

Analysis and Design of Supply-Driven Strategies in TAC SCM

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 the MinneTAC agent for the 2003 Supply Chain Management Trading Agent Competition. Both strategies estimate, as the game progresses, the probability of receiving a customer order for different offer prices. Offers are made to maximize the expected profit margin on each order. The main difference between the strategies is in how they compute the probability of receiving an order and the offer prices. The first strategy works well in high-demand games, the second was designed to improve performance in low-demand games. We analyze empirically the effect of the discount given by suppliers on orders received the first day of the game, and we show that in high-demand games there is a correlation between the offers an agent receives from suppliers 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 [8]. 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 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 they buy their supplies and the market to which they 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, and analyze their performance in different games. We show how the start-effect caused by the large discount given by suppliers on orders made 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 components 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 some or all of the day's RFQs. Customers accept the lowest bid that is at or below their reserve price, and notify the agent the following day. The agent must ship customer orders on time, or pay a penalty for each day an order is late. If a 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 its potential bottlenecks. A *bottleneck* on 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 happens if the demand for PCs is less than the agents' production capacities and the amount of available supplies, (2) a *production bottleneck*, which happens if the limiting factor is the agents' production capacities, and (3) a *supply bottleneck*, which happens when the limiting factor is the amount of available supplies.

To detect what bottlenecks might arise in a game, we start by estimating the maximum potentially profitable production of PCs on a day (which we call *UsableProduction*):

$$UsableProduction = \min(Demand \times BidFraction, ProductionCapacity, SuppliesAvailable) \quad (1)$$

where

$$Demand = \# RFQs \times \overline{Quantity_RFQ}^1 \quad (2)$$

BidFraction is the proportion of *Demand* 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 higher than the cost of the components. Therefore, we can put a lower bound of 0.5 on *BidFraction*. Since components are often available at a discounted rate if ordered ahead of time, in some games *BidFraction* may be as high as 1.0.

ProductionCapacity is the maximum number of PCs that can be produced daily by the agents:

$$ProductionCapacity = \frac{\# Cycles \times \# Agents}{AvgCycles} = \frac{2000 \times 6}{5.5} = 2181 \quad (3)$$

where $\# Cycles$ per day is 2000, and $\# Agents$ is 6. *AvgCycles* is the average number of cycles to build a PC. We compute it assuming that each agent has sufficient supplies and that each of the 16 types of PCs is produced in equal quantities, as $AvgCycles = \sum_{i=1}^{16} Cycles_i / 16 = 88 / 16 = 5.5$. This results in *ProductionCapacity* = 2181. Since the number of cycles needed to build a PC is between 4 and 7, the value of *ProductionCapacity* is in the range [1714, 3000].

SuppliesAvailable is the number of PCs that can be built from the supplies available in a day, assuming that suppliers are producing at maximal capacity and components in each category are produced with equal frequency. Every PC requires four components. The supplier capacity changes daily by a mean reverting random walk. For simplicity, we assume the daily capacity is always equal to *NominalCapacity*, the nominal capacity, which is specified as 500 components per day:

$$SuppliesAvailable = NominalCapacity / 4 = 500 / 4 \quad (4)$$

In a typical game, the initial average number of customer RFQs per day is 200. We assume it is always 200, and we assume *BidFraction* = 0.5; that is, we assume that agents are not able to obtain supplies at a discounted rate. We also assume $\overline{Quantity_RFQ} = 10.5$, since the average quantity specified in each RFQ is chosen randomly from a uniform distribution over the interval [1, 20]. By substituting these values into Equation 1 we obtain:

$$\begin{aligned} UsableProduction &= \min(200 \times 10.5 \times 0.5, \\ &\quad \frac{2000}{5.5} \times 6, \\ &\quad \sum_{i=1}^8 500/4) \\ &= \min(1050, 2181, 1000) \\ &= 1000. \end{aligned}$$

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 $E[Demand] \times BidFraction = 190 \times 10.5 \times 0.5 = 997.5 < 1000$.) In low demand games the number of RFQs per day is typically less than 190, so the bottleneck could be a demand bottleneck. However, if supplies are obtained at a discount, *BidFraction* can be as high as 1, so to have a demand bottleneck the number of RFQs per day has to be less than 95 (If $\# RFQs = 95$, then $E[Demand] \times BidFraction = 95 \times 10.5 \times 1 = 997.5 < 1000$.)

