Value-Driven Procurement in the TAC Supply Chain Game

CHRISTOPHER KIEKINTVELD, MICHAEL P. WELLMAN, SATINDER SINGH, and VISHAL SONI
University of Michigan

The TAC supply-chain game presents automated trading agents with challenging decision problems, including procurement of supplies across multiple periods using multiattribute negotiations. The procurement process involves substantial uncertainty and competition among multiple agents. Our agent, Deep Maize, generates requests for components based on deviations from a reference inventory trajectory defined by estimated market conditions. It then selects among supplier offers by optimizing a value function over potential inventory profiles. This approach offered strategic flexibility and achieved competitive performance in the TAC-03 tournament.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents and multiagent systems*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*economics*

General Terms: Algorithms, Economics

Additional Key Words and Phrases: Trading Agents, E-Commerce, Supply Chains

1. INTRODUCTION

The Trading Agent Competition Supply Chain Management scenario (TAC/SCM) was designed to pose a complex multi-tiered, multi-period problem for automated trading agents in a plausible supply chain game [Sadeh et al. 2003]. The TAC/SCM environment is challenging for many reasons. One is that agents are faced with substantial *uncertainty*: about the local state of other agents in the game, as well as the underlying demand and supply processes. The environment is also *strategic*, comprising six profit-maximizing producer agents. Agents must negotiate *multiattribute* deals with suppliers and customers, so they must be able to reason about the relative values of those attributes. Finally, the SCM game forces agents to make decisions over *multiple stages*, and on different time scales. Agents must make decisions (e.g., component procurement) before all relevant uncertainty is resolved (e.g., customer demand, future component prices).

We designed the University of Michigan's agent, Deep Maize, to participate in the

Author's address: Computer Science and Engineering Division, University of Michigan, Ann Arbor, MI 48109.
Email: {ckiekint,wellman,baveja,soniv}@umich.edu

2003 TAC/SCM tournament. Deep Maize employs *distributed feedback control* to coordinate its various modules and operate robustly despite dynamic uncertainty. Its overall feedback-control approach and the specific methods by which Deep Maize sets its *reference inventory trajectory* are defined elsewhere [Kiekintveld et al. 2004]. Here we focus on how the agent manages its procurement actions given the reference trajectory, by optimizing a *value function* over potential inventory profiles.

Value function representations have been used to make decisions in other manufacturing contexts. For example, Schneider et al. [1998] formulate a production scheduling problem as a Markov Decision Process (MDP) and use reinforcement learning methods to approximate the value function of this MDP. The decision problem we address does not fit the assumptions of the MDP model and must be made in the context of a larger agent making many distributed but interrelated decisions. Our approach uses a heuristic value function representation that incorporates values from several different sources in support of a single decision.

2. PROCUREMENT DECISIONS IN TAC/SCM

Each day SCM agents must make several decisions, two of which comprise the *procurement policy*: (1) What RFQs to issue to component suppliers and (2) Of the offers received from suppliers, which to accept. There are eight suppliers, each producing two component types from four categories of components: CPU, motherboard, memory, and hard disk. The four CPU types are each sold by a single supplier; the two types of components in all other categories are each sold by two suppliers. Each supplier has a *nominal capacity* to produce 500 per day of each component type it supplies. Actual production varies about this capacity in a random walk. To acquire components, an agent sends RFQs to a supplier (up to ten per supplier per day, in priority order), each specifying a desired quantity and due date. The supplier responds the next day with offers specifying quantity, due date, and price, and reserves sufficient capacity to meet these commitments. If projected capacity is insufficient to meet the requested quantity and date, the supplier instead offers a partial quantity at the requested date and/or the full quantity at a later date. Suppliers assume nominal capacity when projecting future availability. To generate responses, suppliers execute the following until all RFQs are exhausted: (1) randomly choose an agent, (2) take the highest-priority RFQ remaining on its list, (3) generate an offer, if possible. Agents must accept or decline each offer the day they receive it.

