Learning market prices in real-time supply chain management

David A. Burke, Kenneth N. Brown, S. Armagan Tarim,

Centre for Telecommunications Value-chain Research and
Cork Constraint Computation Centre,
University College Cork,
Ireland

Brahim Hnich

Faculty of Computer Sciences,
Izmir University of Economics,
Turkey

Abstract

This paper proposes a model for dynamic pricing that combines knowledge of production capacity and existing commitments, reasoning about uncertainty and learning of market conditions in an attempt to optimise expected profits. In particular, the market conditions are represented as a set of probabilities over the success rate of product prices, and those prices are learned online as the market develops. The dynamic pricing model is integrated into a real-time supply chain management agent using the Trading Agent Competition Supply Chain Management game as a test framework. We evaluate the agent experimentally in competition with other supply chain agents, and demonstrate the benefits of incorporating more market data into the dynamic pricing mechanism.

Email addresses: d.burke@4c.ucc.ie (David A. Burke), k.brown@cs.ucc.ie (Kenneth N. Brown), at@4c.ucc.ie (S. Armagan Tarim), brahim.hnich@ieu.edu.tr (Brahim Hnich).

1 Introduction

Supply chain management involves planning and coordinating a range of activities across the supply chain, from raw material procurement to finished goods delivery. Shorter product life cycles, changing and uncertain market trends, and the emergence of e-market places is resulting in a shift in how supply chains are managed. Today's supply chains are generally based on long-term relationships between key trading partners. Introducing more flexible and dynamic mechanisms offer the possibility of more efficient relationships between suppliers and customers that respond to changing market conditions. In a dynamic supply chain, the market is the driving force, and there is less emphasis on relationships with supply chain partners. In this paper, we take the perspective of a single self-motivated trading agent in a market-oriented supply chain.

In a typical supply chain environment, the procurement costs and availability of raw material change over time. The demand is typically uncertain and thus changes over time. Consequently, as competitors enter or leave the market and improve their processes, the market price for finished goods will also change over time. One of the key enablers for an effective supply chain trading agent is the ability to change prices in response to changing market and supply chain conditions. The price offered to a potential customer must be high enough to make a profit over the cost of procurement, inventory, production and delivery, but must be low enough to be profitable to the customer, and must be lower than comparable offers from competitors. In improving inventory management practice and balancing demand and supply, the determination of the optimal price to charge a customer over time for a product whose demand is uncertain and whose supply is limited is a complex task, known as the *dynamic pricing* problem. Charging travellers different fares for the same flight and the pricing of hotel rooms, fashion goods, and discontinued or left over products are examples of dynamic pricing. While the types of pricing policies and methods used in the exchange of goods and services vary greatly, they fall into two broad categories: *posted-price mechanisms* and *price-discovery mechanisms* (bidding processes such as auctions) [1]. Under a posted-price mechanism, goods are sold at take-it-or-leave-it prices determined by sellers.

Much of the dynamic pricing literature deals with maximising expected revenue on sales of perishable goods. In this context [2] investigates the use of dynamic programming with incomplete state information to jointly estimate demand and set prices as time evolves. Parameters for a demand function are learned over time using least squares estimates applied to observed demand and prices. A similar problem is investigated in [3] where a probability distribution is used to model customer demand relative to prices offered and is updated using real-time sales data. They show that their strategy is also ro-

bust when the demand predictions are inaccurate. [4] examines the robustness of dynamic pricing models in greater detail. [5] looks at dynamic pricing in the context of learning customer demand for information goods sold on the Internet. They use a Bayesian model of demand uncertainty and provide a number of algorithms for updating this model after observing sales data. An example of dynamic pricing in the context of prices posted online by e-businesses is [6], where a number of algorithms incorporating game theory and machine learning are evaluated. The effect that production strategies have on pricing is investigated in [7], where they examine how the range, choice and design of products offered plays a part in determing the prices charged for products. The reader is also referred to Elmaghraby and Keskinocak [1] for a more comprehensive review of dynamic pricing with inventory considerations as well as a general review of dynamic posted pricing literature.

In this paper we employ the dynamic posted-price mechanism approach for dynamic pricing. Our key idea is to combine reasoning about the probability of acceptance with reasoning about supply, inventory and production constraints. We represent market conditions implicitly, creating a probability distribution over the prices acceptable to the customers. This price information is then fed into a production/inventory planning model. The aim of this model is to select a subset of tenders and assign one of those prices to each one, to maximise the expected profit, while ensuring that the expected set of accepted offers can be produced by obeying all inventory and production constraints. This abstract model allows us, in principle, to participate in a variety of sealed-bid markets, whether there is one customer or many, one encounter or many, one product or many, and one agent or many competitors. We can then learn these probability distributions as we participate in the market, observing the success or failure of our bids. Specifically, we fix target probabilities, and learn the prices which would be accepted with those probabilities. We study four dynamic pricing strategies of increasing complexity, making more informed decisions about the selected tenders and the prices we offer. To verify our approach, we implement an agent to participate in the Trading Agent Competition Supply Chain Management (TAC–SCM) [8] game, and we show that in competition with a portfolio of third-party agents, the most informed model increases the average profits over a series of simulations.

Note that procuring contracts through dynamic posted prices can be modelled using game theory (e.g. [9]), and it has been shown that approximating the behaviour of competitors in such games can mean that the Nash equilibria are not achievable [10]. However, the nature of real-time dynamic supply chains means much significant information will be unavailable. For example, in a single tender process, the identity of competitors, their cost structures and reward functions, and the number of competitors may all be unknown. Further, a supply-chain agent may not even know its own cost structures, since the availability and price of components may be unknown. If we regard each tender

process as a round in a larger game, then decisions taken in a single round directly affect costs and capabilities in future rounds, but the game parameters for future rounds are unknown (i.e. customer demand), and the actions of competitors in previous rounds may also be unknown. For all these reasons, we suggest that any game-theoretic approach applied to realistic dynamic supply chains will be forced to rely on many approximations. In this paper, we take an alternative approach, and use the repeated rounds to approximate the missing information. The only aspect of the market that we can be sure of is the result of our previous actions (i.e. the successes or failures of our previous bids), and thus we use this as the basis of our approximation, learning prices that have given probabilities of being accepted.

