Distributed Feedback Control for Decision Making on Supply Chains

Christopher Kiekintveld Michael P. Wellman Satinder Singh Joshua Estelle
Yevgeniy Vorobeychik Vishal Soni Matthew Rudary
Computer Science and Engineering Division
University of Michigan
Ann Arbor, MI 48109 USA
ckiekint,wellman,baveja,jestelle,yvorobey,soniv,mrudary@umich.edu

Abstract

Decision makers on supply chains face an uncertain, dynamic, and strategic multiagent environment. We report on *Deep Maize*, an agent we designed to participate in the 2003 Trading Agent Competition Supply Chain Management (TAC/SCM) game. Our design employs an idealized equilibrium analysis of the SCM game to factor out the strategic aspects of the environment and to define an expected profitable zone of operation. Deep Maize applies distributed feedback control to coordinate its separate functional modules and maintain its environment in the desired zone, despite the uncertainty and dynamism. We evaluate our design with results from the TAC/SCM tournament as well as from controlled experiments conducted after the competition.

Introduction

Networks of interdependent production activities, controlled by distinct organizations, are often characterized as *supply chains*. Supply chains constitute a challenging class of environments for planning and scheduling, for several related reasons. First, agents operating on a supply chain face substantial *uncertainty*. Though they may possess high-fidelity information about their own local state, agents typically have at best general knowledge of other operations on the chain, and access to only noisy summarized data about their states. Nevertheless, they cannot avoid making commitments (e.g., to procure component supplies) before all relevant uncertainty is resolved (e.g., about customer demand). Second, supply chain operations can be highly *dynamic*. Decisions made at one time can propagate to affect conditions elsewhere on the chain at disparate times, and external conditions may cause sudden changes in resource availability and production values. This dynamism can be viewed as just another source of underlying uncertainty, but its temporal structure may enable particular solutions based on dynamic control. Third, the other production operations on the supply chain are also controlled by self-interested agents, whose behavior is naturally *strategic*. This also has the effect of amplifying uncertainty, but again in a particular regular way that should be exploited in reasoning about supply-chain environments.

As discussed in our section on related work below, previous research does not provide a comprehensive solution for planning and scheduling on supply chains. Of course, whether or not they expressly deal with all the issues raised above, many existing methods are applicable to a large degree in supply-chain environments. Rather than propose a complete alternative, therefore, we consider how to adapt or overlay methods designed for localized planning and scheduling in support of operation within a supply-chain context. The emphasis of our approach, therefore, is on how to coordinate the local methods, and to ensure they are given problems within their range of applicability.

We developed our approach in order to participate in the 2003 Trading Agent Competition, Supply Chain Management game (TAC/SCM) (Sadeh *et al.* 2003). As described below, TAC/SCM presents an extremely challenging supply-chain scenario, where agents representing PC manufacturers compete in markets for components and finished goods to maximize profits over a simulated year. The game is far too complex to solve analytically or characterize optimal behavior, due largely to the issues of uncertainty, dynamism, and strategy mentioned above. Although evaluating research in such an uncontrolled and complicated environment can be quite difficult, there are distinct advantages to testing agent strategies in a competitive setting where the other agents and the scenario itself are designed by others. The competition also allows comparison of competing approaches to the same game rather than the more usual situation of different researchers evaluating their approaches in different, or at best slightly different, domains. These benefits are counterbalanced somewhat by the distortions introduced by artificial constraints like tournament deadlines, compression of the computational decision cycle, etc.

The basic underlying idea of our approach is a familiar one: define a desired (from the agent's point of view) or *reference* region of operation, and then use feedback control actions to suppress deviations from the reference region. The feedback control policy itself is complicated by the need to manage interactions with entities at multiple tiers of the supply chain. A natural decomposition of the control architecture would separate out the interactions with distinct agents or classes (suppliers, customers), though of course the decisions for each depend strongly on outcomes with respect to others. We therefore investigate a *distributed* control architecture where the feedback mechanisms operate semi-independently, coordinated through highly aggregated environment parameters. Specifically, our feedback mechanisms

communicate via *price signals* defining the reference zone. We show how to derive such prices in the TAC/SCM game using *equilibrium analysis*, which in turn applies Bayesian evidence evaluation to update demand projections throughout the game. In addition to describing our design principles and the resulting system, we report on the results of the competition, which served in effect as a large case study of our distributed feedback control design ideas.