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. In the above calculations we assumed that *BidFraction* = 0.5. If agents get supplies at a discounted price, then we expect that *BidFraction* 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 were 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.

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

4. MinneTAC Sales Strategies

The analysis of the bottlenecks led us to decide to use a *supply-driven* strategy. We designed, implemented, and compared two variants of a supply-driven sales strategy, that we call *MaxEProfit* and *DemandDriven* [6].

Each day the agent 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 $ProfitMargin = (price - cost)/price$. The strategies differ in the way they set prices and they estimate the probability of receiving an order. For each RFQ on which an offer is made, the agent reserves from its inventory a fraction of the computers offered according to the estimated probability of receiving an order, $P(order)$, i.e.

$$QuantityReserved = RFQQuantity \times P(order)$$

Both strategies commit existing inventory to offers, but differ in the way they set prices, and in the way they estimate the probability of offer acceptance.

MaxEProfit determines an offer price that maximizes the expected profit margin from an order:

$$E[ProfitMargin] = ProfitMargin \times P(order)$$

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. MaxEProfit models this probability as a 6-dimensional *OrderProb* matrix with the following dimensions: offer price, quantity, lead time, reserve price, penalty, and product type. Each entry in *OrderProb* contains the probability that a customer will accept an offer given the values of the parameters. The values are updated during the game whenever an offer is accepted or rejected. Initially the values are all set to 1, making the agent overly optimistic, and are adjusted during the game as follows:

$$OrderProb = (1 - \alpha) \times OrderProb + \alpha \times Success$$

where α is the learning rate, and *Success* is either 1 (offer accepted) or 0 (offer not accepted). The value for α we used in the experiments is 0.2.

MaxEProfit attempts to learn a discrete regression model. Since $P(order)$ is estimated from many parameters, often there are not enough data until very well into

the game for the estimates to be accurate. This observation led us to design the DemandDriven strategy.

DemandDriven determines the offer price for a customer RFQ based on a target probability of receiving an order, $TargetProb$, which is computed from the reverse Cumulative Density Function (CDF) that models the order probability. The objective is to make offers to sell out the inventory by the end of the game. $TargetProb$ is updated every 5-days based on 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).

$$TargetProb = \min\left(1, \frac{AvailablePCs + DailyProduction \times DaysLeft}{EstimatedDemand}\right)$$

where $AvailablePCs$ is the number of finished PCs available in the inventory for a particular product. $DailyProduction$ is the number of units of the product built each day, $DaysLeft$ is the number of days left in the game, and $EstimatedDemand$ is an internal parameter that forecasts future demand.

DemandDriven models the probability of a customer order as a 5-dimensional *OrderProb* 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. The matrix is pre-populated with values obtained from analysis of several past games. DemandDriven assumes that only a shift of the order probability curve could occur during the game, so, instead of updating the values of *OrderProb* as done by MaxEProfit, every five days it shifts the values toward higher or lower prices by a fixed percent depending on the difference between the success rate of its offers and $TargetProb$. The fixed percent change we use (which, in the experiments, was 0.5%) causes, at times, the offer prices to over react to the market changes. Overall the prices track well the market.

5. Performance Analysis

Most games can be classified as either high or low customer demand [3]. Table 2 compares the results of the two strategies over a series of high-demand² and low-demand games³ all played on tac5.sics.se:8080. We can see that MaxEProfit performs better in high-demand games, but is generally worse in low-demand games.

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

Table 1. Summary of the games examined in this paper. #RFQ is the total number of RFQs during the game, $\#RFQ/day$ is the average number of RFQs per day, and σ is the standard deviation. Customer RFQs are issued over 219 days in a game. Eini and Tabaluka are variants of MinneTAC. Eini and MinneTAC use DemandDriven, Tabaluka uses MaxEProfit.