The TAC–SCM competition was staged for the first time in 2003. There have been many different approaches to pricing strategies in the competitions to date. Game theory and a variation of the Cournot game is used in [11] to model the customer market. They investigate the relationship between market demand and product price and use this as a way to predict market-clearing price. Kiekintveld et al. [12] perform game theoretic analysis to factor out the strategic aspects of the environment and to define an expected profitable zone of operation. They then use feedback from the market to dynamically adjust both their sales, procurement and production strategies in an attempt to stay within this zone. A mathematical model similar to that which we propose in this paper is used in [13]. Their model tightly couples the bidding and production scheduling decisions of the agent while trying to maximise expected revenue. However, they do not consider component costs which could lead to below cost selling. As with our approach, they also include the probability of being successful with a bid as an input into their model - the probability distribution is derived using linear regression on recent information gained from the market place. A particularly interesting approach is used in [14] where the sales task is looked at as a multi-period optimisation problem. Both multi-stage stochastic programming and Markov Decision Processes are used to consider the impact of current decisions in future periods. Many agents also include some method for estimating the probability of offer acceptance in their algorithms. Ketter et al. [15] examine two different strategies. An attempt to use online learning was not successful due to the lack of available online information (an issue which we address in this paper). A second technique in the same paper assumes that the shape of the probability curve remains the same and uses information learned from the market to shift the probability curve as a whole. Daily market price reports, containing the day's low and high prices, are used in [16] for probability estimates. These probability estimates just consider product type, which is considered to be the most important parameter affecting price. This is extended to also consider the due date of customers requests by introducing multipliers which are learned online comparing predicted and actual orders received for different due dates. Several machine learning approaches to estimating the probability of bid acceptance are compared in [17].

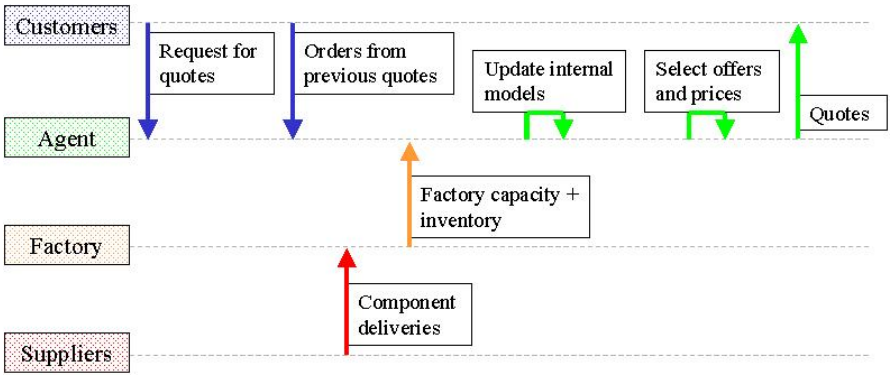


Fig. 1. Timeline of our agent's daily reasoning.

Here, purely historical data is used to learn the probability of acceptance and the impact of different techniques is measured on agent performance.

2 A generic real-time supply chain management model

We consider an intermediate agent in a dynamic supply chain. We assume that the problem involves a series of sealed-bid first-price auctions, in which one or more customers request quotes, the agent selects a subset of the requests and offers a price on each one, and then discovers which of them have been successful. The agent must then supply finished products by a stated due-date. The agent must procure components, arrange for their storage, produce the finished goods, and deliver them. In this general scenario, we make no assumptions about the number of customers, the number of competing intermediate agents, the production process, the inventory management, or the procurement problem. The aim of the agent is to determine the best prices for the best subset of tenders over a predefined period, with the ultimate aim of maximising expected profit. Figure 1 outlines an agent's behaviour in choosing prices in such a scenario.

We now present a general mixed integer linear programming model that will be executed each day to decide what customer requests to bid on and what prices to offer for these requests. This bid set will result in an uncertain number of accepted offers, affected by the actions of the customers and on the actions of any other producers that may be competing for the contracts. We assume no knowledge of how many other producers are bidding, and we assume no information on their cost structures, their objectives, their order books, their supply contracts or their production processes. We do, however, assume that we have a history of our own previous offers and their outcomes, and that from that history we could derive a simple probability distribution over the success of individual bids. We can use this distribution to determine the expected number of accepted orders for each product. Our model is thus based on the

expected orders. The constraints will ensure that we only consider bid sets which give expected orders that could be produced (based on our order book, our inventory state, our expected inventory replenishment, and our production ability), and will compute the profit on the expected orders. The objective function is then to select the bid set such this expected profit is maximised.

The intention is that any specific model will include integrated production and supply constraints. However, the form of the reasoning about prices is independent of the details of those constraints, and could be applied to many different production models. Therefore, for clarity, we concentrate here only on the pricing model; we do not include details of the production and supply constraints in this discussion, giving instead abstract statements. In section 4, we will show how to instantiate the model with specific production constraints for one specific problem.

Note that the result of the model will be an (approximated) expectation. If we receive fewer orders than expected, we will simply make less profit. If we receive more orders than expected, however, or a different distribution of orders, then we might not be able to produce all the products by the due dates specified in the original requests. We assume late delivery would result in penalties that again reduce the profit on that day's trading. The basic pricing model does not include any explicit provision for risk management; instead we assume the specific inventory policy and production constraints include buffers that are set each day before running the model. In our implementation, we do this using the newsboy model [18]. As we are interested in analysing pricing strategies, we construct a model that is only intended for making pricing decisions: procurement, inventory management and production schedules should be based on a different model. Finally, the pricing model is executed each day with a rolling horizon, and thus the results of over-bids or under-bids will be fed into the next day's model as an updated order book, will inform the probability distribution, and may trigger short-term procurement actions.

The details of the model are given in figure 2. The input parameters B_{ij} define a bill of materials for each product, and we assume they do not change over the time horizon. The parameters c_j represent the average unit cost of component j in inventory. The parameters D_{ir}, d_r and s_r are obtained from the set of requests made by customers; s_r is the maximum price the customer will pay for the product. The key to the model is the matrix of prices, P, and the vector of probabilities, p, such that P_{ik} is the price for product i that we believe will be accepted with probability p_k [1]. In section 3, we discuss

[1] An alternative model could have a function mapping prices to probabilities for each product. This would remove the need for the integer variable b_{rk}, but would introduce a non-linear term in the objective function. We choose to use a linear MIP rather than a non-linear program, and we demonstrate in later sections that the MIP can be used successfully in real-time decision making.

