A Dynamic Programming Approach for Agent's Bidding Strategy in TAC-SCM Game

Xiaoqin Zhang*
Soheil Sibdari†
Saban Singh

University of Massachusetts Dartmouth
North Dartmouth, Massachusetts 02747

October 10, 2010

Abstract

Intelligent agents have been developed for a number of e-commerce applications including supply chain management. In Trading Agent Competition for Supply Chain Management (TAC SCM), several manufacturer agents compete in a reverse auction in order to sell assembled computers to customers. The manufacturer agent's tasks include acquiring supplies, selling products, and managing its local manufacturing process. The agent decide whether to accept an arriving bid in order to maximize its long term expected profit. In this paper, we use dynamic programming to provide a pricing strategy for the Trading Agent Competition in Supply Chain Management (TAC SCM). We consider a competition between an individual manufacturer agent and other automated agents in TAC SCM. The experiment results show that this strategy improves the agent's revenue significantly comparing to several other heuristics in the current practice. This approach can also be applied to similar bidding problems in other e-commerce applications.

1 Introduction

Intelligent agent technology has been applied to various stages of Business-to-Customer (B2C) e-commerce process such as product brokering, merchant brokering and negotiation (Pivk and Gams. 2000). On the other hand, agent technology is also very promising in handling a number of complex issues in Business-to-Business (B2B) e-commerce such as supply chain management (Fox, Barbuceanu, and Teigen 2000). Simulated and organized competition is an effective way to understand a market and to examine the final results of some marketing and operational policies without the risks and complexities of real-world environment. The Trading Agent Competition in Supply Chain Management (TAC SCM) is a testbed to stimulate research in this area.

*Computer and Information Sciences Department, College of Engineering
†Corresponding Author. Decisions and Operations Sciences, Charlton College of Business

The TAC SCM is a simulated computer manufacturing market in which a finite number (currently six) of independent software agents compete in a supply chain game. The agents maximize their profits by competing in a reverse auction to sell computers to a random number of customers. This game is studied over a finite horizon. At the beginning of each period (i.e. day), customers send out requests for quote (RFQs); each RFQ includes the number of requested products and their characteristics, the reserve price per unit, the due date expection, and a penalty the manufacturer needs to pay if the due date is missed. For each RFQ, the manufacture agent needs to decide whether to bid on this RFQ and if so at what price. This is referred to as the bidding problem. On the other hand, customers select their best bid and upon the acceptance, the manufacturer agent needs to acquire the components from the suppliers. and then conducts a local manufacturing process to assemble products. Finally, the agent receives its payment upon the delivery of the finished products to the customer.

Because of limited resources, the agent should consider the future conditions and its impact on current decisions. If the agent accepts many RFQs, it might loses the chances of accepting a valuable bid toward the end of the game due to lack of resource and if it rejects many bids, there might be unused resources at the end. By bidding to the right RFQ at the right price, the manufacturer agent can reserve its production capacity for future demand with possibly higher revenues. In current systems in TAC SCM, the agents myopically consider the current period and immediate capacity, and ignore future demand and its impact on arriving RFQs. A practical method that evaluates the agent's policy over the whole sales time horizon can help the agent to maximize its expected profit. We use a dynamic programming model that closes this gap and to provide the best bidding strategy for the agent.

To solve this bidding problem, we first simplify our multi-agent model to overcome with the complexity caused by many variables. More specifically, we define a finite set of RFQs based on possible combinations of *product types*, *due dates*, *reserve prices*, and *penalty costs*. We then map the bidding problem into a stochastic decision problem by estimating the "*value*" of each product. This value is estimated using experimental data from the current TAC SCM game. We use this value as the minimum price that the agent can accept for each product. On the other hand, the customers also receive bids from other agents and choose the one with the lowest offer. We model the impact of this competition on the firm's decision by including a parameter that measures the probability of accepting an offer by the customers. These offers differ only in price, and therefore the acceptance probability can be modeled as a function of the offered price. Finally we use dynamic programming to calculate the optimal bidding strategy that the agent uses in this competition.