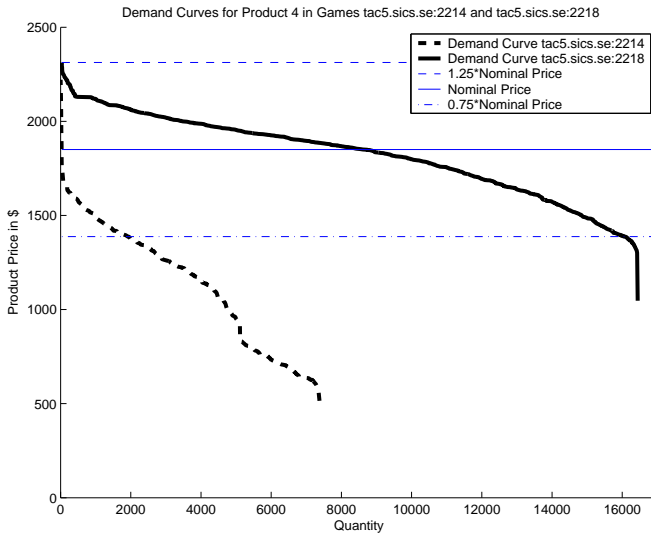


Figure 1. Games 2214 and 2218 – Aggregate demand curves for product 4 over the games.

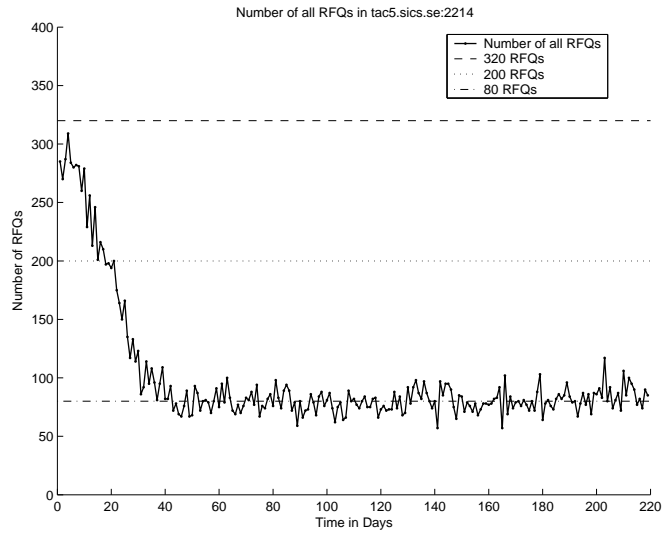


Figure 2. Game 2214 – Total number of RFQs. This is a low-demand game.

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

Table 2. Performance comparison of MaxE-Profit and DemandDriven strategies in high and low-demand games.

We compare our two strategies in a high and a low-demand game (see Table 1), focusing our comparison on product 4, whose nominal price (which is the sum of the base cost of the four individual components) is \$1850. Similar results can be shown for the other products.

In Figure 1, we show the aggregate demand curve for product 4 over the two games by price. The aggregate de-

mand curve for a high-demand game is very different from the curve for a low-demand game. In the high-demand game there is demand for computers at prices above the nominal price, but in low-demand games, as the game in Figure 2, the bulk of the demand is below the minimum reserve price, which equals 75% of the nominal price.

5.1. Performance in high-demand games

We show now how MaxEProfit performs in a high-demand game. In Figure 3 we show the market reports every twenty days and compare them with the agent’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 average offers (circles) made and the average orders (crosses) received.