In the next two sections we discuss related work and describe the TAC/SCM game. We then introduce our agent design, first as a high-level architecture, and then with details of the local decision methods and distributed feedback-control mechanisms. We evaluate our methods in light of results from TAC/SCM competition, as well as a more systematic analysis based on post-competition experiments.

Related Work

Most of the AI and OR literature on scheduling presumes a single agent operating in a deterministic or stochastic environment, with uncertainty (if addressed) typically modeled in terms of stochastic arrival of orders (tasks, jobs, ...) and perhaps some probabilistic resource shocks. There is a large and growing literature in Operations Management dealing directly with supply chains (Chopra & Meindl 2003; Simchi-Levi, Kaminsky, & Simchi-Levi 2002), and information technology for decision support is increasingly prevalent (Lucking-Reiley & Spulber 2001). Although the prior work covers many aspects of supply chain management, it is fair to characterize most of it as taking the perspective of a central optimizer, either of a particular firm in a specified environment, or of some scope encompassing multiple (typically two) interacting agents on the chain. Whereas there are some notable exceptions that address self-interested agents and private information (Carr & Duenyas 2000; Chen 1999), these studies tend (necessarily) to adopt stylized models of the supply-chain environment, in order to derive analytical design results.

In our work, we take the viewpoint of an individual agent operating on the chain, not an overall designer. From this perspective, the agent faces a standard operations management problem in an environment defined by other operations on the chain. Here too, we recognize the existence of a large body of relevant knowledge, and acknowledge that our particular solution for TAC/SCM could undoubtedly make much better use of known techniques. However, it is also clear that no off-the-shelf operations management solution is a complete match for environments as complex as the TAC/SCM game. Therefore, our design is geared toward accommodating existing approaches and tools (optimization and search techniques, simulation methodologies, analytical models, etc.), and coordinating them through aggregate-level signals that may be employed in special-purpose ways within the respective modules.

TAC/SCM Game

In the TAC/SCM scenario,[1] six agents representing PC (personal 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 the market for finished PCs. Agents may assemble for sale 16 different models of PCs, defined by the compatible combinations of the four component types: CPU, motherboard, memory, and hard disk.

Figure 1 diagrams the basic configuration of the supply chain. The six agents (arrayed vertically in the middle) procure components from the eight suppliers on the left, and sell PCs to the entity representing customers on the right. Trades at both levels are negotiated through a *request-for-quote* (RFQ) mechanism, in which buyers issue requests, sellers respond with *offers*, and buyers choose which to accept as *orders*.

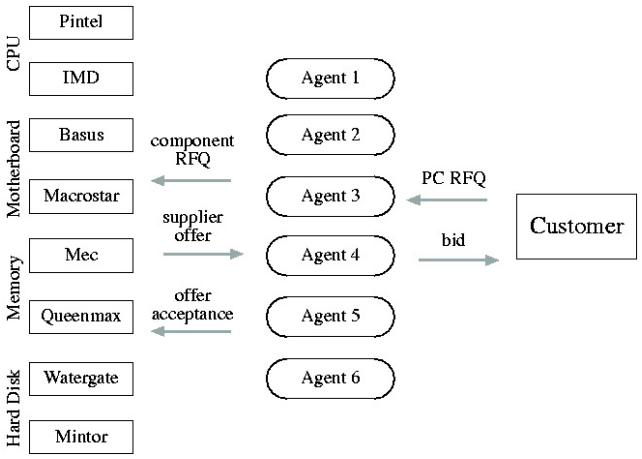

Figure 1: TAC/SCM supply chain.

The game runs for 220 simulated days. On each day, the agent may receive offers and component delivery notices from suppliers, and RFQs and orders from customers. It then must make several decisions:

1. What RFQs to issue to component suppliers.
2. Given offers from suppliers in reponse to the previous day's RFQs, which to accept.
3. Given component inventory and factory capacity, what PCs to manufacture.
4. Given inventory of finished PCs, which customer orders to ship.
5. Given RFQs from customers, which to respond to and with what offers.

In the simulation, the agent has 15 seconds to compute and communicate its daily decisions to the game server. At the

[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.

end of the game, agents are evaluated by total profit, with any outstanding component or PC inventory valued at zero.