In order to provide an evaluation baseline for our model, we also introduce a heuristic using which the agent desires to maximize the immediate revenue (i.e. considering the current period only). Because the firm does not consider the long term impact of its decision on future profit, we call this heuristic *myopic policy*. However, the bidding strategy in a given day impacts the agent's ability to compete in future. By accepting a bid, the agent enters into a contactual agreement and needs to assign resources and facilities to fulfill that demand, which reduces its capacity and therefore its future decision of whether to accept or reject a bid.

The literature in TAC SCM is vast and indirectly belongs to a broad stream of supply chain management. Most studies focus on agent design. Ketter et al. (2008) provides a comprehensive survey of the current

literature in TAC SCM agent design. Benisch et. al. (2004) describes the design of a specific agent in the TAC SCM game, called Botticelli. This agent competes with other agents in order to win the customers and to negotiate with suppliers to procure product components. They used stochastic programming for the bidding and scheduling problem to calculate the optimal solutions. Another optimization technique for the TAC SCM agents is provided by Burke et al. (2006 and 2008). They provide a policy of what customer orders to bid on and what prices to bid by combining constraint-based optimization and learning of market conditions. In their model, the agent maximize the profit subject to capacity and supply constraints. Kiekkintveld et al. (2006) address two issues that a manufacturer agent should consider. First, how to deal with the inherent uncertainty in different aspects of the market and second, how to compete in the market with other agents who play strategic. Greenwald et al. (2009) presents a bidding strategy for the TAC SCM game using the greedy algorithm. Their marginal bidding provides an incremental solution to incorporate general acceptance conditions such as scheduling and component constraint. However, they have ignored the market competition and focused only on the decision-theoretic optimization problem. For researches directly on TAC SCM see Bell et al. (2004) and Benisch et al. (2004).

Our work differs from the existing literature because we consider time inter-dependency in our model. We use dynamic programming that not only considers the expected profit from the immediate bid but also considers the impact of the current decision on future profit. We specifically categorize the RFQs into different classes based on their characteristics such as product type, reserve price (maximum bid), due date, etc. We then define the manufacturer agent capacity as the state variable of a discrete event dynamic programming problem with time of the bid as its stage variable. Upon the arrival of a new customer RFQ, the manufacturer agent (here after *agent*) observes the characteristics of the RFQ, reviews its previous contract and decides whether to bid on this RFQ.

The paper is organized as follows. We begin by describing the TAC-SCM game in Section 2. In Section 3, we develop our mathematical model. In Section 4 we provide two different baseline agents in order to compare with our dynamic programming model. One baseline agent performs based on the the settings of the current system and the other one is a myopic agent who only considers the current profit. Section 5 shows the design of an intelligent agent who uses our mathematical model. In Section 6, we provide the details of our experiment and the evaluation results. Section 7 concludes.

2 TAC-SCM Game

The TAC-SCM game simulates a real world supply chain scenario. The game is operated over a period of 220 days, each day being simulated as 15 seconds. Six agents compete against each other, the one with the most money at the end wins the game. The game has three main entities: customers, manufacturers and suppliers. This game is implemented as a client-server application where the suppliers, customers, and the bank represent the server and the manufacturer agent represents as a the client. The agent interacts with the server twice a day (we define one day as a period). At the beginning of a new period, the agent receives previous information such as PC price, customer RFQs, and supplier bids. It then deliberates and carries out this computations at the end of the day and sends back the information to the server.

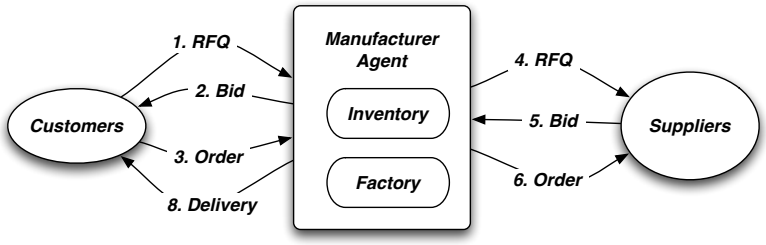

Figure 1: Manufacturer agent's interactions with customer and suppliers

There are three types of entities in the game: customer, manufacturer and supplier. Their interactions are depicted in Figure 1.

Customers order PCs from the manufacturers. They send RFQs to all manufacturers, each RFQ consists of the following information.

