A Marginal Revenue Approach to Bidding in TAC SCM

John Donaldson
jwd@cs.brown.edu

December 17, 2006

Abstract

We describe two strategies for bidding on customer RFQs in the Supply Chain Management Trading Agent Competition. The first strategy, used by Botticelli in previous years, is to model and solve the problem as integer linear program. The second strategy is a greedy algorithm where customer demand is scheduled according to the predicted marginal profit. Both strategies have been implemented in the Botticelli agent. We consider the problem of bidding on today's RFQs with and without knowledge of the future customer demand. Simulation results for various demand scenarios are presented.

1 Introduction

Bidding in response to customer demand is a central function of an agent operating in the context of a dynamic supply chain. The Trading Agent Competition in Supply Chain Management (TAC SCM) offers a simulated environment for the evaluation of competitive bidding strategies. Agents in the TAC SCM game face three simultaneous and interrelated problems. First, the problem of **procurement,** where by agents have to purchase the components required for assembling finished computers. Second, the problem of competitive **bidding** in reponse to changing customer demand for finished computers. Finally, the problem of **scheduling** production of various computers across limited factory capacity.

Although we focus on the solution to the bidding problem, it should be clear that the optimal solution to each problem depends on the soltuion to the other problems. For example, a naive profit-maximizing agent wishes to win many orders at high prices, so its seems wise to bid on all available RFQs. On the other hand, winning more orders than can be produced with available factory capacity will result in penalties and sub-optimal revenue. Further, a robust agent needs to consider the availability and cost of components when formulating bids. For the purposes of this paper we are generally concerned with the problem of bidding and scheduling in so far as it solved in route to

solving our bidding problem. Procurement is not considered. Our simulation environment provides an infinite supply of every component at no cost.

2 Description of the Bidding Problem

The TAC SCM bidding problem takes the form of a reverse auction where sellers compete to provide the lowest bids in response to customer RFQs. Each customer RFQ is a request for a fixed number of a particular type of computers delivered within a specified number of days with a reserve price. The agent's problem then is to submit a mapping from the current day's RFQs to bids.

In order to determine a profit-maximizing mapping of RFQs to bids, an agent needs a model of the probability that a particular bid at a particular price is converted into an order. For each Stock Keeping Unit (SKU), the TAC SCM server provides the previous day's maximum and minimum prices of computers sold. Botticelli combines this information with its own bids, to fit a linear offer-acceptance cdf. Other more complex models of offer acceptance are also possible.

For the purposes the simulations presented in this paper, we assumee the model is known to the agent and perfect, that is the actual probability that an agent recieves an order in response to a bid is exactly that predicted by the agents own model. Put simply, the simulator and the agent share the same model.

The TAC SCM game takes place over 220 simulated days. An agent may wish to consider not only today's RFQs and today's production capacity, but several days of predicted future demand and production capacity. In the actual TAC SCM game, customer demand is drawn from a Poisson distribution. The mean of the Poisson distribution is adjusted each day by a trend. The trend itself is in turn adjusted by a bounded random walk. Again, there are methods for reaching an estimate of future customer demand in the TAC SCM game. For the purposes of this paper we assume future customer demand is fully and perfectly known.

2.1 Formal Description of the Bidding Problem

An agent wishes to submit bid on a subset of the current day's RFQs that maximizes the total expected revenue. Formally,

$$\max \sum_i P\left(R_i = 1 | b\right) * b_i$$

where i indexes the RFQs, b is the bid amount, and $P\left(R\right)$ is the probability of receiving an order for a particular bid.

All orders must completed before due date, to assure this is possible, our agent will bid such that the expected number of cycles in at or below our factory capacity. Formally,

$$\sum_i P\left(R_i = 1|b\right) * cycles(R) \leq FactoryCapacity$$

where i,$P\left(R\right)$,b are as above and $cycles(R)$ is a function that takes an RFQ and returns the required number of factory cycles required to convert that RFQ to the requested number of finished computers.

Bids are additionally subject to several obvious technical restrictions such as a only a single bid is allowed for each RFQ or no bid above the customer's reserve. We do not itemize those constraints here, but they are required elements of a complete ILP solution.

