Analyzing the Procurement Process in a Simplified Version of the TAC SCM Game

Hosna Jabbari

December 9, 2005

Abstract

The TAC supply chain management game presents a simulated market environment with automated trading agents each responsible for making decisions for the whole manufacturing process from procuring raw materials to shipping the completed orders to customers. Agents in this game have to deal with challenging and complicated problems with little or no direct information about the state of the game. Procurement process is one of these challenging tasks and is the first process that agents have to deal with. Efficient procurement process can lead to significant profit in the game. In this paper, we focus on the procurement process and specifically consider the procurement process of the University of Michigan's agent, Deep Maize. Then we introduce and analyze two strategies to exploit this agent. Finally we conclude our paper by showing some future considerations for agent's procurement designs.

1 Introduction

Supply chain management is the process of planning and coordinating the activities of organizations across the supply chain. This process deals with all activities from procurement of raw materials to the delivery of the finished goods. In today's global and complicated economy, effective and efficient supply chain management plays a crucial role to the competitiveness of manufacturing goods. Supply chain management directly influences the ability of manufacturers to meet changing market demands in a timely and cost effective manner. Thus, the impact of the performance improvements in supply chain management is tremendous.

To reduce the complexity of the dynamic market, people have begun thinking recently about automating the human factor in decision making in the supply chain management process. Supply Chain Management (SCM) competition is a new branch of Trading Agent Competition (TAC) which was designed to capture some of the challenges of this process and help researchers gain a first hand experience in this process.

TAC SCM defines a very challenging and complex six-player game in which players (agents) have to deal with severely incomplete information about the

market changes and other agent's strategy spaces. In this game, agents play the roles of computer manufacturers who must handle three processes: buying the raw material from the suppliers, assembling the personal computers and selling the complete computers to customers. During these processes, agents have very limited time to make their decisions based on the incomplete information they get from the market. The problem studied in this paper is motivated by the work of M.P. Wellman, C. Kiekintveld, et al. at the Artificial Intelligence Laboratory, University of Michigan on their work on their SCM agent, Deep Maize, the second entry from the first TAC SCM competition [1].

The rest of the paper is organized as follows. In Section 2, we overview the TAC SCM game. In Section 3, we describe the main design principles of Deep Maize with an emphasis on its procurement process. We then introduce and analyze two strategies to exploit the agent in Section 4 and conclude the paper in Section 5.

2 The TAC SCM Game

In a TAC SCM game [2] six personal computer (PC) assembly agents compete for customer orders and procurement of a variety of components in a simulated market managed by a game server. The length of the game is 220 simulated days, each taking 15 seconds in real time. During each day, customers issue request for quotes (RFQ) for a specific PC and select from the quotes submitted by the agents on the previous day, based on delivery dates and prices. Agents have till the end of that day to respond to those RFQs. They have to procure components from a set of eight suppliers, produce a set of PC based on the capacity of their assembly line and ship the completed PC's to customers. Each day suppliers receive different RFQs from agents for different PC component and they have to respond to the RFQs they received the previous day.

2.1 Procurement Process

In this paper, we focus on the procurement process of the game. In this process agents have to deal with two challenging questions:

1. What RFQs to issue to suppliers and
2. considering the received RFQs from suppliers which to accept.

There are eight suppliers in the game, simulated by game server. Each of them produce two of the component types required for a PC. Each PC have four different components: CPU, motherboard, memory and hard disk. Four CPU's are all produced by one supplier; the two types of components in other categories are each produced by two suppliers. Each supplier has a nominal capacity of about 500 each day but the actual capacity of their production line differs from this nominal capacity by a random walk. Each agent can send up to ten RFQs each day in priority order to suppliers, each specifying a quantity,

due date and a reserve price. The suppliers respond the next day with offers specifying the quantity, due date and price. Suppliers reserve enough capacity to meet these requirements till the next day when the agents accept or reject the offers. If supplier's production capacity is not sufficient to fulfill the RFQ completely, it relaxes one of the constraints in the RFQ and responds with a partial offer (reduced quantity) or an earliest day (change of due date). To generate responses, suppliers execute the following till no more RFQs remain on their list:

1. sort the received RFQs based on the agent's reputation. Reputation is the ratio of agent's accepted quantity to their ordered quantity.
2. pick an agent from the highest reputation set
3. take the highest-priority RFQ remaining on the agent's list.
4. generate an offer if possible.