1. **PC type** (*productID*) There are 16 types of PCs, which fall into three market ranges, namely low, medium and high range.
2. **Reserve price** The maximum unit price the customer is willing to pay.
3. **Due date** The date by then the orders must be shipped to the customer.
4. **Penalty amount** The amount the manufacturer must pay per day for late delivery.

Manufacturer agents are responsible for producing PCs ordered by customers. The manufacturer agent receives multiple RFQs from customers every day. For each RFQ, the manufacturer agent decides whether to bid for it, and if so at what price. The customer selects the agent with the best bid (i.e. lowest asking price) and enters into a contractual agreement with that agent. Upon receiving an order from the customer, the agent then analyzes what types of components and how many of them are needed to fulfill this order. Keeping in mind the components in inventory, it sends RFQs to suppliers for additional components needed. Similarly, the manufacturer agent selects the best bid submitted by the suppliers. It then provides a daily production schedule to its factory, which consists of the product type (1-16) and its quantity. The completed products will be shipped to the customer.

Each agent owns a production factory that, in this game, is measured by production cycles. In our game setting, each agent has 2000 production cycles per day and each PC type consumes different amount of production cycles. At the beginning of each period, the agent sends a production schedule to its factory. The production schedule specifies the product type (1-16) and their quantities. The factory can produce the PCs only if sufficient components and enough production cycles are available.

The manufacturer also owns a warehouse to store both components and the finished PCs with a daily storage cost charged for each unit. A daily storage cost is charged for both components and finished PCs kept in the inventory. The storage cost for component is a percentage of the nominal cost of the component, and the storage cost for a finished PC is a percentage of the sum total of the nominal cost of the components used in the PC. In each game, to determine these percentages, a random value between 0.25 and 0.5 is chosen and remains fixed throughout the game.

The agents also own a bank account. When an manufacturer agent buys components from a supplier, the

amount is deducted from its bank account and when it delivers finished PCs to customers, the payment is made to the account. An agent's bank balance can be negative, which means the agent is getting a loan from the bank, in which case a loan interest applies. Similarly, positive savings in the account generate interest for the agent. The loan interest rate is a random value between 0.06 and 0.12, and the saving interest rate is half of the loan interest rate. These rates are fixed throughout the game.

Suppliers are entities that produce the components required to build a PC. There are eight suppliers in the game. The suppliers are revenue-maximizing agents with a fixed production capacity per day. The production is based on a make-to-order basis, i.e no production is made without an order. When the supplier receives a RFQ from the manufacturer, it checks if it can offer a price below the reserve price. The supplier can also offer a reduced quantity or it can negotiate with the manufacturer agent about the due-date. The production capacity of a supplier is also limited and it can produce a certain quota of components per day. The nominal capacity for a supplier is set to 550 components per day. The actual capacity ranges within 35% of the nominal capacity. After describing the details about the TAC SCM game, we will present the model we developed to handle the bidding problem of the manufacturer agent in the next section.

3 Model

Consider an agent who produces a finite number of PC products. To assemble a PC, depending on the product type, the agent needs to spend a specific amount of time that we call *production cycle*. The game is performed during a fixed period of time that we divide it into equal intervals (say T time periods) such that at most one RFQ can be received in each period. We count the time periods in chronological order so that period T represents the last period of the game.

Each RFQ, say RFQ i, consists of *productID*, *quantity*, *reservePrice*, *dueDate*, and *penalty*. The productID determines the configuration of the PCs and determines what components are used in each product (i.e. bill of materials). Currently, there are 16 types of productID. In each RFQ, the quantity, q_i, is a discrete variable whose value is distributed uniformly in $[q_{min}, q_{max}]$. The customer also specifies dd_i that is a due date of receiving the product. The due date is the current date plus a uniformly chosen order lead time in the interval $[due_{min}, due_{max}]$. The customers also include their reserve price ρ, which is uniformly chosen in $[\rho_{min}, \rho_{max}]$. Finally, penalty x is uniformly chosen in $[\Psi_{min}, \Psi_{max}]$, and is the cost that the agent is committed to pay to the customer if the product cannot be delivered by the due date. For each RFQ, we define an *offer vector* of $I = \{k, q_i, dd_i, \rho_i, x_i\}$ that specifies the productID, quantity, due dates, reserve price, and penalty cost, respectively.

