Botticelli: A Supply Chain Management Agent Designed to Optimize under Uncertainty

MICHAEL BENISCH, AMY GREENWALD, IOANNA GRYPARI, ROGER LEDERMAN, VICTOR NARODITSKIY, and MICHAEL TSCHANTZ
Brown University

The paper describes the design of the agent BOTTICELLI, a finalist in the 2003 Trading Agent Competition in Supply Chain Management (TAC SCM). In TAC SCM, a simulated computer manufacturing scenario, BOTTICELLI competes with other agents to win customer orders and negotiates with suppliers to procure the components necessary to complete its orders. We formalize subproblems that dictate BOTTICELLI's behavior. Stochastic programming approaches to bidding and scheduling are developed in attempt to solve these problems optimally. In addition, we describe greedy methods that yield useful approximations. Test results compare the performance and computational efficiency of these two techniques.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Intelligent agents

Additional Key Words and Phrases: Bidding agents, Trading agents, Supply chain management

1. INTRODUCTION

A supply chain is a network of autonomous entities, or agents, engaged in *procurement* of raw materials, *manufacturing*—converting raw materials into finished products—and *distribution* of finished products. The Trading Agent Competition in Supply Chain Management (TAC SCM) is a simulated computer manufacturing scenario in which software agents tackle complex problems in supply chain management. This paper describes the structure of Brown University's agent BOTTICELLI, a finalist in TAC SCM 2003.

TAC SCM agents face uncertainty about the future, but they must make their decisions before that uncertainty is resolved: e.g., agents procure raw materials and manufacture finished products before customer orders arrive. BOTTICELLI handles this uncertainty using stochastic programming techniques. We formulated a stochastic program, the solution of which is an optimal manufacturing and distribution schedule in TAC SCM. Moreover, we generalized this approach in search of optimal bidding policies that balance the tradeoff between maximizing profits, by placing high bids, and maximizing the likelihood of winning multiple orders, by placing low bids.

Authors' address: Box 1910, Providence, RI 02912.
This research was partially supported by NSF Career Grant #IIS-0133689.

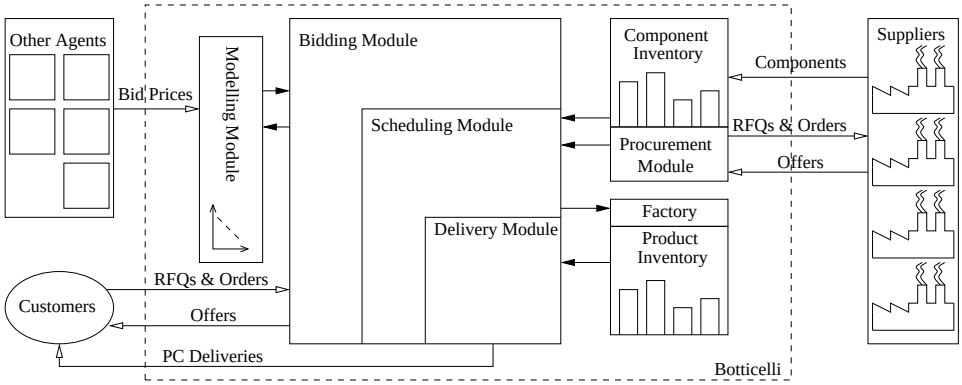

Fig. 1. **Botticelli: A Modular Design**

Inputs	Outputs
Product Pricing Model	Procurement Schedule: set of Supplier RFQs and Orders
Component Cost Model	Bidding Policy: map from Customer RFQs to Prices
Set of Supplier Offers	Production Schedule: map from Cycles to SKUs
Set of Customer RFQs	Delivery Schedule: map from SKUs to Customer Orders
Set of Customer Orders	
Procurement Schedule	
Component Inventory	
Product Inventory	

Fig. 2. **TAC SCM Decision Problem**

2. AGENT ARCHITECTURE

Each simulated TAC day represents a decision cycle for an agent, during which time the agents must solve four problems: procurement, bidding, production scheduling, and delivery scheduling. The *procurement* problem involves communicating with suppliers via RFQs, and selecting components to purchase given offers in response to these RFQs. The *bidding* problem is to decide how to assign offer prices to each customer RFQ. The *production scheduling* problem is to decide how many of each SKU to assemble each day. The *delivery scheduling* problem is to decide which orders to ship to which customers, using product inventory. The objective in all of these problems is to maximize expected profits: revenue less costs less penalties. A high-level description of the TAC SCM decision problem is presented in Figure 2.