In this game, the inventory of finished goods was mostly empty until halfway through the game. When the inventory is empty the sales strategy doesn’t make any offer and consequently does not learn. This situation can be seen as

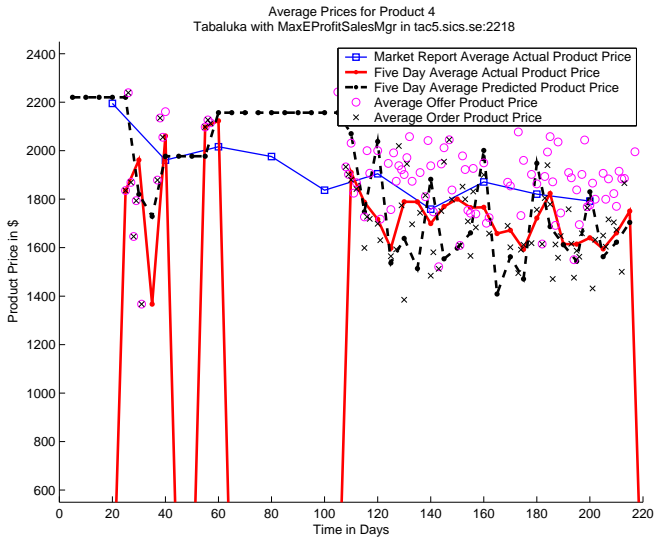


Figure 3. Game 2218 – Timeline for market report: Predicted vs actual values of MaxE-Profit for product 4.

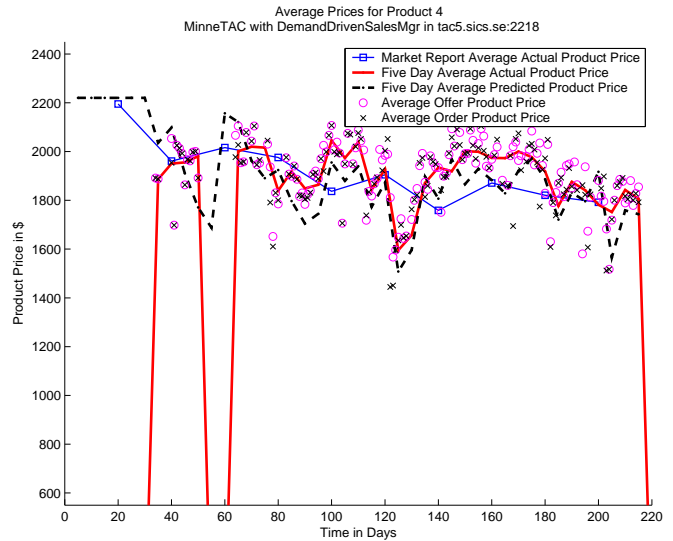


Figure 5. Game 2218 – Timeline for market report: Predicted vs actual values of Demand-Driven for product 4.

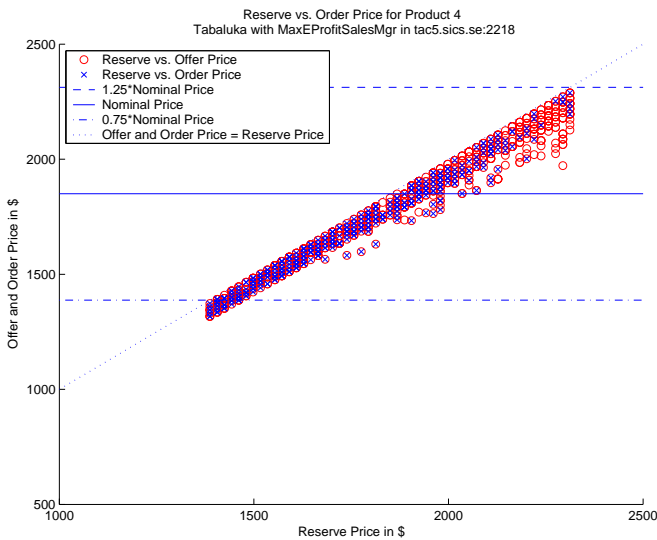


Figure 4. Game 2218 – Offer and order prices of MaxEProfit vs reserve price for product 4.