We now categorize the RFQs by defining *RFQ type* that differentiates between RFQs not only by their PC types but also by their reserve prices, due dates, and penalty costs. For example, two RFQs with the same productID, due date, and reserve price but with different penalty costs are considered two different types. To define RFQ types we first classify due date, reserve price, and penalty cost as three types - high, medium, and low, using appropriate intervals. Therefore there are a total of M RFQ types, which based on our setting $M = 3 \times 3 \times 3 \times 3 = 81$, where the four 3s stand for three types (high, medium, low) of PC type, reservation price, due date, and penalty cost. We define P_i as the probability of receiving RFQ type i in each period. At

the beginning of each period, upon the arrival of an RFQ, the agent should decide whether to bid and if so at what price. We use a dynamic programming method to solve this problem with the goal to maximize the agent's total expected profit from period t until period T.

In order to determine the expected profit of accepting an order based on an RFQ, we need to determine the marginal profit of each RFQ type. We use historical data (i.e. different experiments using the existing game) to estimate the marginal revenue that can be generated from each RFQ. The expected revenue (here after *revenue*) for RFQ $i, i = 1, \cdots, M$ is f_i and we assume that this revenue is given and remains constant over all time periods. Note that f_i consists of all costs toward the production of RFQ i including material and assembly costs. Using the historical data, other parameters for each production type such as the required capacity and marginal revenue can be estimated. We denote the number of production cycles that is needed to assemble j units of PC type i by c_{ij}.

Furthermore, we use the historical data to model the impact of competition in the market. As the TAC SCM game is a multi-agent game, we also measure the interaction between different agents using the historical data. We measure the probability of accepting a bid by a customer as a function of the bid price generated by the manufacturer agent. In other words, we determine the probability that the offers made by other agents be less appealing to the customer. A mathematical model that explicitly addresses this competition and provides the possible Nash equilibrium is an interesting extension of our model. We define $g_i(x)$ as the probability of accepting a bid at price x by the customer and, to ease the reading, we use g_i instead of $g_i(x)$. In Section 4.2, a heuristic is provided to determine the bidding price for RFQ type i.

Lets define $J(c, t)$ as the agent's expected profit at beginning of period t when its production capacity is c. The following dynamic program can be used to calculate the optimal decision of whether to bid or to ignore an RFQ of type i.

$$J(c,t) = \sum_{i=1}^{M} \sum_{j=1}^{K} P_i * Q_{ij} * max(J(c,t+1), g_i * (j * f_i + J(c - c_{ij}, t+1)) + (1 - g_i) * J(c, t+1)) \quad (1)$$

with boundary conditions of $J(0,t) = J(c,T) = 0$ for all values of $c \geq 0$ and $0 \leq t \leq T$, and where:
i - The RFQ type. Currently ranges from 1 to 81.
K - Maximum number of products in a RFQ . Currently ranges from 1 to 20.
P_i - The probability of receiving a RFQ of type i.
Q_{ij} - The probability of requesting j units of products in a RFQ of type i.
g_i - The probability that a bid for RFQ type i will be accepted.
f_i - The revenue for each unit of product in RFQ type i
c_{ij} - The number of production cycles needed to produce j units of products in RFQ type i.

The values for P_i, Q_{ij}, g_i and f_i were gathered from the experiments described in Section 5. For tractability, we assume that the agent does not hold any inventory, or equivalently, we assume that inventory cost is zero.

4 Baseline agents

To provide proxies for our model, we build a manufacturer agent, named Agent B, using some of the best existing practices in TAC-SCM literature. We then modify this agent by using the optimized bidding strategy found with the model presented in Section 3, that is called Agent A. In addition, in order to fulfill the competition between six manufacture agents (that is required by the game setting), we introduce four dummy agents provided by the game designer. In the rest of this section, we will present how the dummy agents perform, i.e. how they make decisions on sales, procurement, production and delivery.