2.2 An Integer Linear Program Approach

Since 2003, Botticelli has used a single-day integer linear programing approach to solving the bidding and scheduling problems. The objective function maximizes both the revenue from fulfilling current orders before they expire and the expected revenue from RFQs. We divide our possible bids uniformly from 0(no bid) to 99% (a bid low-enough that our modeler returns a 99% chance of winning the order) Formally,

$$\max \sum_j O_j * o_j + \sum_i \sum_m P\left(R_{i,m} = 1|b_{i,m}\right) * b_{i,m}$$

where O is a zero-one variable indicating if the jth order is filled before due-date, o_j is the revenue available for filling that order, and R, i, and b apply to RFQs as above and m indexs the number of discrete bid values considered.
Our agent seeks to maximize this objective function subject to several constraints. First, for each SKU, the combination of available inventory and production must meet or exceed the the total quantity of orders fulfilled. Formally,

$$\sum_j O_{s,j} * q_j \leq Y_s + I_s$$

where s indexes the 16 different SKUs, j indexes orders within a particular product type, $O_{s,j}$ is a zero-one indicator variable as described above, q_j is the quantity of an order, Y_s is the total production of a particular SKU, and I_s is the start of day inventory for a particular SKU.

Finally, production cannot exceeed factory capacity on any day. Formally,

$$\sum_s Y_s * cycles(s) \leq FactoryCapacity$$

where Y_s is the number of units of production of a particular SKU and cycles(s) is a function that accepts a SKU and returns the number of factory cycles required to.

We can easily extend both of these constraints to a multiday bidding situation. The RFQs in our planning window will now include both actual customer RFQs sent by the TAC SCM server and modeled future customer demand. Extended to multiple days the objective function becomes

$$\max \sum d \sum_j O_{d,j} * o_j + \sum_d \sum_i + P\left(R_i = 1 | b_i\right) * b_{d,i}$$

The production sufficiency constraint becomes

$$\sum_d \sum_j O_{s,d,j} * q_j \leq \sum_d Y_{d,s} + I_{0,s}$$

Again, production cannot exceeed factory capacity on any day.

$$\sum_s Y_{d,s} * cycles(s) \leq FactoryCapacity$$

Although this ILP produces a valid production schedule and bidding solution, it does not provide for component procurement. Procurement is handled via a heuristic that seeks to combine long and short term procurement strategies to maintain component levels sufficient for the current level of production. Regardless, the ILP must also encode component constraints. Our agent cannot produce a computer for which it has not already obtained the required components. We do not itemize component constraints here. Similarily, we leave out several technical constraints mentioned above such as the requirement to submit at most a single bid for each RFQ.

The number of variables in the ILP grows as we increase the number of days in the schedule, which adds additional RFQs to consider and requires reasoning about multiple possible scheduling days. Increasing the granularity of bid prices also increases the number of variables for the ILP to consider. Empirically, given the 15 seconds allowed by TAC SCM rules, our ILP in CPLEX 10 on a dual-core AMD System with 2G of memory fails to find solutions for a 5-day scheduling window with 160 RFQs to address each day evaluating bids at a 5% granularity.

2.3 A Marginal Profit Approach

In a competitive marketplace with identical products, the only opportunity for an seller to increase market share is to reduce price. If a single unit of a product sells for $10, the marginal profit for selling that product, relative to no sale at all, is $10. Because of the inverse relationship between price and demand, a lower price is required to sell additional units. If two units sell for $9 each, this yields $18 in total revenue and a marginal revenue of $8 relative to the sale of a single unit at the higher price. Applied to the TAC SCM environment, this means an agent that wants to sell additional units of a particular SKU by increasing its market share needs to lower its offering price.

Given this pricing dynamic, the offer-acceptance model described previously will slope downward from high-priced, low market share bids to lower-priced, higher market share bids. We now describe an greedy algorithm that uses this market-structure assumption to solve the bidding and scheduling problems. Our algorithm has five major steps:

- Calculate demand by combining the current days and any future modeled RFQs into a measure of total demand per SKU.

- For each SKU, generate an ordered list of marginal price entries. Each list item contains i) a bid percentage ii) quantity of total demand corresponding to the bid percentage iii) the marginal profit per cycle implied by the price corresponding to this entry.

- Handle orders either by fulfilling from finished computer inventory or scheduling production if components are available.

- Fill factory capacity for as many days demand as were included in step 1 by incrementally scheduling the next best marginal profit from the lists constructed in step 2.

- Determine the bids for the current day's RFQs by applying the bid from the final scheduled incremenental quantity for each RFQ's SKU.

