53	**90.90**	102.07
Finals 2006	5122	63.49	69.15	*62.91*
	5129	70.75	79.91	*67.56*
	5605	*59.18*	65.10	59.65
	5608	*73.13*	73.24	73.49
	5612	65.27	66.18	*64.71*
Average		**64.68**	70.46	64.71

As far as the 2006 competition is concerned, `Mertacor` came with a much better strategy regarding low throughput and placed in a much better place in the seeding rounds than in 2005. After advancing from the quarter-finals with the 2nd biggest balance, agent `Mertacor` was not able to proceed to the finals, due to a network failure. In the second finals, where agent `Mertacor` eventually participated, it won the 1st place (see Table 5).

Using the *Supply Chain Trading Analysis and Instrumentation Toolkit* (Benisch et al., 2005) we analysed certain games from the 2006 semi-finals and finals. In all games *the early procurement strategy yielded satisfactory prices* for the components provided by the suppliers. As far as the *ASP* is concerned, `Mertacor` still held *one of the highest ASPs*, which in relation with the extremely low component prices achieved in the second finals led to the first place.

4.2. Experimental results

In order to measure the performance of the bidding module, we tested it against both the `Follower` and the M5′ model for all the transactions between agents and customers in the semi-finals and finals of 2005 and 2006 competitions. Table 6 provides a qualitative comparison of the three regression (price prediction) schemas for the 2006 finals and second finals games. The M5′ model employed was the one tuned in order to be used in the 2006 second finals. One may observe that M5′ outperforms the `Follower` when tested against the 2006 second finals games, since this model was extracted for the specific competitors and for this specific game stage. This was not the case, though, when a different environment was addressed. When tested against the 2006 finals games, M5′ performed

poorly, making the added-value of the `Follower` apparent. As illustrated in Table 6, the implemented synergy of the two schemas (`Follower` + M5′) into the *BM* of `Mertacor` ensures an acceptable performance for the agent in all game environments, even though unexpected differentiations from the modeled one may occur (excessive demands, agents defections, etc.). In general, it is common knowledge that after the 2005 competition, the bidding mechanisms have reached their limit in the sense that they do no longer add any more value to the final results (Jordan, Kiekintveld, Miller, & Wellman, 2006). Based on that, the bidding mechanism describe here should be adequate for future developments of our agent and the focus should be given on other facets of the supply chain.

Analysis presented in Jordan et al. (2006) has also shown that agent Mertacor had the smallest *bullwhip effect* with respect to other agents participating in the 2005 finals. No correlation was found, though, between the final profit and the bullwhip effect, but still this is an indication of the high performing inventory and of the successful management of demand and supply, by our agent. Additionally, another game theoretic analysis, which can be found in

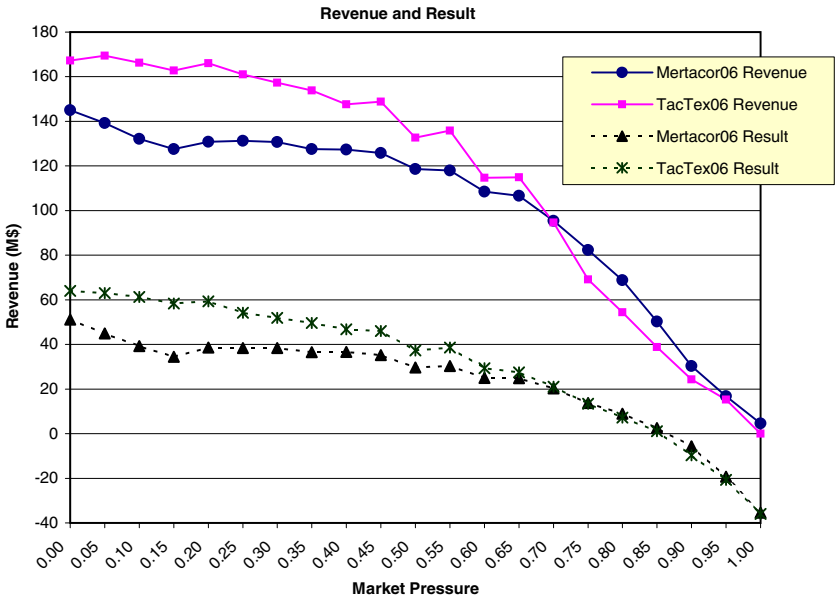

Fig. 9. The profit (revenue) and final balance (result) for Mertacor and Tac-Tex, with respect to market pressure (ranging from 0% to 100%).