Problem parameters

B_{ij} : the number of components j needed to produce one unit of product i,
c_j : the cost of component j,
D_{ir} : the amount demanded for product i in request r,
d_r : the due date in request r,
s_r : the reserve price in request r,
p_k : the k'th element of the probability vector,
P_{ik} : the offer price for product i with acceptance probability p_k,
o_r : the price offset multiplier for request r,
S_{jt} : *the existing supply agreements: quantity of component j for delivery on day t*
O_{it} : *the order book: number and price of product i for delivery on day t*

Decision variables

b_{rk} : a {0,1} variable that takes the value of 1 if an offer is made for request r using the price corresponding to probability p_k

Auxiliary variables

θ : the expected total profit,
q_{it} : the expected quantity of product i for delivery on day t, $q_{it} \in \mathbb{R}^{0,+}$

maximize

$$\theta = \sum_{\forall r} \sum_{\forall k} \sum_{\forall i} D_{ir} \left(P_{ik} o_r - \sum_{\forall j} B_{ij} c_j \right) p_k b_{rk} \tag{1}$$

subject to

$$\sum_{\forall k} b_{rk} \leq 1 \qquad \forall r \tag{2}$$

$$P_{ik} o_r b_{rk} \leq s_r \qquad \forall r, \forall i, \forall k|_{D_{ir}>0} \tag{3}$$

$$\sum_{\forall r} \sum_{\forall k} D_{ir} p_k b_{rk} = q_{it} \qquad \forall t, \forall i|_{d_r=t} \tag{4}$$

supply, inventory and production constraints(S, O, B, q) (5)

Fig. 2. Generic optimisation model.

methods for acquiring and updating this matrix and vector. o_r is a multiplier which we may apply to the prices in P, to adjust for significant features in particular requests. Details of how we derive and use this parameter are given in sections 3 and 4. The remaining (abstract) input parameters are an order book O, which records the details of all previously accepted orders that have still to be delivered, and finally, the existing supply agreements S, which records what components will be arriving from suppliers over the coming horizon. The decision variables b_{rk} indicates whether or not an offer is made for request r at the price with acceptance probability p_k. θ is the total expected profit, and is the quantity to be maximised. The variables q_{it} represents the expected number of product i to be delivered on day t.

Equation (2) ensures only one offer can be made for each request, while (3) ensures any offer made meets the reserve price constraint. For each day t and product i, equation (4) computes q_{it}, the expected number of units of i to be delivered on that day, from the expected bid success on each request that had t as a due date. Abstract equation (5) models the details of our production process, supply contracts, and inventory management, constraining the value of q_{it} (and hence implicitly the bids) to ensure that the expected orders can be produced in addition to the existing orders specified by the order book O. These equations could include any buffers to mitigate against risk. The aim is to optimise the expected profit (1), where the profit on a product is calculated by subtracting the cost of components from the chosen selling price, and multiplying this by the probability that the bid will be accepted.

3 Pricing strategies for a real-time supply chain

In each time period of our dynamic supply chain environment, the agent must determine what prices to offer for the requests. Thus the agent must specify the input to our model, setting appropriate values for the parameters. In particular, the agent must specify the entries for the price matrix, P, from which the model will select the prices. We present four different strategies, making increasing use of information about the market.

Fixed Price: As a baseline to test against, we propose a simple initial strategy, **FixedPrice**, where we will use a fixed price for each product. The fixed price can be chosen based on past market prices as well as expected future prices and will remain constant over all time periods. The probability of acceptance of this price is set to 1, as is each request multiplier. Specifically, we restrict the basic model by setting $|p| = 1$, $p_1 = 1.0$, and $\forall r\ o_r = 1$.

Dynamic Price, Fixed Margin: The previous strategy used a fixed price and although it may function reasonably well, it is not capable of reacting to changing market prices. The most basic problem is that the price makes no allowance for changes to the cost of components. If costs increase, the agent's price may no longer be profitable. To deal with this we define a second strategy, **FixedMargin**, that sets the price of products to include a specific margin, m, over the cost price. The agent can then dynamically adapt the price to reflect changing costs by updating the price in each time period. Again, acceptance probability and request multipliers are set to 1 in the basic model: $|p| = 1$, $p_1 = 1.0$, $\forall i\ P_{i1} = m * \sum_{\forall j} B_{ij} c_j$ and $\forall r\ o_r = 1$

Dynamic Price, Online Learning: The previous strategy guarantees a certain margin on any sold products, but there are still a number of potential weaknesses to our approach:

- if the market is very competitive prices will drop and the agent may be undercut and priced out of the market;

- if the market is booming with high demand then the agent may be better off with higher prices in order to maximise its expected profit;
- the strategy assumes that all offers are accepted and if this is not the case, then both factory capacity and supplies are left unused.

In the market, we may be competing against other agents, each with individual strategies. The pricing strategy that we adopt should reflect the current market state and dynamically adapt to changes. As indicated previously, we consider a strategy which maintains a set of target probabilities, and maintains a set of prices for each product which we believe will be accepted by the customers with those probabilities. If we can successfully predict these prices, then we can increase the number of offers we make, and choose a set of offers that will maximise our expected profit. The only restriction we now place on the basic model is to set $o_r = 1 \; \forall r$.

To learn a price, w, that has a certain target probability, p_{target}, of being accepted, we keep track of the ratio of offers accepted, a, to those made, o, and periodically update the price based on this information using an online learning algorithm (algorithm 1 [2]). If the actual probability, a/o, is greater than the target then we increase the price, and if the actual is less than the target we reduce the price. The update size is set relative to the distance between the target and actual value meaning that the further away from the target the actual value is, then the larger the update. This value is multiplied by a step-size factor, α, which decreases over time by a factor of γ. This allows us to learn quickly initially by allowing larger jumps while progressively smaller jumps are allowed as time progresses until we reach a relatively stable price value. A lower limit can be placed on α to ensure it does not become too small. Prices that reach a level where they are no longer considered profitable, and so will have no offers made on them will gradually be increased ($p_{actual} = p_{target} + \epsilon$), resulting in them reaching a level where they can be tested once again.

Dynamic Price, Online and Historical Learning: For an autonomous agent, working in a real-time supply chain, lack of information about the market is a potential problem that may result in inaccurate price/probability combinations. Up to now, we have considered learning prices for different products but that ignores the price variations that would be caused by other factors such as different lead times or different quantities etc. It is difficult to learn online at such a detailed level because, as we increase the number of maintained prices, we begin to spread the offer/order information more

[2] This algorithm is based on the delta rule [19] (also known as Widrow-Hoff rule [20]), which has successfully been used in online reinforcement learning algorithms and neural networks [21]. The rule is derived by expressing the rate of change of prediction error with respect to the parameter w and then moving the parameter in the direction of decreasing error. This algorithm should be applicable under the assumption that prices are roughly correlated from day to day with gradual shifts rather than drastic changes.