Suppliers set prices based on an analysis of available capacity. The TAC/SCM component catalog [Arunachalam et al. 2003] associates every component c with a *base price*, b_c. The correspondence between price and quantity for component supplies is defined by the suppliers' pricing formula. The price offered by a supplier at day d for an order to be delivered on day $d+i$ is

$$p_c(d+i) = b_c - 0.5 b_c \frac{\kappa_c(d+i)}{500i}, \tag{1}$$

where $\kappa_c(j)$ denotes the cumulative capacity for c the supplier projects to have available from the current day through day j. The denominator, $500i$, represents the nominal capacity controlled by the supplier over i days, not accounting for any capacity committed to existing orders.

3. DEEP MAIZE REFERENCE INVENTORY TRAJECTORY

The *reference inventory trajectory* is the sum of three sources of component requirements: (1) outstanding customer orders, (2) expected future component utilization, and (3) baseline buffer levels. First, *outstanding customer orders* entail a known requirement for specific components in time to produce the orders.

Second, we derive the time series of *expected future component utilization*, based on projections of future customer demand for PCs and market equilibrium calculations. The projection of customer demand uses a dynamic Bayesian network model to estimate the underlying demand state and project these values forward using the specified system dynamics. Market equilibrium is derived for each day by calculating the prices (based on Eq. (1)) at which the supply would equal the projected customer demand. Deep Maize assumes that it will garner, on average, new orders covering 1/6 of the equilibrium quantity Q_d for day d, evenly distributed across the possible PC types. To account for unpredictability in the demand trends, we set a somewhat more conservative reference, based on the demand quantity Q' satisfying $\Pr(Q_d \geq Q') = 0.63$ (0.63 was chosen somewhat arbitrarily). Over the range where existing and prospective customer orders overlap, we phase in the expected utilization proportionately.

The final contributor to our inventory reference is a *baseline buffer level*, maintained to mitigate short-term noise in procurement and sales activity and allow the agent to act more opportunistically. For the tournament, Deep Maize set the baseline level at 6.0 times the current expected daily consumption (also somewhat arbitrary). This level is scaled gradually to zero at the end of the game, at which point inventories become worthless. The sum of these requirements represents the trajectory of gross inventory requirements. To determine the net reference trajectory, we subtract the current inventory of components (including those contained in finished PCs) plus anticipated deliveries of components already purchased from the gross requirements.

4. DEEP MAIZE PROCUREMENT POLICY

Each day agents issue RFQs to suppliers and accept or reject supplier offers received in response to the previous day's RFQs. Deep Maize applies the same RFQ-generation and offer-acceptance policies to every day of the game, with one significant exception: the very beginning of the game (*day 0*). The supplier pricing formula provides a strong incentive to procure large quantities of components on this day, as prices are at their lowest and availability is as high as it will ever be.[1] As a result, in TAC-03 agents employed increasingly aggressive day-0 procurement policies, leading to a mutually destructive overcapacity of components for the aggregate system. We anticipated this effect and introduced our own *preemptive* day-0 strategy that neutralized this behavior somewhat and, in effect, reestablished a setting wherein agents had to procure supplies throughout the game. The details of our day-0 strategy and its effects are described in a separate account [Estelle et al. 2003]. For our present purpose, it suffices to note that we employed special RFQ generation and offer acceptance strategies at the very beginning of the game.

[1]Due to its distorting effects, this will be modified for the 2004 tournament.

4.1 RFQ Generation

Deep Maize generates RFQs to reduce the difference between the current and reference inventory trajectories. Three elements of the reference trajectory are considered in turn: outstanding customer orders, baseline buffer level, and expected future component utilization. This prioritization takes into account the immediacy of current orders and subsequent opportunities to procure components for future consumption. TAC/SCM limits agents to ten RFQs per day per supplier, and Deep Maize uses all these slots. It generates ten RFQs (split across two suppliers) for each non-CPU component type, and five for each CPU.

4.1.1 *RFQs for outstanding customer orders.* Figure 1 depicts the process of generating order-related RFQs. We compute the current inventory trajectory including components in assembled PCs as well as known future component arrivals. For each customer order that cannot be filled using current inventory, we generate an RFQ for the corresponding deficit quantity and due date. If more than 8 RFQs are generated, those with nearby due dates are merged to stay within this quota.