An artifact in the design of TAC SCM 2003 (namely, negligible component prices on day 1), resulted in us placing little emphasis on *procurement*. Rather, we focused on the development of solutions to the *bidding*, *scheduling*, and *delivery* problems. High-level descriptions of the three problems are given in Figure 3. These three problems are highly interconnected. Indeed, an optimal solution to the production scheduling problem yields an optimal solution to the delivery scheduling problem, since revenues depend on which orders are successfully delivered to their respective customers. Moreover, an optimal solution to the bidding problem yields an optimal solution to both scheduling problems, since bidding decisions depend on

Delivery Scheduling
Inputs:
Production Schedule
Set of Customer Orders
Product Inventory
Output:
Delivery Schedule

Production Scheduling
Inputs:
Bidding Policy
Product Pricing Model
Set of Customer Orders
Procurement Schedule
Component Inventory
Product Inventory
Outputs:
Production Schedule
Delivery Schedule

Bidding
Inputs:
Product Pricing Model
Set of Customer RFQs
Set of Customer Orders
Procurement Schedule
Component Inventory
Product Inventory
Outputs:
Bidding Policy
Production Schedule
Delivery Schedule

Fig. 3. **TAC SCM Mini-Decision Problems**

manufacturing and distribution constraints: too few winning bids lead to missed revenue opportunities; too many winning bids lead to late penalties. BOTTICELLI's architecture was designed with these relationships in mind, and thus the bidding module envelops the scheduling module, which in turn envelops the delivery module as shown in Figure 1.

The flow of information through the agent is as follows: Each day the modeling module receives information about other agents' actions on the previous day as well as information about the offers the bidding module submitted and the orders that resulted from those offers. The modeling module uses this information to update its models and passes an updated model to the bidding module. The bidding module uses the new model to produce an offer for each of the day's RFQs. The offer prices are determined with the aid of the scheduling module. When invoked, the scheduling module learns from the procurement module the quantity of each component that is expected to be in inventory on any particular day. It then determines how to allocate machine cycles to make products for existing orders and likely future orders. The scheduling module relies on the delivery module to determine how to allocate product inventory to existing orders and likely future orders. After the bidding, scheduling, and delivery modules finalize their decisions, the procurement module sends to suppliers RFQs for any necessary additional components and orders based on the previous day's offers.

3. BIDDING: A STOCHASTIC PROGRAMMING APPROACH

We now define the delivery scheduling, production scheduling, and bidding problems. The delivery scheduling problem can be solved optimally in TAC SCM using an integer linear programming solver. The production scheduling problem can be formulated as a stochastic program, and an optimal solution can be approximated using sampling techniques such as sample average approximation [Kleywegt et al. 2001] (see [Benisch et al. 2004]). Similarly, the bidding problem can be formulated as a stochastic program, but the aforementioned approximation technique does not produce useful results in a reasonable amount of time because of the vast number of bidding policies. Instead, we propose a coarser approximation based on the expected value method [Birge and Louveaux 1997].

In order to describe our proposed solution to the bidding problem, we rely on a solution to the production scheduling problem. Similarly, our proposed solution to the production scheduling problem relies on a solution to the delivery scheduling problem. Thus, we describe these subproblems before describing the bidding problem and our approximate solution.

3.1 Delivery Scheduling

The delivery scheduling problem is one of allocating SKUs in product inventory to customer orders, given a production schedule (see Figure 3, LHS). The objective is: maximize revenue and minimize penalties. The important resource constraint concerns product inventory: The total quantity of SKU j associated with orders delivered by day t cannot exceed the total inventory of SKU j produced by day $t-1$ plus any initial inventory.

3.2 Expected Production Scheduling

In the production scheduling problem, the objective is to allocate cycles to SKUs not only to fill existing customer orders, but in addition to fill offers—customer RFQs equipped with bid prices—which may or may not become orders. We model this uncertainty using a pricing model that associates probabilities with offers: offers with low bid prices are assigned high probabilities, whereas offers with high bid prices are assigned low probabilities. These probabilities represent one of the key sources of uncertainty in TAC SCM. (See Figure 3, center.)

To handle this uncertainty, this problem can be formulated as a stochastic program (see [Benisch et al. 2004]). In solving this stochastic program, we show that the method of sample average approximation (SAA) [Kleywegt et al. 2001] outperforms the expected value method [Birge and Louveaux 1997] on this problem. Nonetheless, we rely on the expected value method in our implementation of Botticelli because it easily generalizes to bidding, whereas SAA does not.

