MinneTAC Sales Strategies for Supply Chain TAC

Article · July 2004

DOI: 10.1109/AAMAS.2004.243041 · Source: CiteSeer

Wolfgang Ketter
University of Cologne
170 PUBLICATIONS **2,041** CITATIONS
SEE PROFILE

Amrudin Agovic
University of Minnesota Twin Cities
17 PUBLICATIONS **126** CITATIONS
SEE PROFILE

John Collins
University of Minnesota Twin Cities
123 PUBLICATIONS **1,978** CITATIONS
SEE PROFILE

Maria Gini
University of Minnesota Twin Cities
342 PUBLICATIONS **4,755** CITATIONS
SEE PROFILE

MinneTAC Sales Strategies for Supply Chain TAC

Wolfgang Ketter, Elena Kryzhnyaya, Steven Damer, Colin McMillen,
Amrudin Agovic, John Collins, and Maria Gini*
Dept. of Computer Science and Engineering, University of Minnesota

Abstract

We describe two sales strategies used by our agent, MinneTAC, for the 2003 Supply Chain Management Trading Agent Competition (TAC SCM). Both strategies estimate, as the game progresses, the probability of receiving a customer order for different prices and compute for each the expected profit. Offers are made to maximize the expected profit. The main difference between the strategies is in the way the probability of receiving an order is updated, and in the way an offer price is calculated. The first strategy works well in high demand games, but not as well in low-demand games. The second was developed to improve performance in low-demand games. We empirically analyze the effect of the discount given by suppliers on orders received the first day of the game. We show that in high-demand games there is a strong correlation between the offers an agent receives from suppliers on the first day of the game and the agent's performance in the game.

1. Introduction

Competitive scenarios are increasingly being used as testbeds for the development of multiagent systems. A new game, called TAC SCM, was introduced for the 2003 Trading Agent Competition [1]. This game involves a supply chain management (SCM) scenario in which agents attempt to maximize profits by manufacturing personal computers and selling them to customers.

The TAC SCM competition is interesting for many reasons. Agents participating in a TAC game must base their decisions on limited information about the state of the market and the strategies of other agents. Agents must simultaneously compete in two separate but interrelated markets: the market from which the agents must buy their supplies and the market to which the agents must sell their finished products. Agents have a large number of decisions to make in a limited time, so the computational efficiency of the decision-making process is important.

We describe two sales strategies used by our agent, MinneTAC, in high-demand and low-demand games. We analyze a number of games and show how the start-effect caused by the large discount given by suppliers on orders made on the first day, coupled with the random order in which agent requests are considered, affects the outcome of the game.

2. Overview of TAC SCM

Six autonomous agents compete to maximize profits in a computer-assembly scenario. The simulation takes place over 220 virtual days, each lasting fifteen seconds of real time. Each agent has a bank account with an initial balance of zero. The agent with the highest bank balance at the end of the game wins. Agents earn money by selling computers they assemble out of parts purchased from suppliers.

To obtain parts, an agent must send a *request for quotes* (RFQ) to an appropriate *supplier*. Each RFQ specifies a component type, a quantity, and a due date. The next day, the agent will receive a response to each request. Suppliers respond by evaluating each RFQ to determine how many components they can deliver on the requested due date and how long it would take to produce all the components requested, considering the outstanding orders they have committed to and the RFQs they have already responded to this turn. If the supplier can produce the desired quantity on time, it responds with an offer that contains the price of the supplies. If not, the supplier responds with two offers: (1) an earliest complete offer with a revised due date and a price, and (2) a partial offer with a revised quantity and a price. The agent can accept either of these alternative offers, or reject both. Suppliers may deliver late, due to randomness in their production capacities. If the supplier has excess capacity, the price will be discounted; discounted prices may be as low as 50% of the base price.

Every day each agent receives a set of RFQs from potential *customers*. Each customer RFQ specifies the type of computer requested, along with quantity, due date, reserve price, and late penalty. Each agent may choose to bid on

* The Primary author is a student

some or all of the day's RFQs. Customers accept the lowest bid that is at or below the reserve price, and notify the agent the following day. The agent must ship customer orders on time, or pay the penalty for each day an order is late. If product is not shipped within five days of the due date the order is cancelled, the agent receives no payment, and no further penalties accrue.

3. A Priori Game Analysis

Prior to the competition, we analyzed the game to determine the bottlenecks, where a *bottleneck* on some day d is the factor which limits the production of PCs on day d.

We identified three types of bottlenecks: (1) a *demand bottleneck*, which is in effect if the demand for PCs is less than the agents' production capacities and the amount of available supplies, (2) a *production bottleneck*, which is in effect if the limiting factor is the agents' production capacities, and (3) a *supply bottleneck*, which is in effect when the limiting factor is the amount of available supplies.