Algorithm 1 *Updating prices*
if $a == 0$ **then**
  $p_{\text{actual}} = p_{\text{target}} + \epsilon$
else
  $p_{\text{actual}} = a/o$
$\delta = p_{\text{actual}} - p_{\text{target}}$
$w = w + (\alpha * \delta)$
$\alpha = (\alpha * \gamma)$

thinly across all the prices, which means less information to learn from for each individual price. To improve the prediction accuracy we complement the online learning with trends or patterns observed from historical data, **DynamicHistoric**. By analysing the impact that different parameters have on product prices, we can learn different *multipliers* that can be applied to the learned prices to tailor them for specific requests. For example, historical trends may tell us that orders for lower quantities may attract a premium on the price, as fewer competitors bid for the work; in which case we may apply an offset multiplier, o_r which raises the predicted price for low quantity requests and lowers the price for high quantity requests. Examples of such trends and multipliers will be given in section 4.2. This strategy now uses the full basic model.

4 Experimental setup: TAC–SCM

To evaluate our algorithms, we have developed an agent to compete in the international Trading Agent Competition Supply Chain Management (TAC–SCM) game [8]. TAC–SCM is based on a simulated supply chain in which agents must win contracts, obtain supplies, and produce goods, in competition with five other agents. Multiple customers issue requests for quotes for the sale of PCs, including a quantity, a due date, and a penalty cost. A number of production agents make bids for the contract, and for each request, a winning bid is selected. The winning agent must then manufacture the PCs, or provide them from stock, and ship them to the customer by the due date. In order to manufacture the PCs, the agent must procure raw material from suppliers. Again, requests for quotes are issued by the agent, the suppliers respond with offers, and an appropriate offer is selected. Each agent is limited by the capacity of its assembly line, and must select each day which set of products are to be manufactured. An agent has unlimited storage capacity, but must pay an inventory holding cost for each component and each finished PC. Failure to meet a due date on an order results in financial penalties.

The game is dynamic and it involves significant uncertainty. There are in

total 16 different PC configurations possible, each with a different set of components, and each requiring a different number of production cycles. The configurations are divided into three categories or ranges (low, medium and high) based on their expected price. Customer demand is uncertain - each day, the demand for configurations in each of the three categories varies based on a Poisson distribution around a target which is varied as a random walk. The due dates and penalties (and reserve prices) are selected uniformly at random over an interval. Daily reports on the maximum and minimum prices paid for each configuration are available. Every twenty days, the agent receives a market summary report, detailing the average prices. The game runs in real-time and simulates a full trading year, with each day lasting 15 seconds.

4.1 Agent setup

To implement our dynamic pricing strategies in the TAC–SCM environment, we first require basic handling rules for the supply and production side of the agent. Our customer pricing model requires as input the cost of components and the number of components available for use.

For procurement, we use a simple strategy of ordering components in advance and maintaining an inventory. The agent aims to get at least $1/(\#\ of\ agents)$ of the total market demand so its estimated daily component need is calculated to be the components needed to meet the maximum of this and the agent's own recent average daily market share. The replenishment quantities are based on this daily estimate, allowing orders to be tailored to demand. When making requests for components the agent sets a maximum price to be equal to the average market price of the components, which is obtained from the market reports. Although there is a small amount of uncertainty in the delivery dates of supplies, this strategy essentially provides the known inventory levels and component costs needed by the pricing model. However, although our bidding is based on those levels, there is obviously demand uncertainty, caused by the uncertain behaviour of the customers and the unknown behaviour of the other agents, and thus there is a risk that we will receive orders that exceed our inventory capacity. To cater for this risk, we maintain buffer stock, the amount of which is calculated daily according to the newsboy model [18], using the holding and penalty costs provided in the game. This buffer stock will be implemented through a reduced opening inventory given as input to the model. For the occasions where we still exceed the inventory levels, we issue short-term procurement orders to top-up the inventory at short notice.

Production in the game is based on a predefined number of cycles required for each product, with a daily limit in the factory. In order to ensure that the expected number of bids can be scheduled, the pricing model must build

a provisional schedule for the existing and expected orders. Again, to guard against the risk of excessive orders, we keep a buffer of production cycles, implemented by stating the number of usable cycles each day as an input parameter to the pricing model. Actual production is separate from the bidding strategy, and is done daily using a make–to–order strategy. Confirmed orders are ranked by due date, tie-breaking on which order has the largest penalty. Production is then scheduled for the orders if the components exist for them until all the day's available production capacity is used up. If we receive more orders than can be produced on time, this heuristic chooses to delay the ones with the smallest impact on profits.

The full optimisation model for choosing offers to make to customers is an extension of the general model described in section 2. For completeness, we provide the full model in figure 3. The new input parameters which replace the order book and supply agreement parameters from the general model are: l_i, the number of cycles required to produce a single unit of product i; C_t, the maximum production cycles available on day t; and A_{jt}, the number of components of type j scheduled to arrive in inventory on day t, where A_{j0} is the opening inventory level. There are four new auxiliary variables, describing the provisional schedule and inventory levels. f_{jt} is the number of units of component j to be used at time t. m_{it} is the number of units of product i to be manufactured at time t. I_{jt} is the closing inventory level for component j at period t, and Y_{it} is the closing inventory level for product i at period t. Four new constraints on the provisional production schedule and inventory levels replace the abstract constraint from the general model. Equation (10) ensures production does not exceed factory capacity. The expected component need is calculated in (11), and supply constraints are enforced by ensuring that the closing component inventory for any day is non-negative (12). Finally, (13) ensures that the closing product inventory level for each day is non-negative.

4.2 Implementing pricing strategies

We now describe the details of the four pricing strategies adapted for the TAC–SCM game.

- To implement **FixedPrice**, a fixed price for each product, set to 75% of the nominal price[3], is defined and used every day in the game. This value is based on observations that this price is generally both competitive and provides good margins.
- In **FixedMargin**, the input price for each product is dynamically updated each day, set to be the cost of the product plus an arbitrarily chosen margin of 10%.

[3] Nominal prices for each product are provided in the game description.