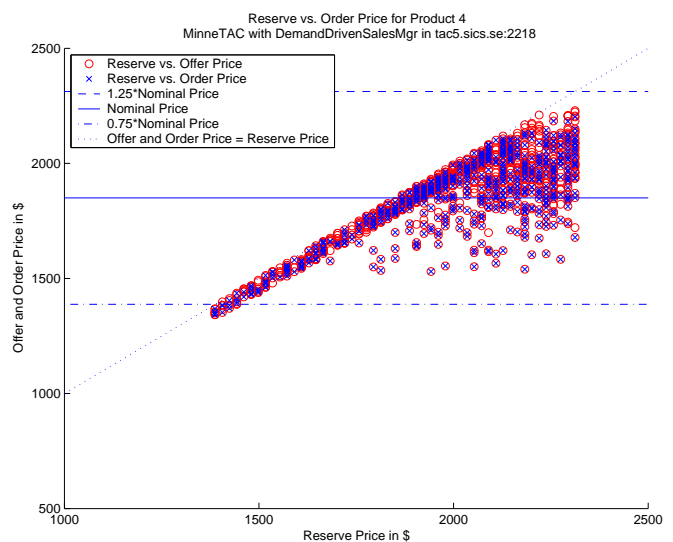


Figure 6. Game 2218 – Offer and order prices of DemandDriven vs reserve price for product 4.

a straight line in Figure 3 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.

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 be-

tween offer/order prices and reserve prices in Figure 4. A circle with a cross inside symbolizes an accepted order. We observe that in this case the reserve and order prices are close. This is not the case in every game.

We now analyze the performance of MinneTAC, which uses the DemandDriven strategy, in the same game. Figure 5 shows the same information as Figure 3, but for Demand-

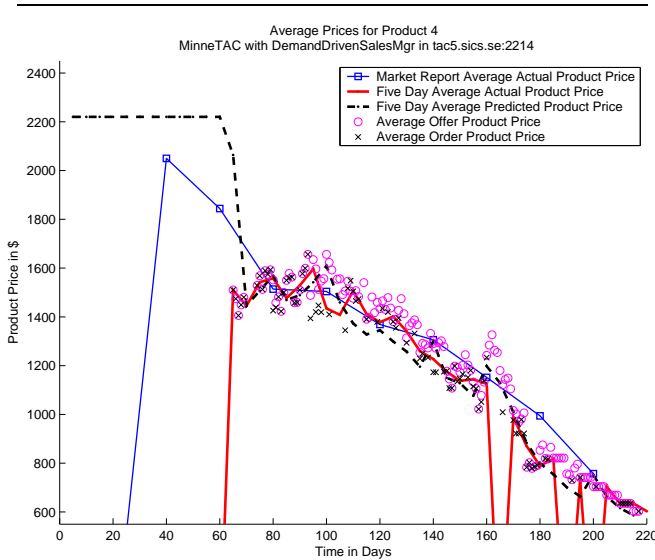


Figure 7. Game 2214 – Timeline for market report: Predicted vs actual values of Demand-Driven for product 4.

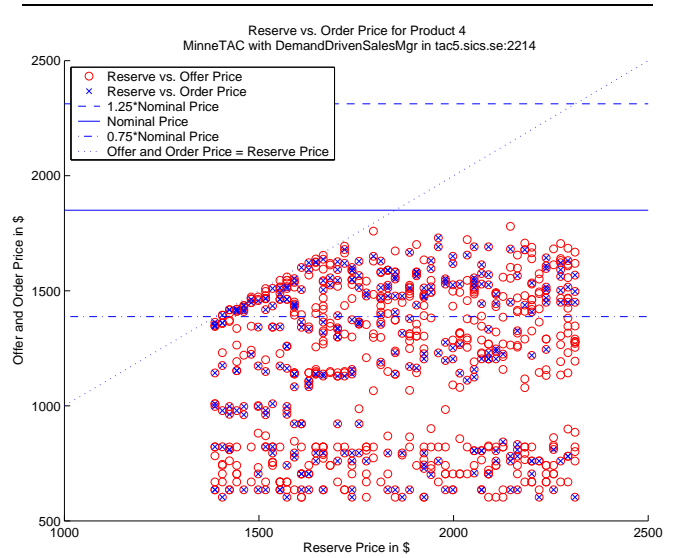


Figure 8. Game 2214 – Prices (offer and order) vs reserve price of DemandDriven for product 4.