The maximum potentially profitable production of PCs on a day can be computed as:

$$\begin{aligned} production_{PCs} = \min(&demand_{PCs} \times bid\_proportion_{PCs}, \\ &production\_capacity_{PCs}, \\ &supplies\_available_{PCs}) \end{aligned} \tag{1}$$

where

$$demand_{PCs} = \#RFQs \times \overline{RFQ\_quantity}^{1} \tag{2}$$

and $bid\_proportion_{PCs}$ is the proportion of $demand_{PCs}$ which actually receives bids. If the reserve price specified in the RFQ is higher than the price of the components used to make the PC, an agent can make a profit by bidding at or below the reserve price, assuming it has production capacity and components. Since customers' reserve prices are chosen randomly in the interval of 75% to 125% of the maximum price of the components, approximately half of the RFQs will specify reserve prices that are definitely higher than the cost of the components. Therefore, we can put a lower bound of 0.5 on $bid\_proportion_{PCs}$. However, components are often available at a discounted rate if the agents order them ahead of time, so in some games $bid\_proportion_{PCs}$ may be as high as 1.0.

We define $production\_capacity_{PCs}$ as the maximum total number of PCs that can be produced daily by agents' production lines:

$$production\_capacity_{PCs} = \frac{\#cycles \times \#agents}{cycles_{avg}} \tag{3}$$

where $\#cycles$ per day per agent is 2000, there are six agents, and the average number of cycles required to produce a PC is 5.5. The average number of cycles is computed assuming that each agent has sufficient supplies and that each of the 16 types of PCs is produced in equal quantities, as $cycles_{avg} = \sum_{i=1}^{16} cycles_i/16 = 88/16 = 5.5$. This results in $production\_capacity_{PCs} = 2181$.

We define $supplies\_available_{PCs}$ as the number of PCs that can be built from the supplies available in a day, assuming that suppliers are producing at maximal capacity. Every PC requires four components, one each from the categories of CPUs, motherboards, memory modules, and hard disks. Components in each category are produced with equal frequency. Thus

$$supplies\_available_{PCs} = supplies_{max}/4$$

where $supplies_{max}$ is the maximum number of supplies produced in a day. To calculate $supplies_{max}$, we need to determine the amount of supplies produced daily by each of the eight suppliers. A supplier i has a production capacity C_i that is determined on each day d by a mean reverting random walk with a lower bound:

$$\begin{aligned} C_i(d) = \max(0, C_i(&d-1) \\ &+random(-0.05, 0.05) \times C_{nominal} \\ &+0.01 \times C_{nominal} - C_i(d-1)) \end{aligned}$$

where $C_{nominal}$ is the nominal capacity, which is specified as 500 components per day. $C_i(0) = C_{nominal}$ is used to compute $C_i(1)$, the supplier's production capacity on the first day of the game.

We can now calculate $supplies\_available_{PCs}$ for a given day d:

$$supplies\_available_{PCs}(d) = \sum_{i=1}^{8} C_i(d)/4 \tag{4}$$

and combine the results of Equations 1, 2, 3, and 4 to predict the maximal number of PCs produced in a day d:

$$\begin{aligned} production_{PCs} = \min\Big(&\#RFQs \times \overline{RFQ\_quantity} \\ &\times bid\_proportion_{PCs}, \\ &\frac{\#cycles}{cycles_{avg}} \times \#agents, \\ &\textstyle\sum_{i=1}^{8} C_i(d)/4\Big) \end{aligned} \tag{5}$$

In a typical game, the initial average number of customer RFQs is 200 per day. For simplicity, we will assume that the number of RFQs per day is always exactly 200, and that suppliers' daily capacity is always the nominal capacity of 500 components per day. We also assume that $bid\_proportion_{PCs} = 0.5$; that is, we assume that agents are not able to obtain supplies at a discounted rate. We also assume $\overline{RFQ\_quantity} = 10.5$, since the average quantity specified in each RFQ is chosen randomly from a uni-

1 the notation $\overline{x}$ denotes the sample mean of the variable x.

form distribution over the interval $[1, 20]$. We can then substitute these values into Equation 5 to obtain:

$$\begin{aligned} production_{PCs} = \min(&200 \times 10.5 \times 0.5, \\ &2000 \times 6/5.5, \\ &\textstyle\sum_{i=1}^{8} 500/4) = \ 1000 \end{aligned}$$

This result shows that the most likely bottleneck for a typical game is supply availability. In fact, the availability of supplies is the most probable bottleneck as long as the number of RFQs per day is greater than 190. (If $\# RFQs = 190$, then $E[demand_{PCs}] \times bid\_proportion_{PCs} = 190 \times 10.5 \times 0.5 = 997.5 < 1000$.)

Since the initial average number of customer RFQs is chosen randomly from a uniform distribution over the range $[80, 320]$, approximately 45.8% of games will start off with a demand bottleneck. A greater number of games might enter a demand bottleneck after some period of time due to fluctuations in the average number of customer RFQs.