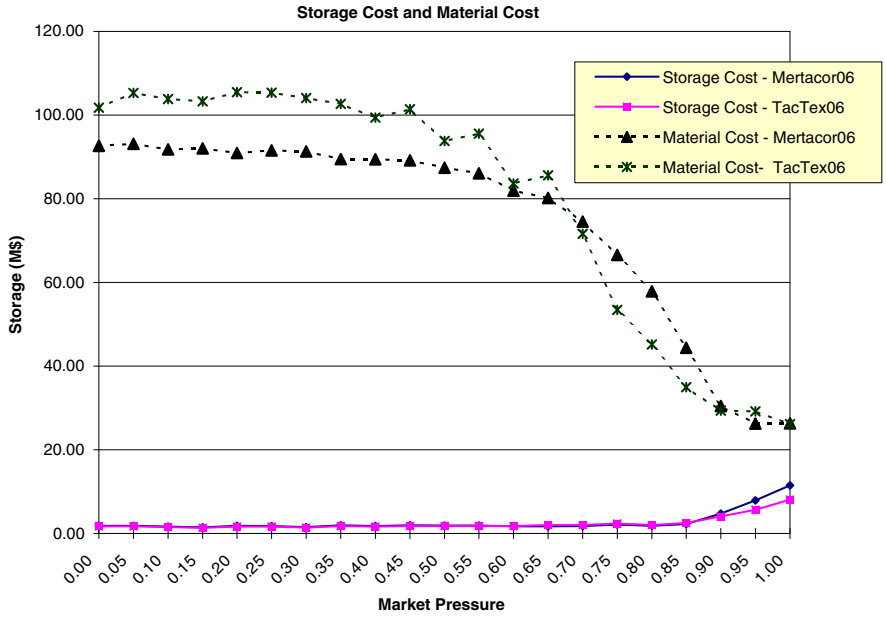

Fig. 10. The costs of materials and the storage cost for Mertacor and Tac-Tex, with respect to market pressure (ranging from 0% to 100%).

Wellman, Jordan, Kiekintveld, Miller, and Reeves (2006), displays Mertacor 2005 to be strong in a wide variety of contexts, along with PhantAgent, 2006 runner-up.

Finally, a controlled experiment was performed using the same idea as in Borghetti, Sodomka, Gini, and Collins (2006). Agent Mertacor was tested against TacTex, winner in the TAC SCM 2005 and 2006 competitions and three "dummy" agents provided by the server. All these agents were tested against a market share control agent, in order to monitor their behavior in a varying environment. The market share control agent was set to win a pre-specified market percentage each day by bidding below the 0.5 times the base price threshold, where no profit can be made. The results of this scenario are depicted in Figs. 9 and 10. For each market percentage won by the market share control agent (ranging from 0–100% in 5% increments) ten (10) games were played in order to eliminate randomness and the result is the average of the final scores. Agents Mertacor and TacTex appeared to outperform (in terms of revenue) each other at different market pressure levels, with Mertacor being more capable when the market share left is low and TacTex when the market is open for the three agents to bid more freely. This validates the observation that agent Mertacor is more efficient in more competitive environments, while its throughput limits its performance when the game is more "open". Storage costs follow an analogous to revenue manner, while material costs were similar for both agents.

5. Conclusions and future work

In this paper, we have introduced Mertacor, a SCM agent designed to participate in the TAC SCM game. What was presented here is the final agent design based on the experience gained over the past two years. The agent employs a combination of OR, heuristic and statistical modeling techniques, in order to manage a wide range of activities in an efficient manner. The architecture proposed is generic, and can be applied to other SCM environments also. Focusing on specific points, one can identify that the inventory management system, designed for the PC assembly line, performed very well, in an uncertain and dynamic environment, outside the assumptions made by the authors. The learning models were able to capture the dynamics of the markets at hand, while the heuristics applied to the supplies and the factory modules worked satisfactorily for the agent that performed well in the competition. Future research work on Mertacor includes the development of more accurate predictors on the behavior of both customers and suppliers. That, along with some improvements in the heuristics, would allow a bigger factory throughput, which is the confining factor for Mertacor. Finally, identification of opportunities throughout the game seems to be one of the most significant pilots in future enhancement of Mertacor.