Driven. 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 when using MaxEProfit in the same game (see Figure 4). The reason is that DemandDriven tries to clear the inventory by the end of the game, and it is willing to make offers far below 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.

5.2. Performance in low-demand games

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

The first day the agent 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 ends up with too many components and finished goods in the inventory.

In Figure 7 we can see the agent's predictions versus the market reports, and the orders and offers made through the game. We can observe 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 7). In a low-demand situation like this, the competi-

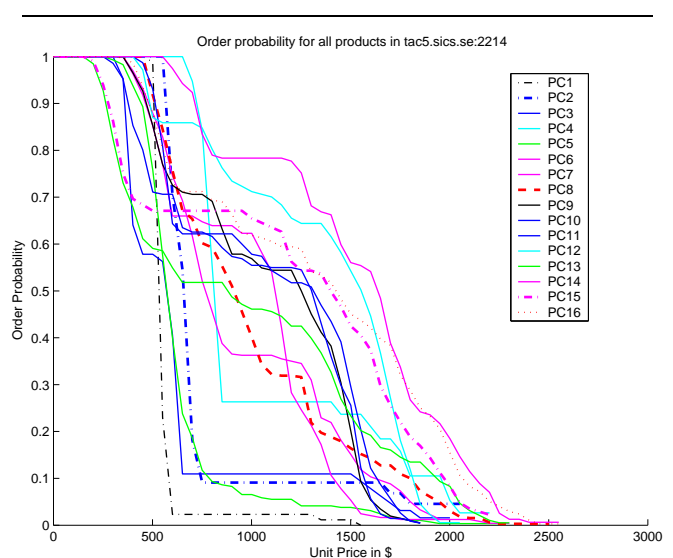


Figure 9. Game 2214 – Probability of order for different product types requested in a RFQ.

tion 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 8). In Figure 9 we can see how low the probability of order is for some products in a low demand game.

Agent	Total Values (in \$M) of					DM (days)
	Value RFQs	Timely Offers	Orders	Discount	Final Result	
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: Value RFQs 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. Eini and MinneTAC use DemandDriven and Tabaluka uses MaxEProfit.

6. 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 6) is the delay, in days, in delivering the components weighted by the component total value.

$$DM = \frac{\sum_{i=1}^{\#RFQs} Value(RFQ_i) \times DueDate(RFQ_i)}{\sum_{i=1}^{\#RFQs} Value(RFQ_i)}$$

where $Value(RFQ_i)$ is computed multiplying the components base price by the quantity in the RFQ, and $DueDate(RFQ_i)$ is the due date of the offer.

Results for game 2218 are shown in Table 3. The measure works very 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 DM is an effective indicator only 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 many 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 and the agents 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 DM.

In high-demand games we calculated a correlation coefficient of 0.5381 between bank status and the total amount ordered on the first day, and a correlation coefficient of -.03456 between bank status and DM. 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 its final score. There is also a strong relationship between DM and the final score, perhaps 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.

7. Related Work

Predicting prices is an important part of the decision process of agents. Our strategies have been inspired by the work of Kephart et al. [5], who explored several dynamic pricing algorithms for information goods, where shopbots look for the best price, and pricebots try to adapt their prices to attract business. Wellman [9] analyzed and developed metrics for price prediction algorithms in the TAC Classic game, similar to what we have done for TAC SCM.

Nearly all agents in last year’s competition used some way of modeling the probability of receiving an order. Botticelli [1] uses a linear CDF to determine the relationship between offer price and order probability. We use a reverse CDF and take other factors into account, such as quantity and due date.

TacTex [7] uses the lowest and highest offer price, which are provided for each product every day by the game server, and determines the probability of an order by linear interpolation. Their estimates depend only on the type of computer requested and the reserve price, whereas we use more parameters (6 for the MaxEProfit strategy and 5 for the DemandDriven strategy). RedAgent [4], the winner of last year

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

TAC SCM, uses an internal marketplace structure with competing bidders to set offer prices. PackaTAC [2] lets other agents set the price and tries to follow. The Jackaroo team [10] applied a game theoretic approach to set offer prices, using a variation of the Cournot game for modeling the product market.