Deep Maize Architecture

Our agent, *Deep Maize*, comprises three functional modules, each of which ran on a separate machine in the configuration that played in the 2003 TAC/SCM tournament. The *procurement module* handles all the interactions with suppliers, the *sales module* deals with the customer, while the *factory module* controls the manufacturing and shipping operations. The three modules connect to the TAC server via a communication layer, which also provides a central cache for inter-module information sharing. Factors motivating our multi-processor distributed architecture included the need to decompose a complex task and to accelerate the agent development process through parallel design and implementation. The distributed architecture in turn raised coordination issues, in particular how to keep the procurement and sales modules working toward consistent ends, cognizant of the state of the factory.

Distributed Feedback Control

As for real supply chains, agents operating in TAC/SCM are presumed to be self-interested, and so the scenario constitutes a *non-cooperative game*. The complexity of this game, however, precludes the direct application of game-theoretic solution methods. Therefore our first major goal was to factor out (through summarization or mitigation, by methods discussed below) the possible effects of strategic play, enabling us to view TAC/SCM as a stochastic dynamical system to be controlled by our agent. Once we can view the agent's problem as a control problem instead of as a game, then a number of different approaches from control theory become available. One possible approach would be to build (or learn) a model of the stochastic effects of all of our agent's actions and then to use dynamic programming to solve the resulting optimal control problem. However, both the model generation and solution tasks are intractable in this setting. Another approach is that of *reference feedback control*, in which one defines a reference trajectory in state space and implements a policy that measures deviation from the reference trajectory and takes (usually short-term) corrective actions (Astrom & Wittenmark 1994). We turned to feedback control in our design for two main reasons: (1) it was conceptually straightforward and computationally efficient to implement, and (2) it promised some robustness to the surprises we would inevitably encounter in the TAC/SCM competition.

Consider how one might implement feedback control in our agent, starting with the sales module. We would need to define a reference customer sales trajectory for the year and then employ a bidding policy to attempt to follow this trajectory. Similarly, for the procurement module we would define a reference delivery trajectory and employ a supplier negotiation policy to follow it. Two complications arise immediately. First, we cannot specify these references statically, given the dynamic, uncertain, and strategic nature of the environment. Second, the procurement and sales activities interact, and by defining their control problems independently of each other we preclude the possibility of coordinating their response to events in their common environment. We address the second problem in part by defining the goal for both modules in terms of a reference *inventory* trajectory, where inventory represents the accumulation of deliveries net of depletion by sales. We then employ aggregate market analysis techniques (described below) both to set the reference trajectory dynamically based on observed conditions on the supply chain, as well as to calibrate the control actions available to each module.

Maintaining the Reference Trajectory

Figure 2 presents a high-level view of Deep Maize's feedback control design. At this level, it is useful to partition the agent's experience of its complex and dynamic environment into three *zones*: a middle *reference zone* in which the agent is approximately meeting its reference inventory trajectory, a lower *short-term deficit* zone in which the agent has too little inventory in the short run (thereby losing out on sales opportunities and paying penalties on existing orders), and an upper *long-term surplus* zone in which the agent has too much inventory in the long run (thereby likely to end up with unused and wasted components). Too much inventory in the short-term only is not a deviation from reference because we expect to deplete it over time and there are no holding costs. Similarly, too little in the long-term is not classified as a deviation because there remains ample opportunity to procure supplies for the future.

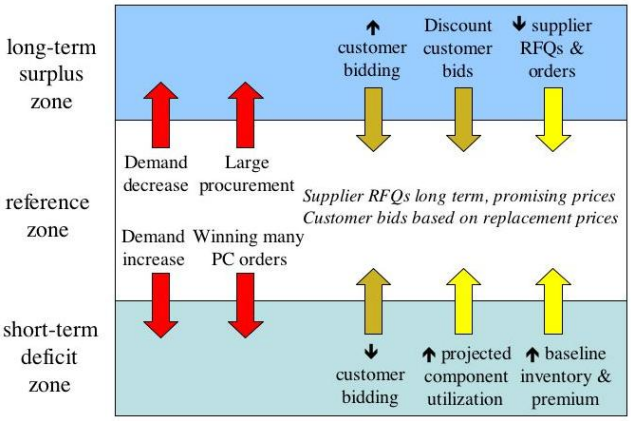

Figure 2: Operating regions and control actions employed by Deep Maize in response to environment changes.