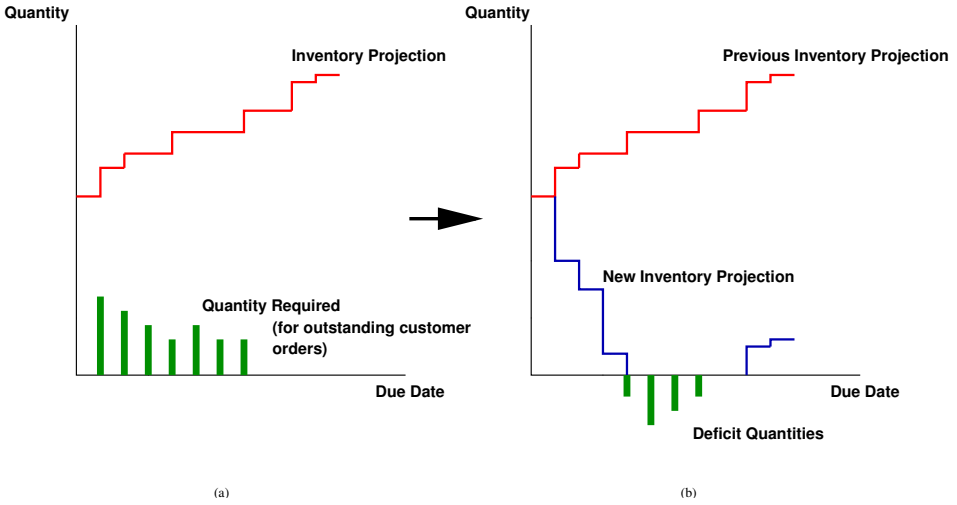

Fig. 1. Generating order-related RFQs for a particular component. (a) Quantities required and (b) Final component inventory projection. RFQs are created for the deficit quantities in (b).

4.1.2 *RFQs for baseline buffer level.* The current inventory trajectory is modified by removing PCs and components already committed to outstanding customer orders. An RFQ is generated for the first day this trajectory drops below the baseline buffer level to make up the difference. If this quantity is large, the RFQ is split in two, with half the quantity sent to each supplier for the component type.

4.1.3 *RFQs for expected component utilization.* Any remaining RFQ slots are used to request components addressing the long-term expected component utilization. The same current inventory trajectory used for generating baseline RFQs is used again, and the expected component utilization curve is subtracted from this

trajectory. A potential RFQ is generated for each day where this quantity is negative. A subset of these RFQs are selected to fill the available RFQ slots. The selection is probabilistically biased towards days when components are expected to be cheaper. To estimate available prices Deep Maize maintains an assessment of each supplier's available capacity profile, based on the offers seen and the supplier pricing function. Each offer yields information about the supplier's current capacity. For example, a partial completion offer means that a supplier has exactly the offered quantity available by the requested due date, and no more than the offered quantity available on any previous day. Calculating the implications of every offer yields an upper bound on capacity (and thus lower bound on price) for each day.

4.1.4 *Probe RFQs.* If RFQ slots are available after all reference inventory trajectory needs have been met, Deep Maize issues *probe* RFQs—for a single component on a random date—to garner additional information about supplier state.

4.2 Offer Acceptance

The offer acceptance mechanism selects which supplier offers to accept based on the reference inventory trajectory. Deep Maize makes its acceptance decisions separately for each component type. For each offer received, it may have three choices: reject (R), accept complete (AC), or accept partial (AP)—the third option is applicable only if the offer includes this option due to the supplier's inability to provide the full quantity by the requested date.

Given a set of n offers, let $o = \langle o_1 \cdots o_n \rangle$, $o_i \in \{R, AC, AP\}$, denote a decision vector. The agent's optimization problem is to identify

$$\arg \max_{o \in \{R,AC,AP\}^n} V(s, o) - \sum_i C_i(o_i), \tag{2}$$

where $V(s, o)$ is the value of the inventory trajectory starting from current state s plus the orders accepted according to o. The state comprises all information relevant to the reference inventory trajectory, including current inventory of components and PCs, anticipated component deliveries, and outstanding customer orders. $C_i(AC)$ (respectively, $C_i(AP)$) denotes the cost of accepting the complete (resp. partial) order i. For all i, $C_i(R) = 0$.