The objective in production scheduling is to maximize profits from orders and *expected* profits from offers while minimizing penalties. The constraints are those of simple scheduling, but the product inventory resource constraint is updated to handle offers: The total quantity of SKU j associated with orders or *expected* offers delivered by day t cannot exceed the total inventory of SKU j produced by day $t-1$ plus any initial inventory.

3.3 Bidding

The objective in the bidding problem is to find an optimal bidding policy. We solve this problem by extending the solution to the production scheduling problem based on the expected value method. In production scheduling, all RFQs are equipped with bid prices, which are constants. In the bidding problem, the prices at which to offer to fill RFQs are variables. Once prices become variables rather than constants, the objective function is no longer linear. (In fact, in our formulation, it is not even quadratic.) Thus, in our implementation we discretize prices to recover a linear formulation. Details are described in a longer version of this paper.

4. BIDDING: A HILL-CLIMBING APPROACH

In the preliminary rounds, BOTTICELLI relied on a hill-climbing bidder, which successively adjusts bid prices according to the results of a scheduler. At a high-level, the bidder is initialized with some set of bid prices; given these prices, an approximately optimal production and delivery schedule is found; based on the results of the scheduler, bid prices are tweaked. The goal of this hill-climbing algorithm is to fill our production schedule, which we assume to be positively correlated with maximizing expected profits. A similar bidding solution is presented in [Pardoe and Stone 2004].

In a preprocessing step, we schedule only orders, no offers. As long as all orders can be scheduled for delivery, we proceed with the hill-climbing bidder.

It is crucial to our approach that the scheduler make use of the probabilities of winning each offer: the scheduler must schedule offers based on *expected quantities*.

We initialize bids to prices at which, according to our model, we will win every RFQ with certainty. At these initial prices, if the scheduler cannot fit every order and RFQ into the schedule, those RFQs which are not deemed profitable enough to include in the schedule at their current prices form a natural set of RFQs for which to raise prices. Indeed, we increase the prices of these RFQs, thereby decreasing their winning probabilities. In the next iteration, the scheduler, which schedules according to expected quantities, may be able to schedule these RFQs for production. Prices are increased (i.e., probabilities are decreased) until all RFQs can be scheduled. This process is guaranteed to converge, since the winning probability of RFQs above their reserve prices is zero, yielding a corresponding expected quantity of zero.

4.1 Scheduling: A Greedy Approach

Our greedy scheduler is passed both orders and offers, which it sorts as follows:

—Orders are placed before offers, since offers might not be won.
  —Orders are sorted by ascending due date, then by descending penalty.
  —Offers are sorted by descending profit per cycle (p_ι/c_j, where $j = f_\iota$), then by ascending due date, and lastly by descending penalty.

Note that offers are not sorted by probability. We experimented with this ordering, but profitability proved to be more important than probability.

Let o be the current order or offer and let j be o's SKU. The greedy scheduler addresses the orders and offers in sorted order as follows:

(1) Schedule backwards from o's due date. That is, start by scheduling as much as possible of SKU j on the day o is due. If more needs to be scheduled, then schedule as much as possible on each successively earlier day until either no more is needed or the current day is reached.
(2) If more of SKU j still needs to be produced, allocate as much as possible from product inventory.
(3) If still more of SKU j is needed, schedule forwards from o's due date until either all of order o is scheduled or the cancellation date is reached.
(4) If the cancellation date is reached, then cancel all scheduled production of SKU j for o.

Note that if o's due date is the current day, then there is no time to produce any more of SKU j. In this case, the greedy scheduler begins at step 2.

5. EXPERIMENTAL RESULTS

The mathematical programming techniques we employ rely on some model of our agent's environment. Rather than model each competing agent's strategic behavior individually, we collapse all agents' behaviors into one model, and optimize with respect to this model. In essence, we use decision-theoretic optimization techniques to approximate solutions to game-theoretic problems.

Not only is the TAC SCM environment uncertain, it is also dynamic. Thus, dynamic optimization models and techniques might be applicable (e.g., MDPs), but to optimize with respect to all possible futures is clearly intractable. Instead, we rely on an heuristic we call the *triangle method*, by which we save production cycles on future days for future RFQs, particularly if prices are predicted to increase.

Before describing our experimental setup and results, we describe our modeling assumptions and the triangle heuristic, on which the former depend.

5.1 Modeling

The modeling module predicts the relationship between the bid price of an offer and the probability of winning that offer. There are several sources of information available for modeling this relationship. In our implementation, we utilize two: the first is a report provided by the server each day with the maximum and minimum closing prices for each SKU on the previous day; the second is BOTTICELLI's past offer prices and the orders that resulted. Our modeling module is concerned only with price and probability relationships for each SKU, rather than for each RFQ, since maximum and minimum prices are SKU-specific.