In the above calculations we assumed that $bid\_proportion_{PCs} = 0.5$. If agents get supplies at a discounted price, then we expect that $bid\_proportion_{PCs}$ will be greater than 0.5. The availability of supplies is then even more likely to be a bottleneck. Competition experience indicates that a substantial number of supplies are indeed obtained for less than the maximum price. The above calculations also suggest that agents' production capacities are not likely to be a bottleneck in any game in which there are at least three functioning agents. However, a single agent could be limited by its production capacity if it acquires substantially more supplies than its opponents.

4. MinneTAC Performance Analysis

The analysis of the bottlenecks led us to decide on a *supply-driven* strategy. Building to order would be a good strategy in the case of a demand bottleneck. However, since the most likely bottleneck is a supply bottleneck, a supply-driven strategy seems preferable.

Ideally, an agent's strategy should be flexible and adjust dynamically in each game without *a priori* assumptions. Because of the first day discount, an agent is better off making an uninformed *a priori* estimate of customer demand and ordering most of the components it anticipates using on the first day. The drawback to this approach is that if an agent acquires too many components, it may be forced to sell them at loss, since the components have no value at the end of the game.

4.1. MinneTAC sales strategies

We designed, implemented, and compared two variants of a supply-driven sales strategy, that we call *MaxEProfit* and *DemandDriven*.

Each day the strategy chosen determines an offer price for each RFQ. Offers are made only from the uncommitted finished goods inventory and are sorted by decreasing profit margin, where $ProfitMargin = (price - cost)/price$. For each offer made the agent reserves a fraction of the computers offered according to the probability of receiving an order, p_{order}, i.e.

$$reserved\_quantity = RFQ\_quantity \times p_{order}(RFQ)$$

This can, and sometimes does, lead to overcommitment and late delivery. It also requires that we estimate p_{order}; the two strategies use different methods to produce this estimate.

To compare these strategies we have created two variations of MinneTAC, that called Tabaluka and Eini. Tabaluka uses MaxEProfit, MinneTAC and Eini use DemandDriven.

Both MaxEProfit and DemandDriven control inventory by pulling stock; procurement and production work to maintain target inventory levels until late in the game. Both commit inventory to customer offers, but differ in the way they set prices, and in the way they estimate the probability of offer acceptance.

Because of the way customer demand is generated in the game server, most games can be classified as either high or low customer demand [2]. We will analyze our two strategies in a high-demand and a low-demand game (see Table 1). We will focus our comparison on product 4. Similar results can be shown for the other products. The nominal price, which is the sum of the base cost of the four individual components, for product 4 is \$1850.

In Figure 1, we show the aggregate demand curve for product 4 over the two games by price. The aggregate demand curve for a high-demand game is very different from the curve for a low-demand game There is demand for computers in the high-demand game at prices above and below the nominal price, but in low-demand games the bulk of the demand is even below the minimum reserve price, which equals 75% of the nominal price.

Table 2 compares the results of the two strategies over a series of high-demand [2] and low-demand games [3] all played on tac5.sics.se:8080. We calculated the minimum, average and maximum score of each strategy in those set of games. We can see that MaxEProfit is better able to take advantage of high-demand games, but generally performs worse in low-demand games.

4.1.1. MaxEProfit The MaxEProfit strategy determines an offer price that maximizes the expected profit margin

2 2385,2393,2396,2401,2407,2409,2410,2411,2412,2416, 2419,2420,2421,2594,2597,2598,2603,2607,2612,2613

3 2383,2387,2390,2392,2394,2399,2415,2417,2423,2425, 2426,2492,2593,2595,2599,2600,2604,2610,2614,2640

Game	*Agents and their Result (in $M)*						*Customer Demand in RFQs*		
Number	1	2	3	4	5	6	#RFQ	#RFQ/day	σ
2214	team2 -10.43	RedSox -18.3	MinneTAC -31.06	arnoch -34.87	RedAgent -38.08	Eini -39.83	21778	99.44	51.8
2218	Tabaluka 31.23	RedAgent 30.69	arnoch 23.24	MinneTAC 20.8	team2 8.86	Eini 7.89	65626	299.66	42.14

Table 1. Summary of the games examined. #RFQ is the total number of RFQs during the game, #RFQ/day is the mean number of RFQs/day, and σ is the standard deviation. Customer RFQs are issued over 219 days in a game. Eini and Tabaluka are disguised MinneTAC agents. Eini and MinneTAC use DemandDriven and Tabaluka uses MaxEProfit.