Step One: Demand Calculation

The first step in the algorithm is to determine the total number of computers that the algorithm will be considering bids for. In a single day implementation of the algorithm, this is only the actual current day's RFQs sent by the TAC SCM Server. In a mutliday version of the algorithm, the algorithm will include both the current day's real RFQs and any predicted future demand. Regardless, the step ends with a count of the total number of computers of each SKU avaiable for bidding. This list will be used to determine the quantities corresponding to various bid percentages selected in step four.

Step Two: Generating Bid-Percentage Lists

Using the offer acceptance model for each SKU, the algorithm exhaustively generates a list of bid percentages based on a pre-specifed bid increment typically 1% or 5%. These percentages are then supplemented with the corresponding quantity associated with the bid percentage, which is calculated by multipling the bid percentage by the quantity calculated in step 1. Finally, each percentage is associated with a bid price. The algorithm combines the price, quantity, and cycles required for the current SKU to

calculate the marginal profit.

Step Three: Handle Orders

Before dealing with RFQs, the algorithm handles outstanding orders. Any orders that can be filled from inventory are immediately delivered. Orders that cannot be fulfilled from inventory, but for which we have components on hand, are scheduled in the factory ahead of any other production work.

Step Four: Fill Factory Capacity

The algorithm begins with a nearly unscheduled factory. Only those cycles used by orders not filled from inventory are assigned. The algorithm searches across all 16 SKUs for the most profitable SKU to produce. The bid percentage list for that SKU is incremented and the corresponding number of cycles are deducted from the available factory capacity. The algorithm repeats looking across all SKUs for the next most profitable marginal quantity to schedule. Eventually, the entire factory capacity is filled or we reach a point where increasing the quantity of any SKU future results in negative marginal profit. At this point, step four terminates and we have a factory schedule. If we are running a single day version of the algorithm the scheduled quantities are themselves the factory schedule. If we are running a multi-day version of the algorithm, we schedule the day's 2000 factory cycles in the same proportions as they larger capacity used in this step.

Step Five: Assign bids

Finally, to assign bids to the current day's RFQs, the algorithm iterates over each RFQ looking up the current bid percentage the tables above for the corresponding SKU.

2.4 On the theoretical optimality of the Marginal Profit Approach

The TAC SCM scheduling problem can be viewed as a knapsack problem, where the value of an order or RFQ is the equivalent of the value of the item to be placed in the knapsack, the number of cycles to produce an order is equivalent to weight, and the total factory capacity is equivalent to the fixed capacity of the knapsack. If we assume all orders are of uniform and identical size of one cycle, the scheduling problem is a *continuous knapsack problem.* In this case, the marginal profit scheduling algorithm produces an optimal solution to this problem. At each step, the algorithm selects the most profitable cycle to add to the factory continuing until the factory is full. The result is an optimal allocation of factory cycles.

2.5 On the actual non-optimality of the Marginal Profit Approach

In fact, the computers in TAC SCM vary in size from four to seven cycles. This means the scheduling problem is a knapsack problem, but is not continuous. In particular, it

is possible to select a most profitable computer that will leave the algorithm with an number of unschedulable cycles. This means a greedy algorithm cannot always find an optimal solution to scenarios which the ILP can handle.

Furthermore, although we would typically expect a smaller increment percentage to more find a more profitable bidding solution, this ordering issue can cause an instance of the marginal pricing algorithm running with 5% step size to lose to an algorithm with 1% step size. Consider for example a simplified version of TAC SCM with only 2 products and a 120 cycle factory capaciy. Under this scenarios we have SKU 1 requiring four cycles and SKU 2 requiring five cycles to produce. Assume that prices for SKU 1 start around 1800 and run to 950, prices for SKU 2 start lower and run to approximately the same lower value.
After 100 cycles, both are bidding entirely on SKU 1.

Table 1: 1% increment.

SKU	Bid %	Cycles Allocated
1	25%	100
2	0%	0

Table 2: 5% increment.

SKU	Bid %	Cycles Allocated
SKU 1 Bid:	25%	100
SKU 2 Bid:	0%	100

If after bidding for 25% of the market, the first SKU is still the most attractive product on a marginal revenue basis, the 5% step algorithm will bid an addtional 5% on SKU 1 and complete it's schedule.

Table 3: 5% increment 120 Cycles allocated.

SKU	Bid %	Cycles Allocated
1	30%	120
2	0%	0

Similarly the 1% algorithm will bid for an additional 1% of SKU 1. Further imagine that after allocating 28% to SKU 1, the marginal price for additional production of SKU 1 is below the revenue that can be achieved for delivering the first unit of SKU 2. The 1% algorithm will now switch from SKU 1 to SKU 2, leaving it bid as shown in table 4.