4.1 Dummy agents

Based on the game setting, the existing dummy agents behave in the following ways. In the sales side, the dummy agent bids for an RFQ only if the due date for the RFQ is greater than five days from the current date, given the component ordering process takes a minimum of five days, and the dummy agent only orders components from the suppliers after receiving the PC order. In addition, the condition $ReservePrice > (BasePrice * 0.9)$ must be hold. The bid price is then calculated using the following formula:

$$\begin{aligned} BidPrice = \quad & (BasePrice * 0.9) + \\ & [ReservePrice - (BasePrice * 0.9)] * (1.0 - RandomFactor * PriceDiscountFactor) \end{aligned} \tag{2}$$

where $BasePrice$ is the sum of the nominal price of all components for the PC, which is given in the product specification. $RandomFactor$ is a random value between 0 and 1, and $PriceDiscountFactor$ is 0.3.

In terms of procurement, the dummy agent sends RFQs for components to suppliers only after it receives orders from the customer. If there is more than one supplier for the component, it randomly chooses a supplier. The reserve price is set to zero. Note that when a reserve price of zero is provided for a RFQ, it means that there is no constraint on the supplier's bid price and that the supplier can quote any price. However, if the reserve price is higher than zero, the supplier can only bid with a price less than the reserve price.

The production occurs within the limitation of components and production cycles, the dummy agent produces to fulfill the orders with the most recent due dates. The products must be delivered within the limitation of available finished PCs, orders with immediate due dates (tomorrow) are shipped from the inventory.

4.2 Intelligent Agent B

We design intelligent agent B according to an inventory-driven strategy, which is introduced by the agent SouthamptonSCM in 2004 (He et al. 2006). Regarding the sales component, it only bids for RFQs according to what is available in its inventory. That way, the agent avoids paying penalties for overcommitting itself, since the quantity of PCs depends on the availability of components and the factory cycles. This method can

lead the agent to receive less orders from the customers and can potentially result in less factory utilization and revenue.

Agent B sends the RFQs one at a time according to two criteria. First, there should be enough cycles available to meet the required quantity of products in the RFQ (the initial available factory cycle of 2000). And second, for the production of the requested products in the RFQ, enough inventory should be available including both the finished PCs and the components in the inventory.

Since customers can decline the bids provided by the agent, instead of actual quantity, we introduce required quantity that is deducted from both the inventory and the production capacity. The required quantity can be calculated as follows.

$$\begin{aligned} RequiredQuantity &= ActualQuantity * AcceptanceRate \\ AcceptanceRate &= \frac{AverageOrderCount}{AverageOfferCount} \end{aligned} \tag{3}$$

The average order count and average offer count are taken over the past 5 days. If a bid is sent for this RFQ, the remaining cycles and inventory are used based on the next RFQ. The bid price for an RFQ is calculated as follows.

$$Bidprice = max(MinimumPrice, EstimatePrice) \tag{4}$$

where ***MinimumPrice*** is also calculated as $MinimumPrice = BasePrice * 1.1$ with a guaranteed profit margin of at least 10%. Every 20 days, the manufacturer agent receives a market report with the average price paid for each of the 16 Product types to all the competing manufacturers in the past 20 days. The $EstimatePrice$ is set differently depending on the number of days (d). If $d \geq 20$, then the agent receives a price report, and the estimated price is calculated as:

$$EstimatePrice = CostFactor * AverageProductPrice$$

where $AverageProductPrice$ is taken from the 20-day market report.
If $d < 20$, no market report is available yet, so the estimated price is calculated as:

$$EstimatePrice = CostFactor * BasePrice$$

where, $CostFactor = LowestProductPrice_{day\#(d-1)}/BasePrice$, and $BasePrice$ is the sum of the nominal price of all components for the PC. The nominal price of components are given. We currently have 16 product types. For example, Type 1 PC has four components, the nominal price for each component is 100, 200, 300 and 400 respectively, then $BasePrice(PC_{type1}) = (100 + 200 + 300 + 400) = 1000$.

Toward the end of the game there is a sharp peak in acceptance rate since the other agents become more reluctant to send bids. In fact, the predicted acceptance rate increases by 70% during the last two periods of the game. In addition, agents provide a discount of 30% on all offers in order to deplete inventory.