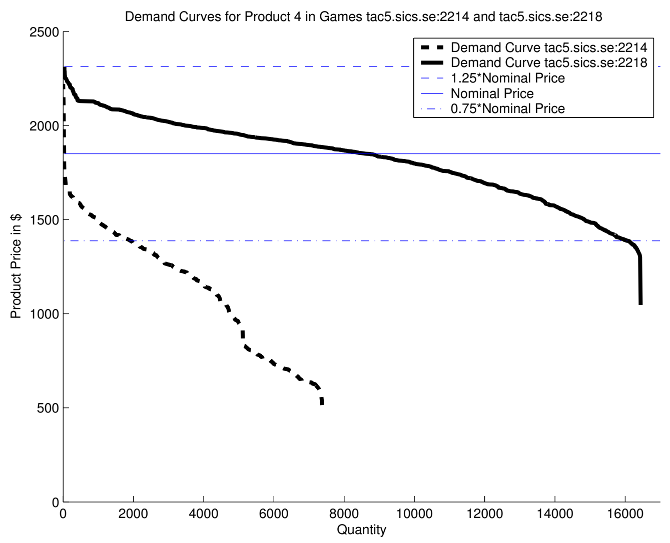

Figure 1. Games 2214 and 2218 – Aggregate demand curves for product 4 over the games.

from a potential customer order x:

$$E[\mathit{ProfitMargin}(x)] = \mathit{ProfitMargin}(x) \times p_{order}(x)$$

with the constraint that $price \geq targetAveragePrice$. *ProfitMargin* is calculated on the agent's moving average cost of the components. *targetAveragePrice* is an internal parameter that reflects the current market prices. The parameter is adjusted every 5-days based on the orders received, and every 20-days based on the market report, time left in the game, production rate, uncommitted finished goods inventory level, and customer demand. This parameter helps to ensure that the agent is not left with winning only unprofitable RFQs.

p_{order} is the estimated probability of receiving a customer order. This probability is influenced by the reserve price, quantity, lead-time, penalty, and product type specified in the RFQ, and also by the price specified in the offer. MaxEProfit models this probability as a 6-dimensional *order_prob* matrix with the following dimensions: offer_price, quantity, lead_time, reserve_price, penalty, and product_type. Each entry in *order_prob* contains the probability that a customer will accept an offer given the values of the parameters.

Since the values in *order_prob* are estimates, they are updated during the game whenever an offer is accepted or rejected. Initially the values are all set to 1, making the agent optimistic.

4.1.2. DemandDriven The DemandDriven strategy determines an offer price for a customer RFQ based on a target probability of receiving a customer order, *target_prob*, from a reverse cumulative density function (CDF) that models the oder probability. The objective is to make offers to sell out the inventory by the end of the game. *target_prob* (see Equation 6) is calculated every 5-days based on the currently observed market conditions (customer demand, and time left in the game) and on internal parameters (production rate and uncommitted finished goods inventory for a specific product).

$$target\_prob = \min(1, \frac{avail\_FG + \#\ built \times days\_left}{estimated\_demand}) \qquad (6)$$

where *avail_FG* is the number of finished goods available in the inventory for a particular product. $\#$ *built* is the number of units of the product built each day, *days_left* is the number of days left in the game, and *estimated_demand* is an internal parameter that forecasts future demand.

DemandDriven models the probability of customer order as a 5-dimensional *order_prob* matrix having the following dimensions: offer_price, customer_demand, lead_time, reserve_price, and product_type. The values of customer_demand are discretized into 3 ranges: low, medium, and high; lead_time is discretized into short and long.

Strategy	*High*			*Low*		
	Values (in $M)					
	Min	Avg	Max	Min	Avg	Max
MaxEProfit	-12.02	12.30	35.99	-66.90	-44.44	-7.36
DemandDriven	-23.65	8.70	30.89	-57.15	-34.49	30.89

Table 2. Performance comparison of the MaxEProfit (Tabaluka) and the DemandDriven (Eini) strategy in high-demand and low-demand games.

The *order_prob* matrix is pre-populated with values obtained from analysis of several past games. DemandDriven assumes that only a shift of the whole order probability curve could occur during the game, so, instead of updating the values of *order_prob* as done by MaxEProfit, every five days it shifts the values toward higher or lower prices depending on the difference between the acceptance rate of its offers and *target_prob*.

4.2. Performance in high-demand games

We show now how MaxEProfit, which is the strategy used by agent Tabaluka, performs in a high-demand game.