At this point the 1% algorithm has finished and although it allocated the most profitable computer at every stage, it is left with a non-optimal total solution. Furthermore

Table 4: 1% increment 117 Cycles allocated.

SKU	Bid %	Cycles Allocated
1	28%	112
2	1%	5

it has lost, on a revenue basis, to the 5% algorithm which was able to fill the entire 120 cycles.

3 Simulation Results

We now show a comparision of the ILP approach using 1% and 5% price discretization to the marginal revenue approach under 1% and 5% demand increment. The simulations are conducted in a mock server environment with infinite free component inventory, known offer-acceptance probabilities, and known future customer demand. The agent in the simulation participates alone so there is no competition for customer orders from other agents.

3.1 Two Day Games

We first consider the relative performance of the ILP bidder and the marginal bidder in a two-day game with totally uniform customer demand and no prediction. Each agent recieves 160 RFQs per day, 10 for each SKU, each of quantity ten. The offer acceptance curve for all products is identical. Only the current day's RFQs are considered when solving the bidding and scheduling problems. The game lasts only two days. The test agent bids and builds based on the first day's RFQs and delivers on day two.

Because the offer acceptance curves are identical, the marginal bidder bids most aggressively on SKU 1 and SKU 9 which require only four cycles to build. It doesn't bid at all on SKU 8 or SKU 16 which require seven cycles to build.

Table 5: 1% Marginal Bidder.

SKU	Avg Bid	SKU	Avg Bid
1	0.5600	2	0.3500
3	0.3500	4	0.1400
5	0.3500	6	0.1400
7	0.1400	8	0.0000
9	0.5500	10	0.3500
11	0.3500	12	0.1400
13	0.3500	14	0.1400
15	0.1400	16	0.0000

The 5% marginal bidder arrives at a similar bid distribution and an identical average overall bid (0.2895), although it is constrained to bids which are multiples of 5% of available demand.

Table 6: 5% Marginal Bidder.

SKU	Avg Bid	SKU	Avg Bid
1	0.5500	2	0.3500
3	0.3500	4	0.1500
5	0.3500	6	0.1500
7	0.1500	8	0.0000
9	0.5500	10	0.3500
11	0.3500	12	0.1500
13	0.3500	14	0.1500
15	0.1000	16	0.0000

The 1% and 5% ILPs achieve similar bid distributions. Note that the 5% ILP is not restricted to bidding the same price for all the RFQs of a particular SKU, so it is able to bid non-multiples of 5%.

Table 7: 1% ILP Bidder.

SKU	Avg Bid	SKU	Avg Bid
1	0.5500	2	0.3500
3	0.3500	4	0.1400
5	0.3500	6	0.1400
7	0.1400	8	0.0000
9	0.5500	10	0.3500
11	0.3500	12	0.1400
13	0.3500	14	0.1400
15	0.1500	16	0.0000

In table 9 we present revenues for 10 trials of the 2-day simulations.

3.2 High Low Demand

We now compare both single and multiday versions of the algorithm under varying demand scenarios. The multiday version of the algorithm has a view of future demand so it can bid more selectively than the single day algorithm. For example, the multiday algorithm can sometimes use factory capacity for orders that will be bid on and delivered in future days.
In this test we consider a stylized demand scenario where RFQs arrive only on even days. The multiday version of the algorithm sees the entire game's demand and can

Table 8: 5% ILP Bidder.

SKU	Avg Bid	SKU	Avg Bid
1	0.5500	2	0.3500
3	0.3500	4	0.1500
5	0.3500	6	0.1400
7	0.1400	8	0.0000
9	0.5500	10	0.3500
11	0.3500	12	0.1400
13	0.3500	14	0.1400
15	0.1400	16	0.0000

Table 9: 2-day revenue under uniform demand.