The arrows on the left side of Figure 2 show some of the stochastic events that can perturb the system away from the reference zone. For example, demand may increase or the agent may win an unexpectedly large number of PC orders, pushing it into the bottom zone. In such a case, the sales module can modify its bidding policy to reduce the rate of new PC orders, and the procurement module can more aggressively seek short-term component deliveries. A decrease in demand or a large procurement of supplies can cause a deviation from reference in the opposite direction. In that

case, the sales module will increase PC sales while the procurement module will reduce future component purchases.

Market Analysis

The primary way that we account for competitive behavior of the other agents is through an aggregate analysis of the market for PCs and components. Deep Maize continually evaluates the conditions of supply and demand, and attempts to find their balance, or *equilibrium* point. It uses the results of this market analysis to establish the reference inventory trajectory, and to parametrize the control actions of the sales and procurement modules.

Price Equilibrium

In the TAC/SCM market, suppliers set prices for components based on an analysis of their available capacity. Prices for finished PCs are determined by a competitive bidding process. Conceptually, there exist separate prices for each type of component, from each supplier, and for each individual PC order. Moreover, these prices vary over time: both the time that the deal is struck, and the time that the good is promised for delivery.

Our analysis simplifies this picture by assessing the equilibrium of an idealized steady state, in which component demand is constant and served uniformly over time. Symmetries inherent in the game specification then enable us to reduce the dimensionality of the price system by relating the prices of every component and finished PC to a central index price, p. Specifically, the TAC/SCM component catalog (Arunachalam *et al.* 2003) associates every component c with a *base price*, p_c. The base price of a particular PC model, *PC*, can then be defined as the sum of base prices of its constitutent components, $p_{PC} \equiv \sum_{c \in PC} p_c$. The overall price level, p, then represents the price of all goods, as a multiple of their respective base prices. An *equilibrium price level* is a value of p such that the aggregate quantity demanded at that price level equals the quantity supplied.

Supply Function: The correspondence between price and quantity for component supplies is defined by the suppliers' pricing formula (Arunachalam *et al.* 2003). The normalized price offered by a supplier at day d for an order to be delivered on day $d + i$ is

$$p(d+i) = 1 - 0.5\frac{c_{avail}(d+i)}{500i}, \qquad (1)$$

where $c_{avail}(j)$ denotes the cumulative capacity 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.

Supplier prices according to Eq. (1) are date-specific, depending on the particular pattern of capacity commitments in place at the time the supplier evaluates a *request for quote* (RFQ). In the steady state, commitments are uniform in any given interval,[2] in which case we can express supplier prices in a date-independent manner:

$$p \;=\; 1 - 0.5\frac{(500 - q_c)}{500} = 0.5 + \frac{q_c}{1000}, \qquad (2)$$

where q_c is the daily production quantity of this particular component. Let q be the steady-state daily production level, in units of PCs produced. By symmetry of the customer demand and supplier costs and capacities, each PC model is equally likely to be produced. Each CPU component, therefore, appears in 1/4 of PCs, for a total of $q/4$ produced per day in the steady state. Non-CPU components each appear in 1/2 of PCs. These, however, are manufactured by two suppliers each, so also are produced at the level of $q/4$ per day per supplier. Thus, for every supplier-component combination, the steady-state relationship between price level and PC production quantity can be expressed as

$$p = 0.5 + \frac{q/4}{1000} = 0.5 + \frac{q}{4000}. \qquad (3)$$

Equation (3) is based on the actual pricing policy implemented by suppliers, and therefore represents the effective *supply function*. Note that the function dictates the *marginal price* faced by a component buyer, that is, the cost of purchasing an additional component unit given an overall PC economy operating at production level q. Given the incremental formation of supply deals, the *average price* paid for components is considerably less.

Demand Function: Demand for PCs on a given day is characterized by the set of customer RFQs issued, each specifying a PC model, quantity, late penalty, and *reserve price*. The reserve price indicates the maximum the customer is willing to pay. Although the actual price is determined by competitive bidding among the manufacturing agents, the customer is committed to buy if any bid is at or below its reserve. Thus, we can define the overall demand q at price level p as the aggregate number of PCs appearing in RFQs with (normalized) reserve price p or more.