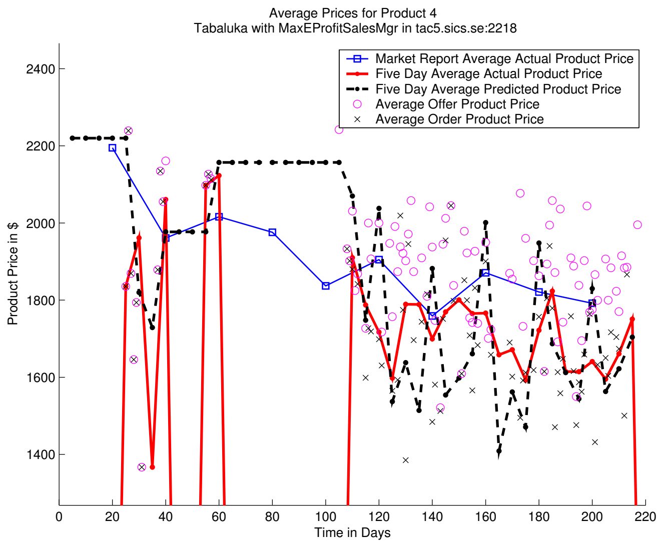

Figure 2. Game 2218 – Timeline for market report: Predicted vs actual values of Tabaluka (MaxEProfit) for product 4.

In Figure 2 we show the market reports every twenty days and compare them with Tabaluka's predictions. Since predictions are updated every 5 days, we show the agent predictions and the real prices over the same periods. In addition we show the offers made and the orders received by Tabaluka. A circle with a cross inside symbolizes an accepted order.

The inventory of finished goods was mostly empty until halfway through the game, as we can see in Figure 3. When the inventory is empty the sales strategy doesn't learn and makes no offers. This situation can be seen as a straight line in Figure 2 for the predicted product price. We observe that predicted product prices match well actual product prices, even though the prediction often over- and undershoots the real values.

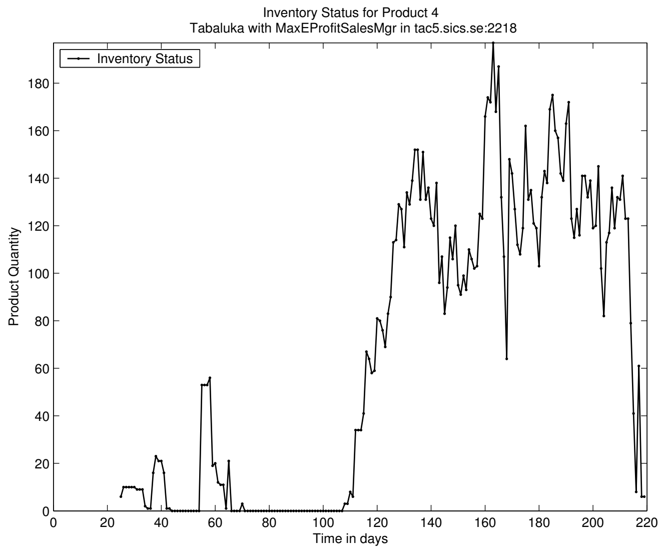

Figure 3. Game 2218 – Timeline inventory status of Tabakula for product 4.

Towards the end of the game this sales strategy tries to sell out the uncommitted finished goods inventory if the finished goods inventory level is higher than what the agent thinks it will be able to sell.

We can see the relationship between offer/order prices and reserve prices in Figure 4. We observe that in this case the reserve and order prices are close. This is not the case in every game.

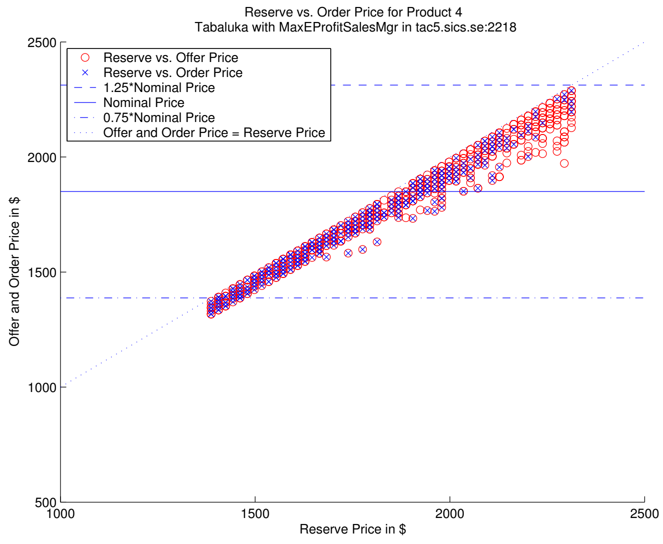

Figure 4. Game 2218 – Offer and order prices of Tabaluka vs reserve price for product 4.

We now analyze the performance of MinneTAC, which uses the DemandDriven strategy, in the same game we examined earlier. Figure 5 shows the same information as Figure 2, but for DemandDriven.