Problem parameters

B_{ij} : the number of components j needed to produce one unit of product i,
c_j : the cost of component j,
D_{ir} : the amount demanded for product i in request r,
d_r : the due date in request r,
s_r : the reserve price in request r,
p_k : the k'th element of the probability vector,
P_{ik} : the offer price for product i with acceptance probability p_k,
o_r : the price offset multiplier for request r,
l_i : the number of processing cycles required to build product i,
C_t : the factory capacity available at day t,
A_{jt} : the number of component j arriving at period t,

Decision variables

b_{rk} : a {0,1} variable that takes the value of 1 if an offer is made for request r using the price corresponding to probability p_k

Auxiliary variables

θ : the expected total profit,
q_{it} : the expected quantity of product i for delivery on day t, $q_{it} \in \mathbb{R}^{0,+}$
f_{jt} : the number of component j used at time t, $f_{jt} \in \mathbb{R}^{0,+}$,
m_{it} : the number of product i manufactured at time t, $m_{it} \in \mathbb{R}^{0,+}$,
I_{jt} : the closing inventory level for component j at period t, $I_{jt} \in \mathbb{R}^{0,+}$,
Y_{it} : the closing inventory level for product i at period t, $Y_{it} \in \mathbb{R}^{0,+}$

maximize

$$\theta = \sum_{\forall r} \sum_{\forall k} \sum_{\forall i} D_{ir} \left(P_{ik} o_r - \sum_{\forall j} B_{ij} c_j \right) p_k b_{rk} \tag{6}$$

subject to

$$\sum_{\forall k} b_{rk} \leq 1 \qquad \forall r \tag{7}$$

$$P_{ik} o_r b_{rk} \leq s_r \qquad \forall r, \forall i, \forall k|_{D_{ir}>0} \tag{8}$$

$$\sum_{\forall r} \sum_{\forall k} D_{ir} p_k b_{rk} = q_{it} \qquad \forall t, \forall i|_{d_r=t} \tag{9}$$

$$\sum_{\forall i} l_i m_{it} \leq C_t \qquad \forall t \tag{10}$$

$$\sum_{\forall i} B_{ij} m_{it} = f_{jt} \qquad \forall t, \forall j \tag{11}$$

$$I_{jt} = I_{jt-1} + A_{jt} - f_{jt} \qquad \forall t, \forall j \tag{12}$$

$$Y_{it} = Y_{it-1} + m_{it} - q_{it} \qquad \forall t, \forall i \tag{13}$$

Fig. 3. Optimisation model for TAC–SCM.

- Multiple prices and probabilities are introduced in **DynamicOnline**. The agent learns weights that will be multiplied to the nominal prices of each product. Weights are learned for each product range and for 3 target probabilities representing low, medium and high success rates, arbitrarily set to 0.3, 0.6 and 0.9 respectively. Attempts to learn more detailed weights or a larger number of probabilities proved unsuccessful due to the limited amount of available information, i.e. a limited number of customer requests to issue offers to. Each day in the game, the three probabilities along with the current learned prices for each product corresponding to these probabilities are input to the model, which then selects one for each offer made.
- To implement **DynamicHistoric**, an analysis was carried out on each game in the seeding round of the 2005 competition and the effect that individual customer RFQ parameters (product type, quantity, due date, reserve price and penalty) had on the sale price was examined. For each day of each game, for each RFQ parameter we calculate the average weights of winning prices for different values of that parameter. Then, we compare the average weights for different values to see how the values relate to each other (a relative comparison is used because the weights themselves will vary depending on the market conditions). E.g. for the reserve price, we divide the possible values into 5 categories: 75-85% of nominal price; 85-95%; 95-105%; 105-115%; 115-125%. We then calculate the average weights of winning bids for each of these categories. We use the weight of the first category as a base measure and then express all other weights relative to this, which resulted in the following *multipliers*: [1, 1.074, 1.121, 1.141, 1.147]. To use these multipliers, the agent learns a single weight per probability per product range as before. Now, let us assume that a weight of w has been learned that has a certain probability p of being accepted. If the agent wishes to bid a price that has probability p of being accepted taking into account the reserve price multiplier, then, it will bid the nominal price $\times w \times 1$ if the reserve price in the RFQ is less than 85% of the nominal price. It will bid nominal price $\times w \times 1.074$ if the reserve price is between 85-95% of the nominal price etc. I.e. we expect that the winning price of an RFQ with a reserve price in the 85-95% range will be 1.074 times greater than that of an identical RFQ that has a reserve of less than 85%.

  Patterns emerged for each RFQ parameter from our data analysis and were consistent across games with different market conditions. The full table of multipliers is given in Figure 4. The reserve price appears to be the parameter that most significantly affects the price of products. As the reserve price increased, so too did the average selling price. The lead time and quantity parameters also showed clear trends although with only minor impact on the weights. **DynamicHistoric** makes use of all of these multipliers, adjusting prices to tailor them for specific requests. The price used for bidding for an RFQ is multiplied by o_r which is a product of the multipliers that apply to that request.

Lead time	3	4	5	6	7	8	9	10	11	12
Multiplier	1	1	1	1	0.999	0.997	0.995	0.992	0.989	0.986

Quantity	1	2	3	4	5	6	7
Multiplier	1	1.005	1.008	1.011	1.012	1.013	1.015
Quantity	8	9	10	11	12	13	14
Multiplier	1.015	1.015	1.016	1.017	1.016	1.017	1.017
Quantity	15	16	17	18	19	20	
Multiplier	1.016	1.015	1.015	1.014	1.013	1.011	

Reserve price	75-85%	85-95%	95-105%	105-115%	115-125%
Multiplier	1	1.074	1.121	1.141	1.147

Penalty	5-8%	8-12%	12-15%
Multiplier	1	1	1.001

Product (low)	1	2	9	10	11	
Multiplier	1	1.015	1.008	1.023	1.027	
Product (mid)	3	4	5	12	13	14
Multiplier	1	1.004	0.981	1.009	0.980	0.992
Product (high)	6	7	8	15	16	
Multiplier	1	1.003	1.006	1.003	1.006	

Fig. 4. Multipliers calculated for each request parameter. Each table shows the multiplier, relative to the first column in the table. To tailor a price for a specific request, the nominal price of the product is multiplied by both the learned *weight* and the *multipliers* that apply to the request.