Each component has a based price. Suppliers set the price of each component for each day based on their available capacity and demand for that component. Their pricing formula as comes in [2] is:

$$p_c(d+i) = p_c^{base}(1 - 0.5(\frac{C_{d,i}^{avl'}}{iC_d^{ac}})) \quad (1)$$

where $p_c(d+i)$ is the component price on day d due on day $d+i$, p_c^{base} is the base price for the component and $C_{d,i}^{avl'}$ is the available capacity of the supplier ignoring the capacity limits and C_d^{ac} is the actual capacity of supplier on day d. Because procurement process is in a tight relationship with other challenging processes agents have to deal with, we think it is worthwhile talking about other processes very briefly in the following subsections.

2.2 Bidding Algorithms

During the bidding process, agents have to decide

1. based on RFQs received from customers, which to bid on, and
2. with what price they should bid on the customer RFQs

This process is a complicated reverse auction with very limited information and lots of uncertainty about the market situation and other agent's strategy spaces. This process has attracted lots of attention in previous TAC SCM games. For examples of the strategies in dealing with these challenges see [3, 4, 5, 6].

2.3 Production and Shipping Process

During the production and shipping process, agents must answer the following questions:

1. How to allocate available component inventory and capacity to the production of PCs, and

2. Which assembled PCs to ship to which customers.

Each customer RFQ will specify the specific PC type, the due date, maximum price that the customer is willing to pay and a penalty term for each day of delay. If the customer order is not ready by five days after the due date order is cancelled automatically. So agents have to prioritize the shipments of PCs based on different factors.

3 Deep Maize

Deep Maize is University of Michigan's TAC SCM agent entry in years 2003 to 2005. With a clear object oriented design as a distributed feedback control system [1] using some probabilistic models to project future demands [7], this agent had been amongst the finalist agents in the previous years of TAC SCM games. These facts and availability of published research results on this agent have been the main motivations for us to analyze this specific agent.

Deep Maize uses Bayesian network model to project the underlying customer demand and trend variable in each of the three market segments [8, 9]. It acts as a distributed feedback system in that it maintains a reference zone of profitable operation and tries to minimize its distance from this profitable zone during the game [1]. It uses the periodic market reports and its own information from the game to estimate the supplier capacity for each product line. This agent shows optimized performance in steady state.

3.1 Procurement process

To reduce the difference between its current state and profitable reference zone, Deep Maize uses reference inventory trajectory [7], which is the sum of three sources of component requirements:

1. outstanding customer orders

2. expected future component utilization, and

3. baseline buffer level.

Deep Maize considers each of these cases in turn and creates RFQs to be sent to suppliers each day to procure components according to these priorities. When calculating its current inventory, Deep Maize also takes into account its future component arrivals. If considering its available inventory, some customer RFQs still cannot be filled, it generates an RFQ for corresponding deficit quantity and due date. When it commits components from its inventory to customer orders, Deep Maize modifies its inventory by removing these committed components. Deep Maize also keeps components for approximately 6 days of its production

in its inventory to be used for future delays. If still some RFQs slots are left uncommitted, it uses them to request components addressing the long-term expected component utilization. To order the components in a more profitable way, it biases its selection towards the days when components are expected to be cheaper. For this purpose it uses its future demand projection model. During each day, it also sends some probe RFQs only to estimate suppliers capacities better. When it comes to the acceptance stage in its procurement procedure, Deep Maize makes its acceptance decisions separately for each component type as it gives different valuations (reserve price) to different component types corresponding to different states of optimization. For example components required to fill existing customer orders are valued higher than the components that address future expected consumption.

4 Strategy Description and Analysis

TAC SCM as comes in [2] is a very complicated and challenging game with little information, which makes it hard to analyze. For the purpose of this paper we simplify this game to a two-player game. We use all other rules of the original game in this paper unless said otherwise explicitly. In the game we consider in this paper, we give two strategies to exploit Deep Maize.

4.1 Fake Inflation Strategy

As said before in TAC SCM game, price of the components are set daily based on customer demand and agent's reputation. So the RFQs submitted by the agents with reputation y will influence the price for agents with reputations $rep \leq y$. If our agent in the game against Deep Maize always accepts a reasonable fraction of its orders (See [2]), it will never loses its perfect reputation and can always increase the component's price for Deep Maize. If our agent does not need to procure specific components for its production at any days in the game, it can use the fake inflation strategy against Deep Maize. To do so, we generate RFQs for all components (which we don't need) of all suppliers setting the quantities to an amount more than suppliers' nominal capacities for each day and setting due date to the $d + 3$ (because if due date is less than three days in future, our submitted RFQs are not considered by suppliers). Now because we don't need those components, we set the reserve price for the RFQs to a very little number (not equal to 0). This way we have influenced the market for Deep Maize in that it cannot estimate the future demand correctly and we have increased the components' price if it wants to procure components. Now if suppliers respond to our RFQs with an offer, we accept their offers and use our very cheap components for our future orders.