We next describe the procurement strategy of the game for Agent B. In order to maintain high factory utilization (80-100%), a very high inventory level is required. So the inventory threshold for components is fixed at 1200 and that for finished PCs is set at 40. During the first period (i.e. day 0), an initial order is

placed for all components, equal to the threshold value of 1200. Additional RFQs are sent to the suppliers to maintain the fixed threshold 1200 throughout the game. The idea of ordering large quantity of components during the first two days and then maintaining the threshold is adopted from Agent Mertacor's design (Kontogounis et al. 2006). The threshold is decreased toward the end of the game (i.e. after the 200th day). The threshold is 400 between day 201 to 205, 100 between days 206 and 210, and 0 between days 211 to 219. Under this procurement strategy the reserve price of the components is calculated as follows.

$$ReservePrice = BasePrice + (0.1 * BasePrice)$$

where if $d > 20$, $BasePrice$ is the average market price from the market report. And if $d <= 20$, $BasePrice$ is the mean cost of the component according to the agent's record over the past days. Note that the due day is set to four days from the current day.

Regarding the production strategies, Agent B uses a greedy approach, which caters to orders whose due date is closer. In addition to producing the PCs based on orders, if enough components and production cycles are available, Agent B also uses a make-to-stock policy in order to maintain a threshold of 40 PCs of each type in inventory (Collins et al. 2007).

For delivery, Agent B also uses the greedy approach. It ships PCs available in the inventory to meet deadlines even if those PCs were produced for a different order (Collins et al. 2007).

5 Optimizing the Bidding Strategy - Agent A

The main difference between Agent A and Agent B is that Agent A uses the Expected Profit Matrix (EPM) to decide whether to bid on a customer RFQ. In order to use this matrix, we need first obtain this matrix.

5.1 Obtaining Matrix

To build the matrix, the first task is to categorize RFQs.

RFQ categorization

There are four attributes about RFQ specified in the specification document (Collins et al. 2006): product type, reserve price, due date and penalty. Total of 81 RFQ types were defined using these four attributes. Each attribute can be classified in three categories based on given specification. Product type can be low, medium (mid), or high. Due date falls into one of the three following categories: short [3-6 days], medium [6-9 days], and long [9-12 days]. Reserve price can be low [75 - 90%], medium [90 - 100%], or high [110 - 125% of base price]. Penalty is categorized as low [5-8%], medium [8-12%], or high [12-15% of reserve price] as well.

Collecting Data

To generate $J_M(c, t)$, the expected profit the manufacture agent can make at time t with production capacity as c (Equation 1), the following data are gathered from the experiments for each RFQ type i: the probability of getting an RFQ of type i, P_i; the probability of having j units of products in one RFQ of type i, Q_{ij}; the probability that a bid for that RFQ type is accepted by the customer, g_i; and the revenue

that can be generated from each product in RFQ type i, f_i. The revenue is calculated as the following:
$Revenue = (SalePrice - ComponentCost - Penalty)$
$SalePrice$ - the average price at which a product of that RFQ type is sold at.
$ComponentCost$ - the average price paid for the components.
$Penalty$ - the average penalty amount that is accumulated for one product of that RFQ type.

Among this data, revenue f_i and acceptance rate g_i are highly dependent on other agents' strategies. Since we are planning two types of evaluation tests: one intelligent Agent (Agent A or B) competes with 5 dummy agents, and Agent A and Agent B and 4 dummy agents compete together. So, we collected data in the following two experiment settings and generated two estimated profit matrices that are used by Agent A in above two tests, respectively. In the first setting, there are one Agent B and 5 dummy agents. In the second setting, there are two Agent B and 4 dummy agents. Total of 10 games were generated for each setting and the average values are used to generate the matrix.

Calculating EPM

Our decision problem is that for each arriving RFQ, should the agent bid for this RFQ (assuming that the bid price has been determined) or not (to reserve the production capability for other RFQs)? This decision can be made using the EPM, $J_M(c,t)$, the expected profit the manufacture agent can make at time t with production capacity as c, as we described in Equation 1.

$J(c,t) = \sum_{i=1}^{M} \sum_{j=1}^{K} P_i * Q_{ij} * max(J(c,t+1), g_i * (j * f_i + J(c - c_{ij}, t+1)) + (1 - g_i) * J(c,t+1))$

The values for P_i, Q_{ij}, g_i and f_i are gathered from the experiments described earlier. Dynamic programming method is used to generate a matrix of size 22000 (c) x 320 (t). This size was chosen for the following reasons. The maximum due date for any RFQ cannot be more than 12 days from the arrival date. So, 11 days was chosen as the planning period. Hence, the total capacity at our disposal was (2000*11). Additionally, according to the specification document (Collins et al. 2006), the maximum number of RFQs that can be sent out by the customer in one day is 320. So we split one day as 320 time slices and there is no more than one RFQ arriving on within any one time slice.