4.3 2005 TAC–SCM competition

An initial implementation of this agent, using the name ***Foreseer***, competed in the 2005 TAC–SCM competition. The version of the agent that took part was similar to **DynamicOnline**, but with a more naive method of setting buffer levels. Despite this and despite the simple algorithms for both procurement and production, ***Foreseer*** finished 14th out of 32 agents in the seeding round, and qualified for the quarter finals, but then missed qualification for

the semi-finals by one place. This performance is evidence that our procurement, buffering and production algorithms, although simple, provide a solid basis for our investigation of dynamic pricing, and that at least one of our dynamic pricing strategies is sufficient to provide creditable performance.

5 Experiments

Our hypothesis is that pricing strategies which make use of more information about the supply chain and the market will show, on average, increased profits. To test this, we run experiments in the TAC-SCM framework, as described above, competing against four independently designed agents downloaded from the TAC agent repository. Those agents are listed in table 1, also showing the positions in which they finished in the TAC-SCM 2005 seeding rounds and finals (although the downloaded agents may have been modified since the competition).

We performed a set of pair-wise comparisons of our different strategies competing against those four agents. We used this method for two main reasons:

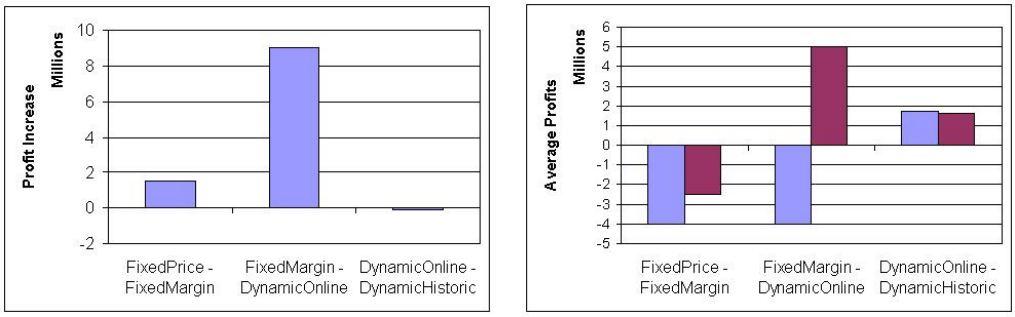
(1) it is undesirable to have too many of our own agents in any one game, since they share many features, and so is not comparable to competing against a set of unique agents, each with their own strategies;
(2) it is currently not possible to recreate exact game scenarios, and since game conditions can vary significantly this makes it impractical and unreliable to compare each of our strategies individual performance against the other agents.

We ran three sets of 25 games, comparing **FixedPrice** against **FixedMargin**, **FixedMargin** against **DynamicOnline**, and **DynamicOnline** against **DynamicHistoric**. We expected to see a steady increase in profits as we move from **FixedPrice** to **DynamicHistoric**.

Figure 5(a) shows how the average profit changes with each advance in the

Table 1
TAC–SCM agents that were used as competitors in experimental evaluation, all obtained from `http://www.sics.se/tac`.

Name	Seeding Round Position	Final Position	Reference
GoBlueOval	2	4th in quarter-final	
Mertacor	10	3rd in final	[22]
CrocodileAgent	12	4th in quarter-final	[23]
GeminiJK	19	5th in quarter-final	



(a) Profit difference **(b) Actual profits**

Fig. 5. Comparisons of profits from different models.

pricing model, while 5(b) shows the average profits obtained in each pair-wise comparison. As expected, the poorest model is the one with fixed prices - reasoning about the cost of production for individual orders (as we do in **FixedMargin**) gives a clear benefit. The most significant increase is the introduction of the learning mechanism over the fixed margin model, which is our first use of truly dynamic pricing. The ability to reason about the overall market conditions, and to use this reasoning to determine prices for each order produces a dramatic increase in profits. Note that this reasoning about the market involves no knowledge of how many customers there are, nor of the number of competitors or their bidding or production decisions. The third pair-wise comparison gave an initially surprising result - adding the extra information gained from analysis of the TAC–SCM 2005 seeding rounds over all 32 agents actually produced a slight decrease in profits (although at 0.5%, not a significant one). We discuss the reasons for this below.

Figure 6 shows the revenue, market share and factory utilisation data for the three comparisons. Each of these measures increases as we include more information in the pricing models, with again the most significant increase coming when we first introduce reasoning about the current market state. Note, however, that increased revenue and increased market share does not always correspond to increased profit: **DynamicHistoric** shows an improvement in these secondary measures over DynamicOnline, despite getting no improvement in profit. It has a 40% increase in market share, but this is at reduced margins.

Why does the incorporation of historic information not lead to increased average profits? Analysis of the probability estimates of both models (Figure 7) showed that **DynamicHistoric** has a 92% accuracy rate compared to **DynamicOnline**'s 79%. This improved accuracy leads to more accurate prediction of factory (Figure 8) and raw materials usage, and we expected this improvement to be reflected in the profits of the agents. However, Figure 7 reveals that both models are, on average, too cautious, and set prices that are too high to win the expected number of orders. In all scenarios **DynamicOn-**

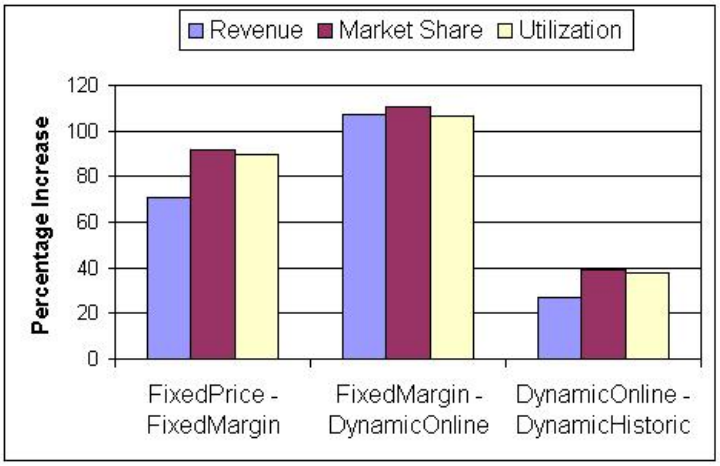


Fig. 6. Comparisons of pricing models showing the increase of each model compared to the previous one. E.g. **DynamicHistoric** has a market share which is a 40% increase on the market share of **DynamicOnline**.