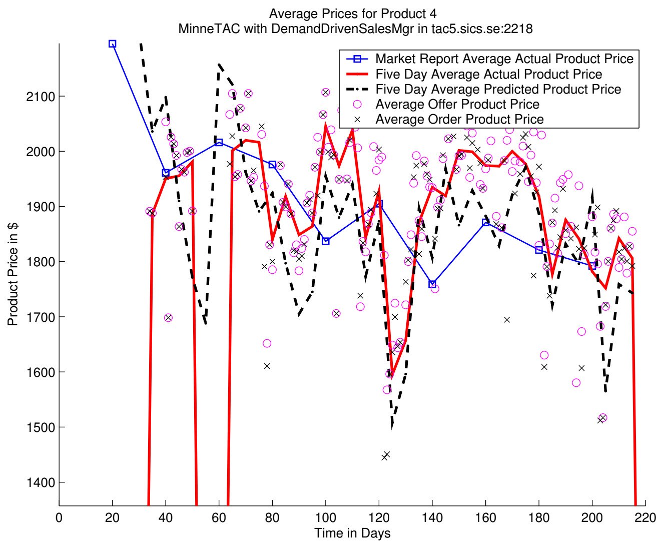

Figure 5. Game 2218 – Timeline for market report: Predicted vs actual values of MinneTAC (DemandDriven) for product 4.

We can see the relationship between offer/order prices and reserve prices of MinneTAC in Figure 6. We observe that in this case the reserve and order prices are not as close, as for agent Tabaluka in the same game (see Figure 4). The reason is that MinneTAC uses DemandDriven, which tries to clear the inventory by the end of the game. Therefore we observe quite a few offer and order prices far under the reserve price. DemandDriven is often too aggressive in selling goods because it assumes that a constant supply of raw material is being supplied until the end of the game. This is a drawback of DemandDriven that causes the agent to sell goods when it would be better off holding out for a higher price.

4.3. Performance in low-demand games

We will now analyze the performance of the DemandDriven strategy in low-demand games. MaxEProfit is not discussed here, since the analysis is similar.

The first day MinneTAC orders large amounts of components. The sales strategy starts to make offers only when there are finished goods in the inventory. Whenever finished goods are almost sold out, MinneTAC purchases new components to retain a certain level of raw inventory. This is problematic in very low-demand games, since the agent

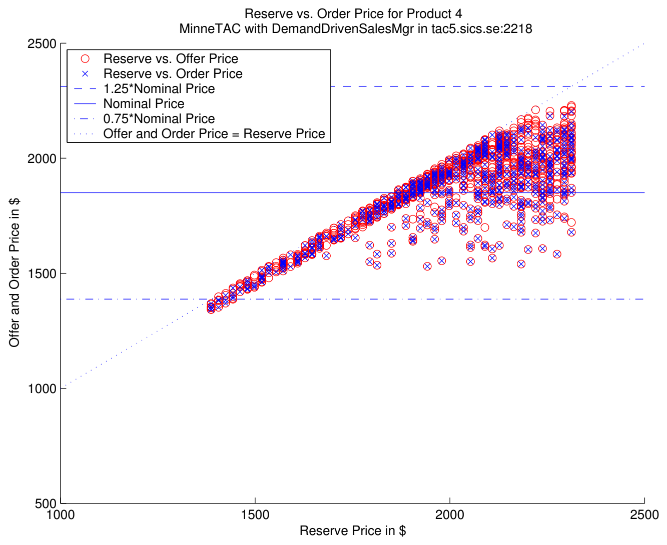

Figure 6. Game 2218 – Prices (offer and order) vs reserve price of MinneTAC for product 4.

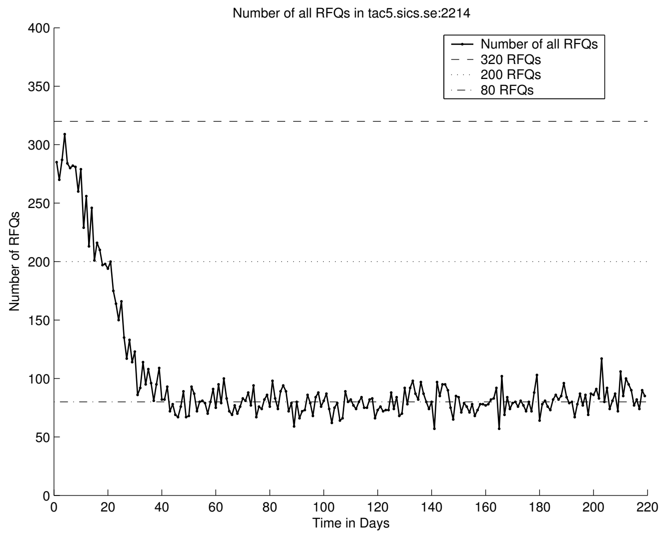

Figure 7. Game 2214 – Total number of RFQs. This is a low-demand game.

ends up with too many components and finished goods in the inventory. In Figure 7 we can observe that the game started off as a high-demand game, but after a short time the customer demand dropped to a very low level. A sudden mode change like this or a constant low or high-demand situation arises nearly in every game. In Figure 8 we can see MinneTAC's predictions versus the market reports, and the orders and offers made through the game. We can observe