5.2 Applying Matrix

Agent A is similar to Agent B, except on the following modifications. Agent A uses the EPM to decide whether to bid for a RFQ or ignore it, during a 11 day planning period. Agent A also uses a capacity vector to prevent it from overbidding and incurring penalty. The capacity vector is a 11 day window ranging from (current day + 1) to (current day + 11). Initially, the available cycles on each day are 2000. Every day before the RFQ selection process, the capacity required for current orders is deducted. The remaining capacity can then be used for the selection process. The acceptance rate for Agent A is similar to Agent B's, the only difference being that Agent A's acceptance rate is inflated by 10% so as to avoid penalty in cases of over bidding. The amount of components in the inventory is deflated by 200. For example, if there are 600 units of component A, then bidding is done as if there were only 400 components. The reason for the deduction is that the production schedule is created after the bidding process is done. So in the bidding process, the agent needs to reserve a certain portion of the inventory for the current day's production. The production

Table 1: Results for the First Setting: one Intelligent agent with 5 dummy agents

	Agent A		Dummy Agents		Agent B		Dummy Agents	
	Average	STDV	Average	STDV	Average	STDV	Average	STDV
Revenue (in Millions)	16.26	3.63	6.78	3.89	8.01	4.13	4.16	1.84
Factory Utilization (%)	95.21	0.78			91.40	0.77		
Total RFQ	40712	3946			40521	3487		
Total Offer	16451	321			14063	300		
Total Order	7538	97			6512	133		
Penalty (in Millions)	0.797	0.356			0.156	0.137		
Penalized RFQ	137	44			36	22		

cycles per day is also deflated by 10% (100 cycles). This is also done to make room in case of overbidding. The generated matrix is then used to decide whether to bid for a RFQ or ignore it. This is done as shown below.

For ($t = 1$, $t <= 320$, $t++$)

for the RFQ R of type i received at t,

if bid for R, $revenue_A = g_i * (J_M(c - c', t + 1) + Revenue(R))$

$+(1 - g_i) * J_M(c, t + 1)$, where c' is the capacity needed for R

if ignore R, $revenue_B = J_M(c, t + 1)$

if $revenue_A > revenue_B$, then bid for R otherwise ignore R.

6 Experiment Results

In this section, we present the experimental results in two different settings.

6.1 First Setting: One Intelligent Agent and Five Dummy Agents

There are two types of setting and in each setting two types of competitions have been run. The first competition includes Agent A and five dummy agents; and the second competition include Agent B and five dummy agents.

Each type competition is repeated 8 times and the average values and standard deviations are shown in Table 1. The average revenue made by Agent A (16.26) is almost twice as the average revenue made by Agent B (8.01). Both agents receive about the same number of RFQs from customers. Agent A actually produces 15.7% more orders compared to Agent B. Agent A maintains a slightly higher factory utilization percentage (95.21%) compared to Agent B (91.40%). So we can conclude that most of the extra revenue made by Agent A is from the careful selection of RFQs to respond by using the expected profit matrix.

Table 2: Results for the Second Setting: Agent A, B and 4 dummy agents

	Agent A		Agent B		Dummy Agents	
	Average	STDV	Average	STDV	Average	STDV
Revenue (in Millions)	16.66	2.82	13.0	5.2	7.6	1.9
Factory Utilization (%)	91.46	3.54	86.68	5.17		
Total RFQ	39726	4226	39726	4226		
Total Offer	24732	913	22473	1210		
Total Order	7186	315	6364	259		
Penalty (in Millions)	0.662	0.151	0.361	0.301		
Penalized RFQ	131	32	76	59		

6.2 Second Setting: Two Intelligent Agents and Four Dummy Agents

In the second setting, the competition is among two intelligent manufacture agents (A and B) and four dummy agents. The competition is repeated 10 times and the average values and standard deviations are shown in Table 2. In this setting, Agent A and B compete directly with other 4 dummy agents. The average revenue gained by Agent A (16.66) is 28% more than Agent B (13.0). Both agents receives exactly the same set of RFQs from customer, Agent A responded to 8.7% more RFQs and received 12.9% more orders from customer comparing to Agent B. So part of the extra revenue obtained by Agent A should be attributed to the usage of the expected profit matrix. The improvement of agent A's performance over agent B is less significant in this setting compared to the first setting. The reason could be the bigger gap between the environment (2 Agent B + 4 dummy agents) where the data is collected to generate the EPM and the environment (Agent A + Agent B + 4 dummy agents) where the EPM is applied. The more realistic data the EPM is built upon, the better performance can be achieved when using the EPM.