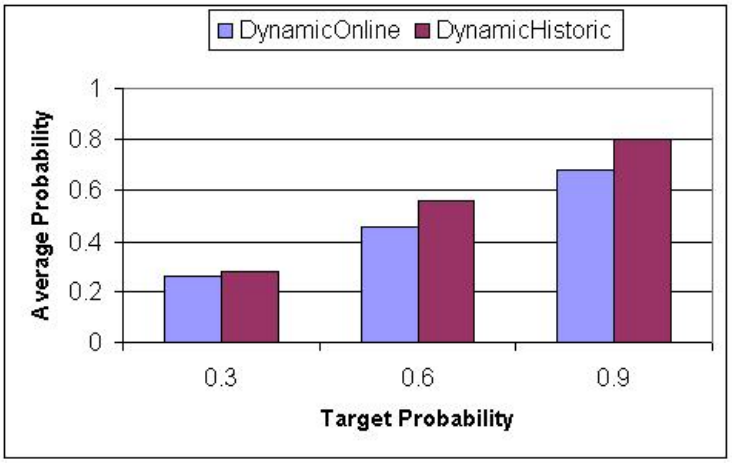
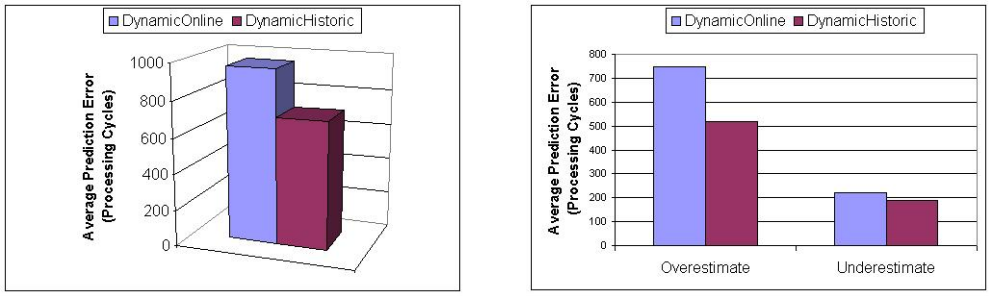


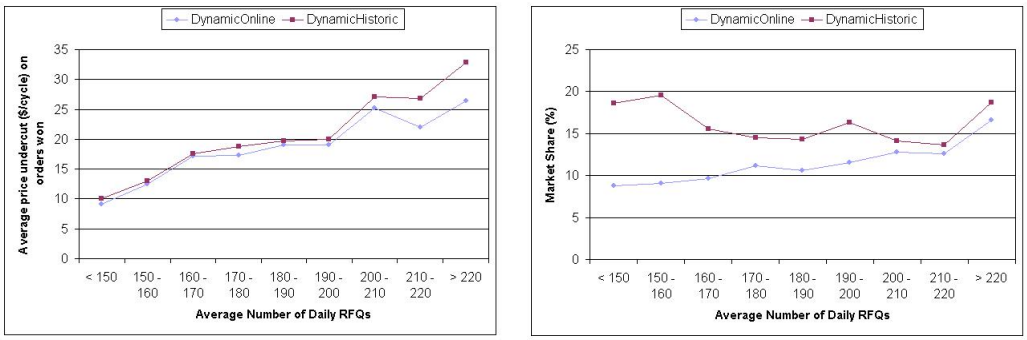
Fig. 7. Comparison of actual with target acceptance probabilities.

line chooses prices that on average undercut the competitors by less (and thus are higher prices) when an order is won (Figure 9 (a)), leading to higher profits on the accepted offers. However, Figure 9 (b) shows that **DynamicHistoric** maintains a higher market share in all market scenarios, while **DynamicOnline** has a particularly low market share when the demand is lower. In scenarios with high customer demand, **DynamicOnline** is able to win enough orders even with its high prices, and these high prices then translate to larger profits from fewer orders. In low demand scenarios, or in scenarios where the other agents are more competitive, then there are fewer possibilities for high prices to be successful and **DynamicHistoric** outperforms **DynamicOnline**. We confirm this in figure 10(a), which shows the average profits of the two agents and figure 10(b), which shows the frequency that one agent finished ahead of the other, both plotted against the level of demand. Of the 25 scenarios involving the two agents, **DynamicOnline** finished ahead in 13, and the av-



(a) Absolute Capacity Error **(b) Capacity Over/Underestimate**

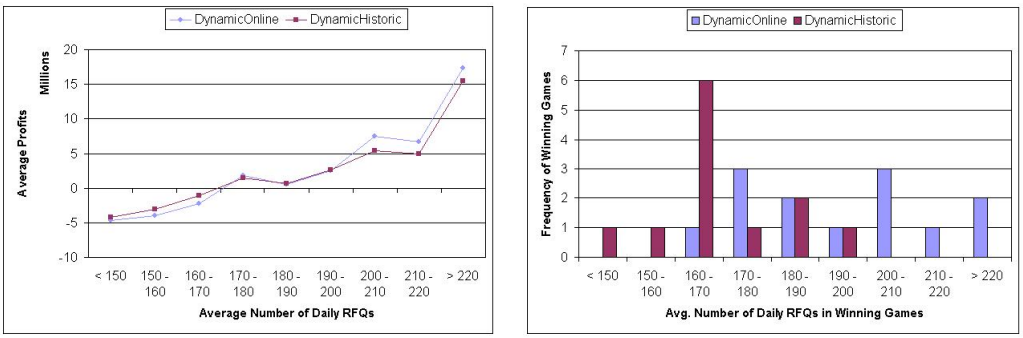
Fig. 8. Comparison of (a) average absolute capacity prediction error for different learning techniques and (b) over/underestimated capacity error. The absolute capacity error is the difference between the predicted and actual capacity needed to process orders. Overestimates occur when less orders are received than expected, and underestimates is the opposite.



(a) Price undercut **(b) Market share**

Fig. 9. **DynamicOnline** chooses prices that on average undercut the competitors by less, leading to higher profits on orders won. However, **DynamicHistoric** maintains a steady market share across all types of market conditions while **DynamicOnline** loses market share in low demand scenarios.

erage daily number of RFQs in those scenarios was 196. In the 12 scenarios that **DynamicHistoric** finished ahead, the average number of RFQs was 169. The overall average number of RFQs was 183: in scenarios with fewer RFQs than the average, **DynamicOnline** made approximately 134% greater loss than **DynamicHistoric**, while in scenarios with more RFQs than the average, **DynamicOnline** took approximately 27% greater profit. This confirms that incorporating historical information about the way market trends relate to profitable prices is important in competitive markets. But it also suggests that an agent should adapt its strategy in times of high demand, since higher profits can be made.