The three sources of component requirements contribute differentially to the value function. Components required to fill existing customer orders are valued at the entire price of the order plus the penalty charges avoided by meeting the order. The penalty amount specified in the customer RFQ is paid each day an order is late until it expires, so the penalty charges saved vary depending on when the order can be met. These values are quite high compared to the cost of an individual component, implying that (almost) any offer necessary to meet an existing customer order will be accepted, regardless of price. Including penalty charges allows the agent to reason about cases where an order can be met earlier by accepting one offer over another, reducing penalty payments. Components that address future expected consumption are valued at the expected equilibrium price for the projected day of consumption. This reflects an assumption that the market will be in equilibrium, and thus any components obtained for less than the equilibrium price will lead to profitable future production.

Components that fill baseline inventory are valued at the equilibrium price for the current day plus a *baseline premium*. The baseline premium is defined on a sliding scale, with higher premium values for the first components.[2] The premium values are intended to heuristically account for two factors: (1) the fact that components actually have value only when combined with other components and (2) the opportunity cost of having production constrained by available components. The baseline values are decayed by a constant multiplicative factor each day to represent a bias towards achieving the baseline inventory as soon as possible.

To value an inventory trajectory, we start by creating a sorted list of possible component values for each future day. Unit values calculated from the current reference trajectory according to the rules above are inserted into the list for the last day the need can be met. Each component is then valued in order of arrival. The algorithm looks forward from the day of arrival to find the maximum possible value that can be assigned to the component and removes this value from the corresponding list. If the value is from a later day than the component arrives, the value is replaced with the highest value from an earlier day. This is necessary to ensure that the maximal sum is always assigned. To see why this is so, consider a highly simplified example with single components arriving on days 0 and 2 and the following unit values for the next three days:

```
Original Values        Without Replacement     With Replacement
Day 0:  20, 5          Day 0: 20, 5            Day 0: 5
Day 1:  15, 10, 10     Day 1: 15, 10, 10       Day 1: 15, 10, 10
Day 2:  30, 15, 5      Day 2: 15, 5            Day 2: 20, 15, 5
```

That is, having one unit on day 0 is worth 20, and the second is worth an additional 5. The center and right columns reflect the values after assigning (and removing) one value, with and without replacement. Without replacement, the value 30 is assigned to the first component and 15 to the second. These values are not maximal since the first component could be assigned the value 20 and the second assigned the value 30 with both components still arriving in time. By replacing the value 30 with the value 20 when it is removed we can represent the possibility that a later arriving component could fill this need, freeing the first component to fill an earlier high-valued need.

When all components arriving on a given day have been valued, any remaining order-based or baseline values are propagated to the next day, accounting for penalties paid, order expirations, and decay of baseline values. Expected utilization values are never propagated. Once the entire inventory trajectory is processed, its total value is simply the sum of the values assigned to its components. The net value of the order-acceptance decision is this value minus the cost of accepted orders (Eq. 2).

Using this procedure to evaluate candidate choices, we set up a search problem for each component type. The search space is defined by the possible acceptance decisions. Since there are at most three possible decisions and $n \leq 10$ original RFQs, there are up to 3^{10} inventory trajectories to evaluate. This is too many given the limited time available (15 seconds) for each day's decisions, so we perform a local

[2]Values from 25–100% of the component base price were used during the tournament

Table I. Deep Maize tournament procurement by RFQ type. Prices are normalized to a percentage of the base price.

	Strategic	Orders	Baseline	Utilization	Probe
Percentage of RFQs	0.8 %	1.7 %	6.6 %	27.7 %	63.3 %
Acceptance Rate	32.3 %	10.3 %	24.7 %	53.0 %	18.4 %
Percentage of Quantity	51.2 %	0.7 %	14.0 %	33.1 %	9.4 %
Average Price	57.6	84.8	68.5	65.3	61.4

optimization using hill-climbing search. To the extent that an offer is worthwhile or not independent of which others are accepted, hill-climbing should quickly find a near-optimal solution.

The search starts from a node representing the state maximally accepting offers. At each node, the neighborhood of states with one decision changed is examined. If a higher-valued state is found, that state becomes the new current search node, and its neighborhood is expanded. When no higher-valued state is found in this neighborhood, the search is extended to a neighborhood including states with two decisions changed. Search terminates when the extended search fails to find a higher-valued state, or when time runs out for making a decision. To allow equal opportunity to find reasonable acceptance sets for all component types we run searches for all 10 types in parallel, evaluating one state each turn. Once all searches terminate, orders are sent for the highest-valued acceptance sets.