For each SKU, the modeler plots the minimum and maximum prices from the previous day at probabilities 1 and 0, respectively. Intuitively, low prices are likely to be winning prices, while high prices are likely to be losing prices. In addition, for each of the previous d days, BOTTICELLI's average offer prices are plotted against the ratio of the number of offers won to the number of offers issued. In total, our modeling module is provided with $d + 2$ points, which it fits using a least-squares linear regression. This linear *cdf* (price vs. probability graph) is adopted as the model that is input to the bidding module. (See Figure 4.)

By experimentation, we found the value of 5 to be a good choice for d. This value allowed BOTTICELLI to be responsive enough to the changes in price that often accompanied another agent receiving a shipment of supplies, but prevented any drastic overreactions. We experimented with using additional information to create more stable models, such as providing weights for points based on the number of offers they represented, and maintaining the average of the d previous days' minimum and maximum prices. These methods, however, did not respond well to price jumps that were typical of the 2003 TAC SCM competition.

5.2 The Triangle Heuristic

BOTTICELLI's scheduling module relies on the following heuristic: in scheduling for multiple days of production, do not use all cycles on all days, but rather save production cycles on future days for future RFQs as depicted in Figure 5.

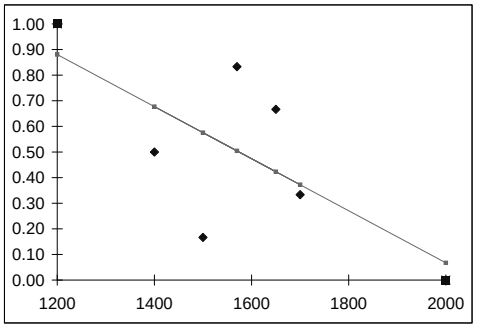

Fig. 4. Price vs. Probability for a SKU. Diamonds are data points from offers sent during the past d days. Squares are data points from the previous day's minimum and maximum prices.

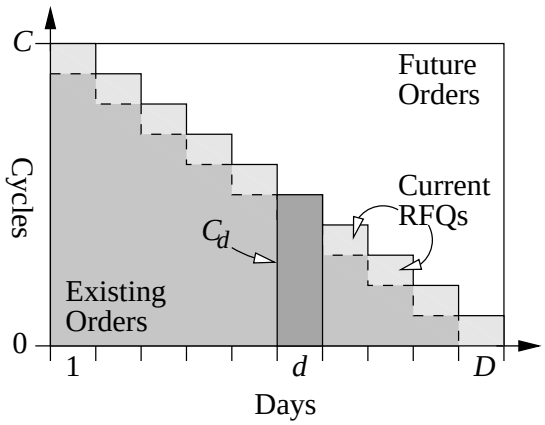

Fig. 5. On day d, only $C_d = \frac{C((D-d)+1)}{D}$ cycles are made available to the scheduler. Cycles outside the triangle are reserved for future orders. D is number of days of production in the schedule. C is the daily production capacity.

This heuristic is based on two assumptions. First, higher revenues can be earned by winning the same quantity of RFQs over multiple days, rather than winning a large quantity of RFQs on one day, since, according to our model, an agent can only win a large quantity on one day by bidding a very low price. Second, the "character" RFQs of tomorrow will not differ significantly from the RFQs of today, since all RFQs are drawn from a uniform distribution. Thus, one can assume that future RFQs will not be significantly better or worse than the today's RFQs in terms of quantity, due date, etc.

If, however, a change in the *number* of RFQs is predicted, BOTTICELLI saves more (fewer) cycles if the number of RFQs is predicted to increase (decrease), since prices tend to increase (decrease) accordingly.

Parameter	Range
Price	[$1600, $2300]
Quantity	[1, 20]
SKU	[1, 16]
Penalty	[5%, 15%] of Price

Table I. Uniform Distribution Ranges

	Profits	Deliveries	$\overline{\text{Price}}$	Penalty
HG	$7,781,100	6,847	$1,193	$505,610
HE	$8,019,600	7,286	$1,095	$285,950
EB	$9,600,900	7,860	$1,222	$113,660

Table II. Experimental Results: Profits, Deliveries, Average Prices, Penalties

5.3 Experimental Setup

We designed experiments to compare the performance of three bidding algorithms, one based on the stochastic program, one hill-climbing bidder, and one blend of the two. To isolate the effects of these algorithms, we relied on models that could perfectly predict the likelihood of winning any RFQ at any price.