The underlying demand level is defined by an integer parameter Q, which evolves according to a given stochastic process, discussed below. Each day, the customer issues a set of $\hat{Q}$ RFQs, where $\hat{Q}$ is drawn from a Poisson distribution with mean value Q. The number of PCs requested in an RFQ is drawn uniformly from the interval [1,20], and so the expected total quantity of PCs requested is $10.5Q$. Reserve prices are drawn uniformly from the interval [0.75, 1.25]. The expected total quantity of PCs requested at a price of p or less, therefore, can be described by

$$q = \begin{cases} 10.5Q & \text{if } p \le 0.75 \\ 10.5Q\left(\frac{1.25-p}{0.5}\right) & \text{if } 0.75 \le p \le 1.25 \\ 0 & \text{if } p \ge 1.25. \end{cases} \qquad (4)$$

Equilibrium Price and Quantity: To derive the price equilibrium, we simply solve for the intersection of supply (3) and demand (4) functions. Combining the two equations for the case when $0.75 < p \le 1$, we obtain

$$p^* = \frac{2000 + 26.25Q}{4000 + 21Q}. \qquad (5)$$

[2] To the extent the pricing formula (1) reflects actual costs, we see that costs are increasing with committed capacity. Efficient production would thus indeed spread out component manufacture as evenly as possible.

We can verify that Eq. (5) leads to prices in the assumed range, as long as $95 < Q < 380$. The actual range for Q is [80,320], so we need consider only the lower boundary. For $Q \leq 95$, the effective demand quantity is the entire amount requested, so prices are determined entirely by the supply function (3), with $q = 10.5Q$.

Customer Demand Projection

The equilibrium price (5) depends centrally on Q, the stochastic parameter controlling overall demand. In each game instance, an initial value, Q_0, is drawn uniformly from [80,320]. If Q_d is the value of Q on day d, then its value on the next day is given by (Arunachalam *et al.* 2003):

$$Q_{d+1} = \min(320, \max(80, \tau_d Q_d)), \quad (6)$$

where τ is a trend parameter that also evolves stochastically. The initial trend is neutral, $\tau_0 = 1$, with subsequent trends updated by a perturbation $\epsilon \sim U[-0.01, 0.01]$:

$$\tau_{d+1} = \max(0.95, \min(1/0.95, \tau_d + \epsilon)). \quad (7)$$

Each day, the agent observes a number of RFQs, $\hat{Q}_d \sim Poisson(Q_d)$. We can represent the transition and observation graphically in a Bayesian network fragment, as illustrated in Figure 3. Note that the updated demand, Q_{d+1}, is a deterministic function of Q_d and τ_d, as specified in Eq. (6).

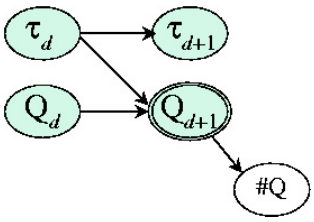

Figure 3: Bayesian network model of demand evolution.

This deterministic relationship simplifies the update of our beliefs about the demand state given a new observation. Let $\Pr(Q_d, \tau_d)$ denote the probability distribution over demand states given our observations through day d. W can incorporate a new observation $\hat{Q}_{d+1}$ as follows:

$$\Pr(Q_d, \tau_d | \hat{Q}_{d+1}) \propto \Pr(\hat{Q}_{d+1} | Q_d, \tau_d) \Pr(Q_d, \tau_d). \quad (8)$$

Eq. (8) exploits the fact that the observation at $d+1$ is conditionally independent of prior observations given the demand state (Q_d, τ_d). We can elaborate the first term of the RHS using the demand update (6):

$$\Pr(\hat{Q}_{d+1} | Q_d, \tau_d) = Poisson(\min(320, \max(80, \tau_d Q_d))).$$

After calculating the RHS of Eq. (8) for all possible demand states, we normalize to recover the updated probability distribution. We can then project forward our beliefs over (Q_d, τ_d) using the demand and trend update equations (6) and (7), to obtain a probability distribution over (Q_{d+1}, τ_{d+1}).