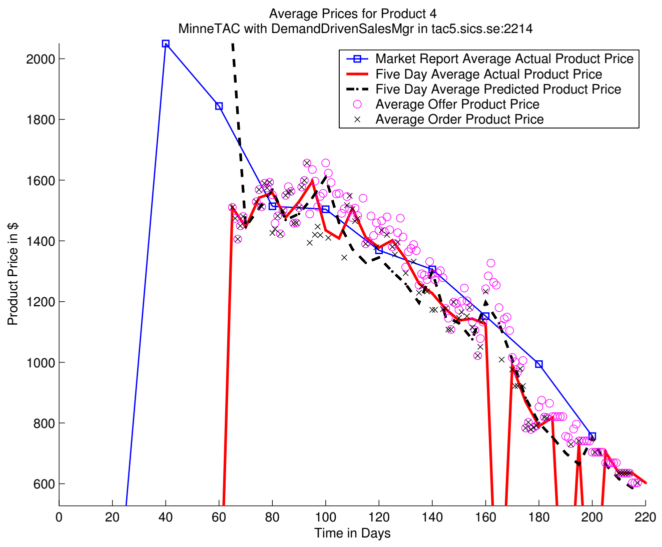

Figure 8. Game 2214 – Timeline for market report: Predicted vs actual values of MinneTAC for product 4.

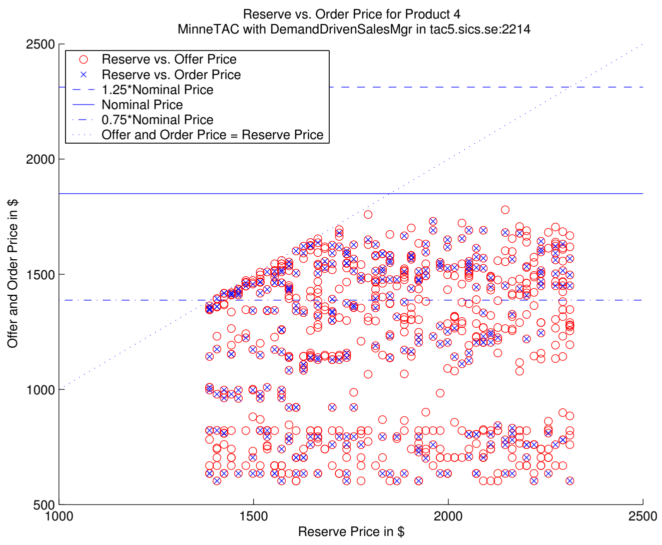

Figure 9. Game 2214 – Prices (offer and order) vs reserve price of MinneTAC for product 4.

how well the prices offered tracked the real market price.

Offer and order prices decrease with respect to the reserve prices as the game progresses (see Figure 8). In a low-demand situation like this, the competition to sell out products is very high and therefore the prices lower so much. In high-demand games the prices are usually between 75% and 125% of the nominal product price (see Figures 4 and 6), while the bulk of the orders in this low-demand game is far below 75% of the nominal product price (see Figure 9).

5. Analysis of Start-Effect

We will now analyze the start-effect in the game and show that the total volume an agent orders on the first day and the timeliness of the offers that it accepts have a strong impact on that agent's final score.

We developed a start-effect measure called *delay measure* (DM). DM (see Equation 7) is the delay, in days, in delivering the components weighted by the component total value. In this equation *BPrice* refers to the component's base price and *DDate* refers to the due date of the offer.

$$DM = \frac{\sum_{i=1}^{\#RFQs} Qty(RFQ_i) \times BPrice(RFQ_i) \times DDate(Offer_i)}{\sum_{i=1}^{\#RFQs} Qty(RFQ_i) \times BPrice(RFQ_i)} \quad (7)$$

Game 2218 was taken as an example for calculating the start-effect. Results are shown in Table 3. The measure works exceptionally well for high-demand games, but not as well for low-demand games. We can conclude that a low value of DM leads to a better final score. Tabaluka had the lowest DM and ended up first, Eini had the highest DM and came in last. A low delay measure is only an effective indicator when a high volume of goods is ordered on the first day. In this game team2's low DM is not a good indicator of its final result because of its low order volume on the first day.

In addition to looking at single games we looked at high-demand and low-demand games played at the 2003 International Conference for Electronic Commerce (Pittsburgh, October 2003). The games make a good test set because the configurations for each agent didn't change much and the agents themselves were robust. To determine if a game is high-demand or low-demand, we looked at $ratio = \frac{\sum \# ComputersOrdered}{ActivePlayers}$ and selected the 20 games with the highest [4] and lowest [5] ratios. Then we calculated the correlation coefficients between the bank status at the end of the game, the volume of first day orders, and the delay measure.

