# 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 negotiation. 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 context [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. TAC/SCM 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 are forced to 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 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 inventory profiles.

## 2. THE TAC/SCM GAME

In the TAC-03/SCM scenario,[1] six agents representing computer assemblers operate in a common market environment, over a simulated year. The environment constitutes a *supply chain*, in that agents trade simultaneously in markets for supplies (PC components) and finished PCs. Agents may assemble 16 different models of PCs, defined by the compatible combinations of the four component types: CPU, motherboard, memory, and hard disk. At the end of the game agents are evaluated by total profit, with outstanding inventory valued at zero.

Each day the agent 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. The four CPUs are sold by a single supplier; all other component types are 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.

[1]For complete details of the game rules, see the specification document [Arunachalam et al. 2003]. This is available at `http://www.sics.se/tac`, as is much additional information about TAC/SCM and TAC in general.

## 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 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 component utilization*, based on projections of future customer demand and market equilibrium calculations. The projection of customer demand uses a 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 $\Pr(Q_d \geq Q') = 0.63$. 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. 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.[2] 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, restablished 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 a special day-0 RFQ generation and day-1 offer acceptance strategy.

[2]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.

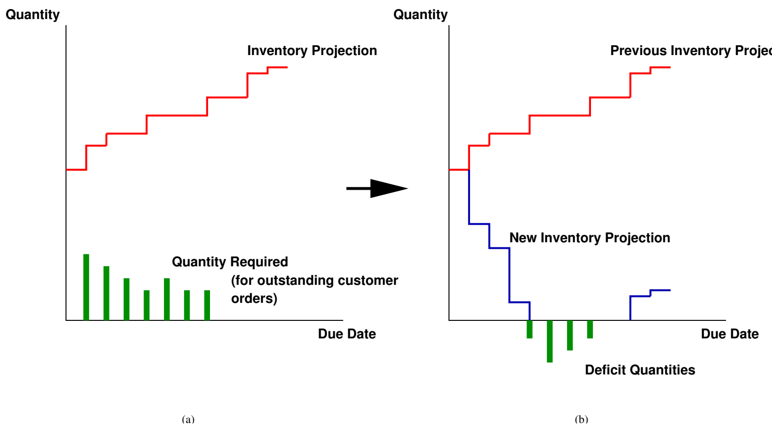

Fig. 1. Generating order-related RFQs for a particular component. (a) Quantities required and (b) Final component inventory projection. RFQs are created for the deficit quantities in (b).

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 is selected to fill all 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.[3] 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 (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 optimization using hill-climbing search. To the extent that an offer is worthwile or not independent of which others are accepted, hill-climbing should quickly find a near-optimal solution.

[3]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 Day 0-2 | 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 |

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

To better understand how this procurement strategy operates in practice we examine results from 28 games Deep Maize played in the TAC-03 tournament.[4] These games span three tournament rounds in which Deep Maize used somewhat different day-0 procurement strategies. The pattern of results presented here also holds when rounds are considered individually instead of in aggregate.

First, we discuss statistics for the offer acceptance search process. Over 99% of search instances completed the basic search, finding a local optimum with respect to a 1-decision neighborhood. Over 80% also completed the extended search with a 2-decision neighborhood, yielding a better solution approximately 15% of the time. The additional nodes in this (much larger) neighborhood can represent the replacement of any accepted offer with another. That this extension infrequently yields better results supports the hypothesis that offers are usually worthwhile independant of which other offers are accepted. Thus, we can be fairly confident that our incomplete search is finding near-optimal solutions. In this case finding the true optimum is not crucial, since the value function being optimized is itself an heuristic approximation.

We next consider a breakdown of how the different types of RFQs Deep Maize sends contributed to procurement in TAC-03. 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. It is also encouraging that Deep Maize purchases 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.

[4]Includes all games except 1241, 1269, and 1429 when Deep Maize experienced network problems. Game 1245 is also excluded from table II because network problems caused anomalies for HarTAC.

Table II. Deep Maize procurement vs. opponents. Prices are normalized to a percentage of the base price.

| | Deep Maize | All Other Agents |
|---|---|---|
| Average offer acceptance rate | 29.2 % | 21.4 % |
| Average component price | 61.5 | 61.2 |
| Quantity ordered day 0 | 37.2 % | 49.0 % |
| Average day 0 price | 0.5 | 0.5 |
| Remaining days quantity ordered | 62.8 % | 51.0 % |
| Remaining days average price | 68.3 | 71.9 |

To provide a reference point for gauging the overall effectiveness of Deep Maize's procurement policy, we compare its behavior against other agents in the same game instances using the metric of average price paid for components. Table II shows that Deep Maize paid approximately the same overall prices as its opponents. However, an interesting difference emerges when we separate out day-0 procurement. Deep Maize purchased relatively fewer cheap components on day 0, but was able to compensate for this by procuring at better prices for the remainder of the game.

Value-driven procurement guided by a reference inventory trajectory was used effectively by Deep Maize in TAC-03. It was flexible enough to express many types of procurement requirements, and robust enough to be used in conjunction with several different initial procurement strategies. Deep Maize's tournament behavior corresponded well to the intended behavior represented by the value function. The resulting procurement policy was competetive with those of other agents. However, there are many possibilities for improvements. Several parameters defining the reference trajectory (e.g. baseline level, baseline premiums) were set somewhat arbitrarily and could be improved by additional analysis, tuning, or learning. Estimates of supplier capacity could be used more effectively to target RFQs at suppliers with available capacity and low prices, potentially improving offer acceptance rates. Supplier capacity estimates could also be used to improve projections of future consumption by identifying times when production will be impossible because key components are not available.

## 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.

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.

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.

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.