(a) Average profits **(b) Winning games**

Fig. 10. **DynamicHistoric** is more successful than **DynamicOnline** in low demand scenarios making more profit on average. The opposite is the case for high demand scenarios.

6 Conclusion and future work

In this paper, we propose a simple approach to the dynamic pricing problem in real-time supply chain management. The main feature is that we represent the market conditions as a set of bid prices which have a certain probability of being accepted, and that these price/probability pairs can be applied without any knowledge of the number of customers, or the number, identity, costs or strategy of competitors. In particular, we learn approximations to these prices using both online and historical information, and we update our approximation after each day of trading. Experimental results in a complex real-time stochastic supply chain simulation show that the incorporation of up-to-date market information in the form of our approximated probability distributions gives a significant increase in average profits tested against a portfolio of third-party agents. We have also shown that incorporating historical information on the relevance of particular parameters in the calls for tenders provides an increase in profits when the market is competitive and when the demand is relatively low.

There is plenty of scope for continuing this research into dynamic pricing. Further analysis on the impact of different customer demand levels on prices and further experimentation on algorithm parameter settings provide the potential for increased accuracy in the probability predictions for winning customer bids. Another important issue is to consider expected future demand when making today's decisions. Our current models attempt to optimise today's expected profit, even though it might be wiser to ignore some of the lower profit requests and reserve some inventory and factory capacity in anticipation of winning bids with higher profits tomorrow.

The supply chain trading agent as a whole offers even greater potential for research. To date we have developed an effective trading agent by focusing

primarily on the demand side of the supply chain. We have implemented only basic strategies for procurement and production. These areas are critical to the overall performance of the supply chain agent and efficient strategies need to be developed and integrated with the dynamic pricing mechanism.

Acknowledgements

We would like to acknowledge the creators of the TAC–SCM competition and software, who have provided a very useful test-bed for experiments, and an excellent environment for competitive comparison of implemented research. We would also like to thank the people whose agents we have used in our experiments, namely, the teams responsible for CrocodileAgent, GeminiJK, GoBlueOval and Mertacor. Finally, we would like to express our gratitude to Onur Koyuncu for his help implementing the model, to Chris Beck for his ideas on production scheduling, and to the anonymous referees for the suggestions on how to improve the paper. This work is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain Research (CTVR).

References

[1] W. Elmaghraby, P. Keskinocak, Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions, Management Science 49 (10) (2003) 1287–1309.

[2] D. Bertsimas, G. Perakis, Dynamic Pricing; A Learning Approach, Vol. 101 of Applied Optimization, Springer, 2006, pp. 45–80.

[3] K. Y. Lin, Dynamic pricing with real-time demand learning, European Journal of Operational Research 174 (1) (2006) 522–538.

[4] A. E. Lim, J. G. Shanthikumar, Relative entropy, exponential utility, and robust dynamic pricing, Operations Research. Forthcoming.

[5] E. Cope, Dynamic pricing of information goods under demand uncertainty, in: INFORMS Revenue Management and Pricing Section Conference, 2004, `http://web.mit.edu/orc/informs/Presentations/Day1_Revenue%20Management%20in%20Information%20Services/Cope.ppt`.

[6] J. O. Kephart, J. E. Hanson, A. R. Greenwald, Dynamic pricing by software agents, Comput. Networks 32 (6) (2000) 731–752.

[7] W. J. Hopp, X. Xu, Product line selection and pricing with modularity in design, Manufacturing and Service Operations Management 7 (3) (2005) 172–187.

[8] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, S. Janson, The supply chain management game for the 2005 trading agent competition, Tech. Rep. CMU-ISRI-04-139, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, `http://www.sics.se/tac` (2005).

[9] E. Rasmusen, Games and Information, Blackwell, 2001.

[10] K. K. Haugen, S. W. Wallace, Stochastic programming: Potential hazards when random variables reflect market interaction, Annals of Operations Research 142 (1) (2006) 119–127.

[11] D. Zhang, K. Zhao, C.-M. Liang, G. B. Huq, T.-H. Huang, Strategic trading agents via market modelling, SIGecom Exchange 4 (2004) 46–55.

[12] C. Kiekintveld, M. P. Wellman, S. P. Singh, J. Estelle, Y. Vorobeychik, V. Soni, M. R. Rudary, Distributed feedback control for decision making on supply chains., in: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS), 2004, pp. 384–392.

[13] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, M. Tschantz, Botticelli: A supply chain management agent, in: Proc. Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004, pp. 1174–1181.

[14] S. Bell, M. Benisch, M. Benthall, A. Greenwald, M. C. Tschantz, Multi-period online optimization in tac scm: The supplier offer acceptance problem, in: Proc. Workshop on Trading Agent Design and Analysis at the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004, pp. 21–27.

[15] W. Ketter, E. Kryzhnyaya, S. Damer, C. McMillen, A. Agovic, J. Collins, M. Gini, Analysis and design of supply-driven strategies in TAC-SCM, in: Proc. Workshop on Trading Agent Design and Analysis at the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004, pp. 44–51.

[16] D. Pardoe, P. Stone, Tactex-03: A supply chain management agent, SIGecom Exchange 4 (2004) 19–28.

[17] D. Pardoe, P. Stone, Bidding for customer orders in tac scm: A learning approach, in: Proc. Workshop on Trading Agent Design and Analysis at the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004, pp. 52–58.

[18] E. L. Porteus, Foundations of Stochastic Inventory Theory, Stanford University Press, 2002.

[19] S. Russell, P. Norvig, Artificial Intelligence, a Modern Approach, Prentice Hall, 1995.

[20] B. Widrow, M. Hoff, Adaptive switching circuits, in: IRE Western Electric Show and Convention Record, Part 4, 1960, pp. 96–104.

[21] T. Mitchell, Machine Learning, McGraw Hill, 1997.

[22] P. Toulis, D. Kehagias, P. A. Mitkas, Mertacor: a successful autonomous trading agent, in: Proc. Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2006, pp. 1191–1198, http://danae.ee.auth.gr/mertacorweb.

[23] A. Petric, V. Podobnik, G. Jezic, The crocodileagent: Analysis and comparison with other tac scm 2005 agents, in: Proc. Joint Workshop on Trading Agent Design and Analysis & Agent Mediated Electronic Commerce at the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2006, pp. 202–205, http://www.tel.fer.hr/index.php?option=com_content&task=view&id=1069&Itemid=869.