References

Arunachalam, R., & Sadeh, N. (2005). The supply chain trading agent competition. *Electronic Commerce Research and Applications, 4*, 63–81.

Benisch, M., Andrews, J., Bangerter, D., Kirchner, T., Tsai, B., & Sadeh, N. (2005). Cmieux analysis and instrumentation toolkit for TAC SCM. Technical Report CMU-ISRI-05-127, School of Computer Science, Carnegie Mellon University, September.

Borghetti, B., Sodomka, E., Gini, M., & Collins, J., 2006. A market-pressure-based performance evaluator for TAC-SCM. In *AAMAS-06 workshop on trading agent design and analysis (TADA/AMEC)*.

Chen, Y., Peng, Y., Finin, T., Labrou, Y., & Cost, S., 1999. A negotiation-based multi-agent system for supply chain management. In *Workshop on agent-based decision support in managing the internet-enabled supply-chain atagents'99* (pp. 15–20).

Cheng, F., Ettl, M., Lin, G., March 2001. Inventory-service optimization in configure-to-order systems. Technical Report RC 21781, IBM.

Collins, J., Arunachalam, R., Sadeh, N., Ericsson, J., Finne, N., & Janson, S. (2005). The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-05-132, CMU, November. http://www.sics.se/tac/tac06scmspec_v16.pdf.

Dahlgren, E., & Wurman, P. R. (2004). Packatac: A conservative trading agent. *SIGecom Exchanges, 4*(3), 33–40.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. Springer, URL http://www-stat.stanford.edu/tibs/ElemStatLearn/.

He, M., Rogers, A., David, E., & Jennings, N. R. (2005). Designing and evaluating an adaptive trading agent for supply chain management applications. In *IJCAI-05 workshop on trading agent design and analysis* (pp. 35–42). http://www.sics.se/tada05/.

He, M., Jennings, N. R., & Leung, H. (2003). On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering, 15*(4), 985–1003.

Jordan, P. R., Kiekintveld, C., Miller, J., & Wellman, M. P. (2006). Market efficiency, sales competition, and the bullwhip effect in the TAC SCM tournaments. In *AAMAS-06 workshop on trading agent design and analysis (TADA/AMEC)*.

Levi, S. D., Kaminsky, P., & Levi, S. E. (2000). *Designing and managing the supply chain*. Illinois: McGraw-Hill.

McQueen, J. B. (1967). Some methods of classification and analysis of multivariate observations. In L. M. L. Cam, & J. Neyman (Eds.), *Proceedings of fifth Berkeley symposium on mathematical statistics and probability* (pp. 281–297).

Pardoe, D., & Stone, P. (2004). TacTex-03: A supply chain management agent. *SIGecom Exchanges: Special Issue on Trading Agent Design and Analysis, 4*(3), 19–28http://www.cs.utexas.edu/pstone/Papers/bib2html/.

Pardoe, D., & Stone, P. (2005). Bidding for customer orders in TAC SCM. In P. Faratin & J. Rodriguez-Aguilar (Eds.), *Agent mediated electronic commerce VI: Theories for and engineering of distributed mechanisms and systems (AMEC 2004). Lecture notes in artificial intelligence* (Vol. 3435, pp. 143–157). Berlin: Springer-Verlag http://www.cs.utexas.edu/pstone/Papers/bib2html/.

Quinlan, J. R. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence* (pp. 343–348).

Sadeh, N., Hildum, D., Kjenstad, D., & Tseng, A. (1999). Mascot: An agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In *Workshop on agent-based decision support in managing the internet-enabled supply-chain at Agents'99* (pp. 133–138).

Symeonidis, A. L., Nikolaidou, V., & Mitkas, P. A. (2006). Exploiting data mining techniques for improving the efficiency of a supply chain management agent. In *2006 IEEE/WIC/ACM international conference workshop on interaction between agents and data mining*.

Wellman, M. P., Jordan, P. R., Kiekintveld, C., Miller, J., & Reeves, D. M. (2006). Aamas-06 workshop on game-theoretic and decision-theoretic agents. In *AAMAS-06 workshop on game-theoretic and decision-theoretic agents*.