7 Conclusions

We described a supply chain model in Trading Agent Competition (TAC), an environment that allows the testing of various approaches for several interrelated problems. We focus on the RFQ selection problem for manufacture agents. We built an expected profit matrix using dynamic programming method and then used this matrix to make bidding decisions. We implement this approach in an intelligent agent - Agent A, which is otherwise the same as Agent B - an intelligent agent that combines some of the best practices in literature for other problems. The experimental results show that the modification for Agent A improves its performance significantly, hence demonstrate the power of the formal methods on solving supply chain problems. The future work includes extending this model by calculating the optimal bidding price, and to find Nash equilibriums by explicitly modeling the competition in the market.

References

Bell, S.; Benisch, M.; Benthall, M.; Greenwald, A.; and Tschantz, M. C. 2004. 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*, 21–27.

Benisch, M.; Greenwald, A.; Grypari, I.; Lederman, R.; Naroditskiy, V.; and Tschantz., M. 2004. Botticelli: A supply chain management agent. In *Third International Conference on Autonomous Agents and Multiagent Systems*, 11741181.

Burke, D. A., and Brown, K. N. 2008. A constraint based agent for TAC SCM. *Working paper*.

Burke, D. A.; Brown, K. N.; Tarim, S. A.; and Hnich, B. 2006. Learning market prices for a real-time supply chain management trading agent. *AAMAS 2006 Workshop on Trading Agent Design and Analysis*.

Collins, J.; Arunachalam, R.; Sadeh, N.; Eriksson, J.; Finne, N.; and Janson., S. 2006. The supply chain management game for the 2007 trading agent competition. Technical Report CMU-ISRI-07-100, Carnegie Mellon University, Pittsburgh, PA 15213.

Collins, J.; Ketter, W.; Gini, M.; and Agovic, A. 2007. Software architecture of the minnetac supply-chain trading agent. Technical Report 07-006, University of Minnesota, Dept of Computer Science and Engineering, Minneapolis, MN.

Fox, M. S.; Barbuceanu, M.; and Teigen, R. 2000. Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems* 12(2-3):165–188.

Greenwald, A.; Naroditskiy, V.; Odean, T.; Ramirez, M.; Sodomka, E.; Zimmerman, J.; and Cutler, C. 2009. *Marginal Bidding: An Application of the Equimarginal Principle to Bidding in TAC SCM*. Springer Verlag. 217–239.

He, M.; Rogers, A.; Luo, X.; and Jennings, N. R. 2006. Designing a successful trading agent for supply chain management. In Nakashima, H.; Wellman, M. P.; Weiss, G.; and Stone, P., eds., *AAMAS*, 1159–1166. ACM.

Ketter, W.; Collins, J.; and Gini, M. 2008. A survey of agent designs for TAC SCM. In *AAAI-08 Workshop on Trading Agent Design and Analysis (TADA-08)*.

Kiekintveld, C.; Miller, J.; Jordan, P. R.; and Wellman, M. P. 2006. Controlling a supply chain agent using value-based decomposition. *The Proceedings of 7th ACM Conference on Electronic Commerce* 208–217.

Kontogounis, I.; Chatzidimitriou, K. C.; Symeonidis, A. L.; and Mitkas, P. A. 2006. A robust agent design for dynamic scm environments. In Antoniou, G.; Potamias, G.; Spyropoulos, C.; and Plexousakis, D., eds., *SETN*, volume 3955 of *Lecture Notes in Computer Science*, 127–136. Springer.

Pivk, A., and Gams., M. 2000. Intelligent agents in e-commerce. *Electrotechnical Review* 67(5):251–260.