4.2 Hunt the Hunter Strategy

In this strategy, we put a trap for Deep Maize to exploit itself by bidding on too many customer RFQs. Basically, on the first day of the game, when Deep Maize has an empty baseline buffer level and no information about the game's demand, we don't participate in bidding on customer RFQs. Because Deep Maize was not designed to look for such a scenario, it will bid on all the customer RFQs, which are at least 25 RFQs in high and low range markets and at least 30 RFQs in mid range market. According to the game specifications [2] these RFQs have due dates between 3 to 12 days in future. With a simple calculation we find that even if Deep Maize had all the required components in its inventory, it would still take 9 days to produce the PCs. So if we block one of the PC components over the next 9 days, Deep Maize will be forced to pay huge penalties for its overdue orders and most of its orders would be cancelled. The cheapest component to block Deep Maize on is motherboard. So our strategy is to send RFQs with more than nominal capacity of suppliers for motherboards due on 3 days in future. To make sure that we have blocked Deep Maize on motherboard completely, we send multiple RFQs to motherboard suppliers everyday to make sure that we got all their available capacities. We will use a normal strategy to procure other PC components and after day one we also participate in bidding on customer orders. After the $9th$ day we change the strategy to a normal one and we only implement our fake inflation strategy so exploit Deep Maize's future demand projection. The loss that Deep Maize goes through during these 9 days will give us a big winning factor for the rest of the game.

5 Conclusion and Future work

In this paper, we investigated procurement strategy of the University of Michigan's TAC SCM agent, Deep Maize. We also showed two different strategies to exploit that agent. In the fake inflation strategy, we exploit Deep Maize's ability to project future demand and to estimate future prices and at the same time we increase the components' price by sending large RFQs to suppliers. We also found that the design of Deep Maize does not deal with the cases when the agent bids on too many customer orders and we used it in our "Hunt the Hunter" strategy to exploit Deep Maize by not participating in the auction for customer orders the first day of the game and blocking Deep Maize on the cheapest PC component. This flaw in Deep Maize's design comes from the fact that in its probabilistic model, the probability of such situation was estimated as zero. But in real life and even in TAC SCM such situations may happen by collision among the opponents. For example if in the TAC SCM game the finalist agents collude against one specific agent by implementing our "hunt the hunter" strategy, they will buy themselves higher probability of winning in the game.

The results discussed in this paper were only considered theoretically and were not tested in practice. So one of the main tasks for our future work is to

implement an agent with these strategies and study all these cases in practice , in a real game environment. Another main task for our future work would be to find a good strategy to detect these traps and respond to them efficiently.

References

[1] C. Kiekintveld, M. P. Wellman, S. Singh, J. Estelle, Y. Vorobeychik, V. Soni and M. Rudary, "Distributed Feedback Control for Decision Making on Supply Chains," *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, pp. 384–392, 2004.

[2] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne and S. Janson, "The Supply Chain Management Game for the 2006 Trading Agent Competition," Technical Report.*CMU-ISRI-05-132, School of Computer Science, Carnegie Mellon University*, Nov. 2005.

[3] D. Pardoe and P. Stone, "Bidding for Customer Orders in TAC SCM," *AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems*, Jul. 2004.

[4] 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 Transactions on Computational Logic*, vol. 4, No. 3, pp. 29–37, 2004.

[5] P. W. Keller, F. Duguay and D. Precup, "RedAgent-2003: An Autonomous, Market-based Supply-chain Management agent," *Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2004.

[6] E. Dahlgren and P. R. Wurman, "PackaTAC: A Conservative Trading Agent," *SIGecom Exchanges*, vol. 4, No. 3, pp. 33–40, 2004.

[7] C. Kiekintveld, M. P. Wellman, S. Singh and V. Soni, "Value-Driven Procurement in the TAC Supply Chain Game," *ACM SIGecom Exchanges*, vol. 4, No. 3, pp. 9–18, 2004.

[8] C. Kiekintveld, Y. Vorobeychik and M. P. Wellman, "An Analysis of the 2004 Supply Chain Management Trading Agent Competition," *IJCAI-05 Workshop on Trading Agent Design and Analysis*, 2005.

[9] M. P. Wellman, J. Estelle, S. Singh, Y. Vorobeychik, C. Kiekintveld and V. Soni, "Strategic Interactions in a Supply Chain Game," *Computational Intelligence*, vol. 21, 2005.