Our experiments consisted of 20 day trials, which proceeded as follows: On each day, the algorithms received a randomly generated set of RFQs drawn from a distribution similar to that of the TAC SCM game specification. Specifically, 300 RFQs were generated at random, with parameters uniformly distributed in the ranges shown in Table I. Given these RFQs, the algorithms produced a bidding policy as well as production and delivery schedules for $D = 10$ days.[1] Based on its bid prices and the corresponding probabilities, an algorithm won orders for some of the RFQs. The algorithms were then responsible for producing and delivering the products for these RFQs before their due dates or they were penalized according to the rate specified in the RFQ. The tests continued in this fashion for 20 days; this number was long enough to allow the algorithms to distinguish themselves, but short enough to allow several hundred iterations.

In order to mitigate any start effects in our experiments, the algorithms were initialized with the same set of 150 customer orders (thus, the first day looked like all other days). We made the simplifying assumption that all algorithms had an infinite component inventory, which, as alluded to earlier, is an artifact of the TAC SCM game design in 2003.

5.4 Experimental Results

The algorithms included in our experiments were the hill-climbing bidder with a greedy production scheduler (HG), the hill-climbing bidder with an expected production scheduler (HE), and the expected bidder (EB), which used its own schedule for production. Both of the hill-climbing bidders utilized a greedy scheduler to evaluate candidate bidding policies, as such policies needed to be evaluated hundreds of times. (The greedy scheduler completed in .01 seconds, on average, whereas the

[1] In TAC SCM, 10 days are sufficiently many to produce all current orders and RFQs on time.

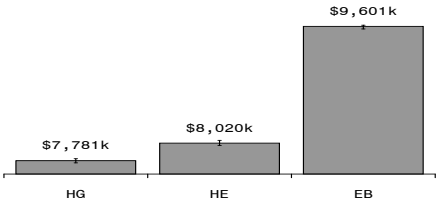

Fig. 6. Mean Profit–95% Confidence Intervals

expected production scheduler completed in 1 second.) However, we allowed one of the hill-climbing bidders to utilize an expected scheduler for production scheduling only. Our hypothesis was that the expected bidder with built in scheduling and delivery modules would out perform all of the others, as it would be capable of performing a more global optimization while solving the bidding problem.

Relevant statistics of the 500 trials are given in Table II. The mean profits of each algorithm over 20 days with 95% confidence intervals are shown in Figure 6. These results validated our hypothesis. The expected bidder outperformed both instances of the the hill-climbing bidders in every category in Table II. The 95% confidence intervals shown in Figure 6 reveal that the difference in profits is statistically significant. The addition of the expected scheduling algorithm to the hill-climbing bidder helped it to achieve fewer penalties by improving the production scheduling solutions; however, the lack of a global bidding strategy still crippled its abilities. It seems that the expected bidder produced results that were close to optimal, since its total penalty was relatively small and it managed to utilize its factory at nearly full capacity each day without wasting many finished products.

6. CONCLUSION

Following [Kiekintveld et al. 2004], we identify three key issues in supply chain management that are modeled in TAC SCM: (i) *uncertainty* about the future; (ii) *strategic behavior* among the entities; and (iii) *dynamism*—the temporal nature of the chain. Botticelli, a finalist in TAC SCM 2003, focused on the first of these three issues. Indeed Botticelli's expected bidder proved to be an effective technique for addressing the uncertainty in the TAC SCM market economy. In future work, we plan to validate (or invalidate) our modeling technique and the triangle heuristic, which were designed to address the strategic and dynamic aspects of TAC SCM, respectively.

REFERENCES

Benisch, M., Greenwald, A., Naroditskiy, V., and Tschantz, M. 2004. A stochastic programming approach to TAC SCM. In *ACM Conference on Electronic Commerce*. To Appear.

Birge, J. and Louveaux, F. 1997. *Introduction to Stochastic Programming*. Springer, New York.

Kiekintveld, C., Wellman, M., Singh, S., Estelle, J., beychik, Y. V., Soni, V., and Rudary, M. 2004. Distributed feedback control for decision making on supply chains. In *Fourteenth International Conference on Automated Planning and S cheduling*. To Appear.

Kleywegt, A., Shapiro, A., and Homen-De-Mello, T. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization 12*, 479–502.

Pardoe, D. and Stone, P. 2004. TacTex-03: A supply chain management agent. *SIGecom Exchanges 4,* 3 (February), In this issue.