5. DISCUSSION

In this section we present some results from the TAC-03 tournament to show how Deep Maize's procurement strategy works in practice. Table I shows a breakdown of how the different types of RFQs Deep Maize sends contributed to its procurement during the semi-final and final rounds.[3] Procurement quantities were split almost evenly between strategic and steady-state behaviors. Order-based needs were the most expensive to fill, followed by baseline and future utilization needs. This is consistent with the relative values assigned to those needs during the offer acceptance process, and indicates that the premium values change the acceptance decisions in the intended way. Deep Maize purchased the bulk of its components using the strategic and future utilization mechanisms, avoiding the need to pay high premiums for components in all but rare instances.

We also compared the average prices paid and quantities procured by Deep Maize to the other five agents in the tournament finals. The results are broken down into three time periods: day 0, days 1–2, and the rest of the game. Day 0 is when most of the strategic interactions played out, but RedAgent and Deep Maize both had fallback strategies that procured substantial additional quantities immediately after day 0. Days 3–219 represent the days when our procurement module was using its steady-state policy. In Table II we see that all agents purchased a substantial fraction of their components on day 0. Whitebear purchased components only on day 0, but ended up purchasing far fewer total components than most other agents. All

[3]Includes all games played by Deep Maize except 1241, 1269, and 1429 when Deep Maize experienced network problems (a total of 28 games). We also note that Deep Maize used somewhat different day-0 procurement strategies in the three rounds. The pattern of results presented here holds when the rounds are considered individually instead of in aggregate.

Table II. Component quantities purchased during different game periods in the TAC-03 final round. Checkmarks indicate a statistically significant difference with Deep Maize ($p \leq 0.05$), while x indicates a less significant difference ($p \leq 0.10$). Agents are listed in the order they placed in during the tournament finals.

	Day 0	Days 1–2	Days 3–219	Ave. Total Purchased
RedAgent	40.9 % x	19.5 % ✓	39.7 %	254101 ✓
deepmaize	31.1 %	29.4 %	39.5 %	226423
TacTex	33.0 %	0.8 % ✓	66.3 % ✓	80396 ✓
Botticelli	41.2 % x	0.0 % ✓	58.8 % ✓	213340
PackaTAC	20.4 % ✓	3.8 % ✓	75.8 % ✓	117545 ✓
whitebear	100.0 % ✓	0.0 % ✓	0.0 % ✓	53571 ✓

Table III. Average normalized prices paid for components during the TAC-03 final round. Checkmarks indicate a statistically significant difference with Deep Maize ($p \leq 0.05$).

	Days 0–219	Days 1-2	Days 3–219
RedAgent	0.633	0.822 ✓	0.676 ✓
deepmaize	0.632	0.767	0.630
TacTex	0.641	0.656 ✓	0.722 ✓
Botticelli	0.575 ✓	-.-	0.630
PackaTAC	0.734 ✓	0.854 ✓	0.796 ✓
whitebear	0.500 ✓	-.-	-.-

other agents purchased throughout the game, with significant procurement activity from days 3–219.

Some information about the procurement strategies of the other agents after day 0 can be found in their published accounts. RedAgent [Keller and Duguay 2004] and PackaTAC [Dahlgren 2003] both used variations of a strategy based on maintaining a threshold inventory level with a reorder policy if inventory levels below the threshold were detected. Both used fairly simple offer acceptance strategies that did not reject offers on the basis of price. TacTex [Pardoe and Stone 2004] had a procurement strategy somewhat similar to Deep Maize's. It projected inventory needs for the next 50 days and issued RFQs to fill those needs for days with low expected prices. Offers were accepted if marginal values exceeded the cost of the order (independently from other offers). TacTex projects needs and calculates marginal values based on historical data, while Deep Maize uses projections of future customer demand and market analysis. Botticelli [Benisch et al. 2004] did not consider the procurement decision as part of their overall optimization approach because of the day-0 effects, and Whitebear did not purchase any components after day 0.