In the high-demand games we calculated a correlation coefficient of 0.5381 between the bank status and the total amount ordered on the first day, and a correlation coefficient of -.03456 between bank status and the delay measure. This shows that in high-demand games there is a strong relationship between the amount of parts an agent orders on the first day and their final score. There is also a strong relationship between the delay measure and the final score, per-

4 1641,1646,1650,1651,1652,1660,1661,1662,1663,1665, 1666,1667,1670,1673,1674,1682,1683,1685,1686,1690

5 1640,1642,1643,1644,1649,1653,1654,1657,1658,1659, 1668,1669,1671,1672,1676,1679,1680,1681,1687,1688

Agent	*Total Values (in $M) of*					*DM*
	RFQs	Timely Offers	Orders	Discount	Final Result	(days)
Tabaluka	142.93	57.90	118.48	59.24	31.23	73.57
RedAgent	132.03	0.00	14.80	57.40	30.69	91.94
arnoch	79.00	7.97	42.24	21.12	23.24	92.40
MinneTAC	142.93	28.31	95.46	47.73	20.80	102.39
team2	2.00	0.00	2.00	1.00	8.86	51.35
Eini	142.93	21.18	72.27	36.13	7.89	119.63

Table 3. Start-Effect for Game 2218: RFQ is the value of the components requested the first day, Timely Offers is the value of the components offered at the requested time, Orders is the value of the components ordered, and Discount is the discount (50%) obtained. DM is the delay measure in days.

haps because agents which received better offers were more likely to accept them. This indicates that the order in which suppliers process RFQs has a strong impact on the outcome of high-demand games.

In low-demand games the correlation coefficient between bank status and the total amount ordered on the first day was -0.3242 and the correlation coefficient between bank status and DM was 0.3904. This indicates that agents who did not order a lot of parts (perhaps because they received poor offers on the first day) did better in low-demand games, because they were less likely to be trapped with inventory they could not sell at a profit.

6. Related Work

Stone [4] discusses multi-agent competitions, including RoboCup and TAC. He gives an overview of the rules of these competitions, notes some similarities and differences between the two, and shares his views on the benefits and drawbacks of such competitions. Wellman [5] analyzed and developed metrics for price prediction algorithms in the TAC Classic game similar to what we have done for TAC SCM. Predicting prices is an important part of the decision process of agents. Our strategies have been inspired by the work of Kephart et al. [3], who explored several dynamic pricing algorithms for information goods, where shopbots look for the best price, and pricebots try to adapt their posted prices to attract the most business.

7. Conclusions and Future Work

There are a number of factors which can limit agent performance in the TAC SCM game. The limit to an agents profitability is usually either customer demand or supplier capacity. Due to the bimodal nature of the customer demand algorithm this limit is usually severe.

MinneTAC orders parts based on the assumption of a supplier capacity bottleneck (high-demand game). We have explored two different sales strategies to compensate in the case of a customer demand bottleneck (low-demand game). Both MinneTAC's sales strategies were competitive in high-demand games, but often sold larger quantities at lower margins compared to other agents. Even though the Demand-Driven strategy performs better than the MaxEProfit strategy in low-demand games it is still unable to compensate for the large number of parts ordered on the first day.

Agents in TAC SCM must decide how many parts to order from suppliers on the first day, and this decision has a strong influence on their performance in the rest of the game. We have developed a measure to describe the quality of supplier responses to an agents RFQs. A low value indicates timely delivery dates, and correlates with good performance in high-demand games. However, in low-demand games a low value correlates with bad performance. This indicates that the order of supplier response to agent RFQs is very important on the first day of the game.

8. Acknowledgements

We would like to express our gratitude to the many teams who participated and to the people who organised 2003 TAC SCM, especially R. Arunachalam, J. Eriksson, N. Finne, S. Janson, and N. Sadeh. Partial support for this research is gratefully acknowledged from the National Science Foundation under award NSF/IIS-0084202.

References

[1] R. Arunachalam, J. Eriksson, N. Finne, S. Janson, and N. Sadeh. The TAC supply chain management game. Technical Report Draft Version 0.62, Swedish Institute of Computer Science, Sweden, 2003.

[2] J. Estelle, Y. Vorobeychik, M. Wellman, S. Singh, C. Kiekintveld, and V. Soni. Strategic interactions in a supply chain game. Technical report, University of Michigan, USA, 2003.

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

[4] P. Stone, R. E. Schnapire, M. L. L. J. A. Csirik, and D. McAllester. ATTac-2001: A learning, autonomous bidding agent. Submitted to the Eigtheenth National Conference on Artificial Intelligence (AAAI-2002), January 2002.

[5] M. P. Wellman, D. M. Reeves, K. M. Lochner, and Y. Vorobeychik. Price prediction in a trading agent competition. *Journal of Artificial Intelligence Research*, 2003.