Since we estimated the bottleneck was going to be in the supply and not in the production, we did not worry, as other teams [1, 7], about optimizing the production of our agent.

8. Conclusions and Future Work

We analyzed the factors which can limit the performance of an agent in TAC SCM. The limit to profitability is usually either caused by limited supplier capacity (in high-demand games) or, more rarely, by limited customer demand (in low-demand games).

MinneTAC orders parts aggressively, assuming a supplier capacity bottleneck. This works well in high-demand games, but in low-demand games the agent ends up buying too many components and so it ends up having to sell computers at a loss.

We have examined two different sales strategies, MaxEProfit and DemandDriven, that take into account both a supply bottleneck and a customer demand bottleneck. Both sales strategies were competitive in high-demand games, but often sold computers at lower margins compared to other agents. Even though the DemandDriven strategy performs better than the MaxEProfit strategy in low-demand games often it is still unable to compensate for the large number of parts ordered on the first day.

We have shown that the random order in which agent's RFQs are selected by suppliers and the huge discount on components ordered the first day affect the outcome of the game so much that for high-demand games the winner of the game can be predicted from the first day replies of suppliers. We have developed a measure of the delay in obtaining supplies, and shown that the a low value of delay correlates with good performance in high-demand games. However, in low-demand games a low delay correlates with bad performance.

The TAC SCM rule-set have undergone significant changes for the 2004 competition, which will affect some parts of our analysis. Customer demand will be evened out so that there will no longer be a clear distinction between high-demand and low-demand games. Changes to supplier pricing rules will reduce the start-effect by making it more difficult to acquire large quantities of cheap supplies on the first day.

There are a number of areas where the strategy of our agent can be improved, and we are in the process of exploring some of them. We are replacing the discrete regression model used by MaxEProfit with a continuous regres-

sion model, and we are developing more flexible procurement and production managers.

The design of the MinneTAC agent [6], in which each basic behavior is implemented in a separate, configurable component with minimal and well defined dependencies on other components, facilitates experimenting with different strategies. This design allows each user to focus on a single problem and work independently, and it allows multiple user to tackle the same problem in different ways.

Acknowledgments

We would like to express our gratitude to the teams who participated and the people who organized 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] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. Tschantz. Botticelli: A supply chain management agent designed to optimize under uncertainty. *ACM Trans. on Computational Logic*, 4(3):29–37, 2004.
- [2] E. Dahlgren and P. Wurman. PackaTAC: A conservative trading agent. *SIGecom Exchanges*, 4(3):33–40, 2004.
- [3] 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.
- [4] P. W. Keller, F.-O. Duguay, and D. Precup. Redagent - winner of the TAC SCM 2003. *SIGecom Exchanges*, 4(3):1–8, 2004.
- [5] J. O. Kephart, J. E. Hanson, and A. R. Greenwald. Dynamic pricing by software agents. *Computer Networks*, 32(6):731–752, 2000.
- [6] W. Ketter, E. Kryzhnyaya, S. Damer, C. McMillen, A. Agovic, J. Collins, and M. Gini. Design and analysis of the MinneTAC-03 Supply-Chain Trading Agent. Technical Report 04-016, University of Minnesota, Dept of Computer Science and Engineering, Minneapolis, MN, 2004.
- [7] D. Pardoe and P. Stone. TacTex-03: A supply chain management agent. *SIGecom Exchanges*, 4(3):19–28, 2004.
- [8] N. Sadeh, R. Arunachalam, J. Eriksson, N. Finne, and S. Janson. TAC-03: A supply-chain trading competition. *AI Magazine*, 24(1):92–94, 2003.
- [9] 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.
- [10] D. Zhang, K. Zhao, C.-M. Liang, G. B. Huq, and T.-H. Huang. Strategic trading agents via market modeling. *SIGecom Exchanges*, 4(3):46–55, 2004.