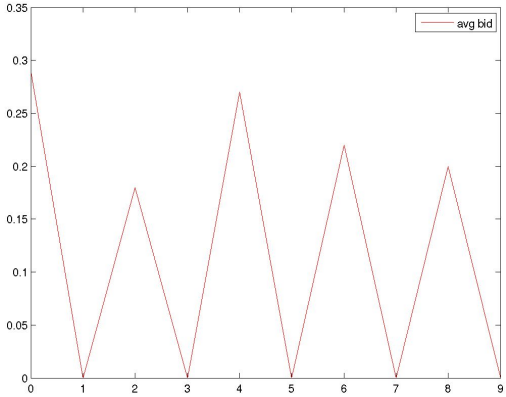
Agent	Revenue	Standard Deviation
ILP 1%	547,568.00	32,048.66
ILP 5%	549,840.00	35,138.94
Marginal 1%	547,160.00	32,059.22
Marginal 5%	545,440.00	32,419.03

adjust accordingly. It raises its bid percentages on the even days such that it generates enough orders to keep the factory busy during the odd days. Table 11 shows the revenue improvement under this high-low demand scenerio for both multiday bidders. The single day algorithms use an average bid of 12.37% and 13.22% where as the mutliday algorithm bids 14.23% and 15.09%. These higher bids are, in turn, reflected in high factory utilization. The single day algorithms use 1,436 and 1,274 cycles per day for 1% and 5% step sizes. The single day standard deviations are over 650 cycles reflecting the fact that the factory is nearly idle on days when their is no new customer demand. The mutliday algorithms use 1,979 and 1,978 cycles with std deviation of less than 15 cycles.

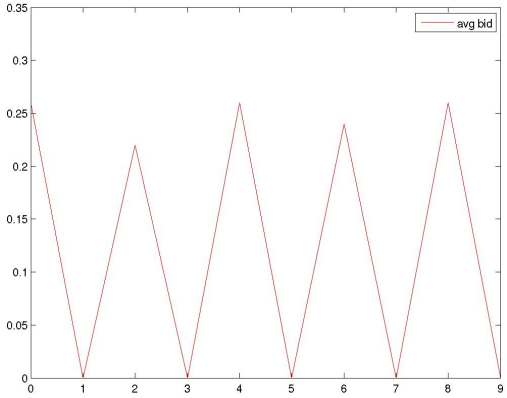
Table 10: 10-day revenue under high-low demand.

Agent	Revenue	Standard Deviation
Marginal 1%	5,021,448.00	162,182.59
Marginal 5%	4,462,260.00	190,423.33
Marginal 1% Multiday	6,958,755.20	54,227.53
Marginal 5% Multiday	6,973,236.00	55,545.34

These charts depict the average bid value for each day. The scale on both charts is identical. The second chart depicts the mutliday algorithm bidding for slightly more



of the market on even days.



3.3 Increasing Demand

A more realistic demand scenario is a gentle change in demand. Below we consider the revenue performance of the single and multiday algorthims fro a variety of steadily increasing demand profiles. We describe each scenario in terms of the number of RFQs available to the agent on the first day and the number of additional RFQs that arrive on each following day. Revenue advantages still accrue to the agents with the ability to lookahead to future demand, but not as dramatically as in the high-low tests.

4 Conclusions and Future Work

We haved described a marginal revenue based method for bidding on customer demand in the TAC SCM environment. The algorithm's revenue performance is competitive with an integer linear programming approach, but is significantly faster. Furthermore, as we add additional days of prediction to the algorithm, the ILPs performance rapidly

Table 11: 10-day revenue under increasing demand.

First Day RFQs	Daily Increase	Agent	Revenue	Standard Deviation
100	10	Marginal 1%	6,218,825.60	59,579.99
100	10	Marginal 5%	6,186,788.00	62,014.56
100	10	Marginal 1% Multiday	6,365,508.00	62,990.82
100	10	Marginal 5% Mutliday	6,273,548.00	73,256.09
100	40	Marginal 1%	6,715,347.20	124,244.02
100	40	Marginal 5%	6,709,392.00	127,109.77
100	40	Marginal 1% Multiday	6,702,052.80	225,101.84
100	40	Marginal 5% Multiday	6,609,364.00	241,709.62
170	10	Marginal 1%	6,549,108.00	103,730.76
170	10	Marginal 5%	6,556,680.00	65,731.30
170	10	Marginal 1% Multiday	6,780,241.60	113,083.23
170	10	Marginal 5% Multiday	6,713,612.00	122,418.02
170	40	Marginal 1%	6,976,202.40	85,918.75
170	40	Marginal 5%	6,971,820.00	81,588.17
170	40	Marginal 1% Multiday	7,108,596.80	134,681.71
170	40	Marginal 5% Multiday	7,008,548.00	139,667.26

degrades. Under our formulation, each additional day of RFQs increases the number of variables in the ILP. The marginal revenue algorithm is able to profitably incorporate information about future demand.

It remains to be seen if the algorithm can be extended to the procurement problem.