Looking at Table III we see that Deep Maize achieved significantly better prices than every agent except Botticelli from days 3–219. Botticelli and Deep Maize had almost identical performance. In the first column we see that the top three placing agents paid very similar average prices for components over the entire game. PackaTAC paid significantly higher prices, while Botticelli and Whitebear paid lower prices (in Whitebear's case because they only purchased on day 0). Interestingly, looking only at these procurement numbers Botticelli appears to have the most efficient overall procurement strategy. It purchased a similar total quantity of components as the top two agents, and purchased them for lower prices. That this agent placed fourth overall is testament to the fact that average prices do not tell

the full story; when components arrive and what the agent does with them also matter.

This analysis shows that Deep Maize had a competitive overall procurement policy. The steady-state part of this policy was based on value-driven procurement guided by a reference inventory trajectory. It was flexible enough to express many types of procurement requirements, and robust enough to be used in conjunction with several initial procurement strategies. The tournament behavior of the procurement module corresponded well to the intended behavior represented by the value function. This resulted in Deep Maize achieving average price performance equal to or better than all other finals agents when the steady-state policy was active. One caveat to consider is that the importance of day 0 procurement in TAC-03 may have caused many of the other agent developers to put less effort into developing policies for procurement during the rest of the game. Since changes will be made for TAC-04 to reduce the impact of day 0 procurement, we look forward to the 2004 competition as a better test for Deep Maize's steady-state procurement policy.

There are many possibilities for improvements to Deep Maize's procurement approach. For instance, some of the parameters defining the reference inventory trajectory (e.g. baseline level and premiums) were set somewhat arbitrarily and could undoubtedly be improved by additional tuning, analysis, or learning methods. Our estimates of supplier capacity could also be used more effectively to target suppliers with available capacity at low prices, potentially improving offer acceptance rates and component prices. These same estimates could also be used to improve our projections of future component consumption by identifying times when production will be impossible because key components are not available or are very expensive. We believe that these and other improvements will be interesting avenues for further research on value-driven procurement in future TAC/SCM competitions.

ACKNOWLEDGMENTS

We gratefully acknowledge the help of many people who made the TAC/SCM tournament possible including R. Arunachalam, J. Eriksson, N. Finne, S. Janson, and N. Sadeh. At the University of Michigan, Deep Maize was designed and implemented with the additional help of Joshua Estelle, Yevgeniy Vorobeychik, Matthew Rudary, Kevin O'Malley, Thede Loder, and Shih-Fen Cheng. This work was supported in part by NSF grant IIS-0205435.

REFERENCES

Arunachalam, R., Eriksson, J., Finne, N., Janson, S., and Sadeh, N. 2003. The TAC supply chain management game. Tech. rep., Swedish Institute of Computer Science. Draft Version 0.62.

Benisch, M., Greenwald, A., Naroditskiy, V., and Tschantz, M. 2004. A stochastic programming approach to scheduling in TAC SCM. In *Fifth ACM Conference on Electronic Commerce*. New York.

Dahlgren, E. 2003. PackaTAC: A conservative trading agent. M.S. thesis, Lund University.

Estelle, J., Vorobeychik, Y., Wellman, M. P., Singh, S., Kiekintveld, C., and Soni, V. 2003. Strategic interactions in a supply chain game. Tech. rep., University of Michigan.

Keller, P. W. and Duguay, F.-O. 2004. RedAgent - winner of TAC SCM 2003. *SIGecom Exchanges 4,* 3.

Kiekintveld, C., Wellman, M. P., Singh, S., Estelle, J., Vorobeychik, Y., Soni, V., and Rudary, M. 2004. Distributed feedback control for decision making on supply chains. In *Fourteenth International Conference on Automated Planning and Scheduling*. Whistler, BC.

Pardoe, D. and Stone, P. 2004. TacTex-03: A supply chain management agent. *SIGecom Exchanges 4,* 3.

Sadeh, N., Arunachalam, R., Eriksson, J., Finne, N., and Janson, S. 2003. TAC-03: A supply-chain trading competition. *AI Magazine 24,* 1, 92–94.

Schneider, J. G., Boyan, J. A., and Moore, A. W. 1998. Value function based production scheduling. In *Fifteenth International Conference on Machine Learning*. Madison, WI, 522–530.

Received January 2004; Revised February 2004; Accepted February 2004;