A sequence of updates by Eq. (8) represents an exact posterior distribution over demand state given a series of observations. To maintain a finite encoding of this joint distribution, we consider only integer values for demand, $Q \in \{80, \ldots, 320\}$, and divide the trend range $\tau \in [0.95, 1/0.95]$ into T discrete values, evenly spaced. For demand estimation in the 2003 tournament, Deep Maize took $T = 11$, for a total of 2651 possible demand states.

Given a distribution over demand states, we can project forward to evaluate the expected demand at future points in time. Projection calculations yield the distribution over Q for particular days, as well as the mean Q over a range of days, up to the horizon of the game. The derived Q values can then be employed in equilibrium calculations, for particular days or longer periods.

Reference Inventory Trajectory Generation

The reference inventory trajectory is the sum of three sources of component requirements. First, for the immediate term, *existing customer orders* for PCs entail a definite commitment of inventory. Second, beyond the horizon of current orders, we derive the time series of *expected component utilization*, based on the price equilibrium and demand projection calculations described above. Deep Maize assumes that the overall market activity level on a given day will be the competitive equilibrium quantity corresponding to the demand projected for that day, and expects that it will garner 1/6 of that quantity on average. It also assumes that the distributions of PC orders it gets will have, on average, a uniform distribution over the possible PC types. To account for unpredictable movement in demand trends, Deep Maize sets a somewhat more conservative reference, based on the demand quantity Q' satisfying $\Pr(Q_d \geq Q') = 0.63$, as opposed to the expected Q_d (the particular threshold 0.63 is somewhat arbitrary). Over the range where existing and prospective customer orders overlap, we phase in the expected utilization proportionally.

The final component of our inventory reference is a baseline level that we attempt to maintain in order to mitigate short-term noise in our procurement and sales activity. Having the buffer increases the agent's flexibility, allowing it to act more opportunistically. Deep Maize sets the buffer level at 6.0 times the current expected daily utilization (this multiplier is also somewhat arbitrary). The baseline level is scaled gradually to zero over the last 25 days of the game, at which point accumulated inventories become worthless.

In tracking the reference inventory trajectory, the modules take into account current inventory of components and finished PCs, as well as anticipated deliveries of components the agent has already committed to purchasing.

Functional Modules

The procurement module can request supplies for delivery on any future day, thereby excercising direct influence over the inventory trajectory in both the short and long term. However, this influence is one-directional, as procurement can only add to inventory. The sales module can move inventory in the other direction, by selling PCs. Since customer RFQs have a maximum horizon of 12 days in advance, the sales module can exert direct influence only in the short term. The factory module does not attempt to influence the overall inventory level, but rather serves as conduit between

procurement of components and PC sales. We describe the particular policies of each module in turn.

Procurement Module

To acquire components, an agent sends RFQs to a supplier (up to 10 per supplier per day), each specifying a desired quantity and due date. The supplier responds with an offer specifying quantity, due date, and price. If the supplier cannot meet the requested quantity and date, it will instead offer a partial quantity at the requested date and/or the full quantity at a later date, to the best of its ability given its existing commitments. Agents must accept or decline each offer the day they receive it. Thus the procurement module has two tasks: *RFQ generation* and *offer acceptance*.

The RFQ mechanism generates RFQs to bridge the gap between the current and reference inventory trajectory. One RFQ is generated for each customer order that cannot be fulfilled using current inventory, and one RFQ is sent to each supplier of a component for any shortage in the baseline inventory. The remaining RFQs request components addressing the long-term expected component utilization on particular days. To stay within the quota of 10 RFQs per supplier, we give priority to those addressing customer orders, then baseline, then utilization for days expected to be available at the lowest prices.[3]

The offer acceptance mechanism has to decide on the subset of all offers to accept. This presents an optimization problem: selecting the subset that leads to the state with maximal value. The value of a state is the sum of incremental values of the components accepted. Components necessary to fill outstanding customer orders are assigned high values (the entire value of the customer order plus any penalties that may be due). Components that fill baseline inventory are valued at the equilibrium price plus a *baseline premium*, defined on a sliding scale from 25%-100% of the component base price. Components addressing expected component utilization are valued at their *equilibrium price*, based on our market analysis discussed above.

Given a representation of the value function, we can evaluate any subset of supplier orders in terms of the resulting inventory trajectory and expense. Since our decision time is limited, we search the space of candidate offer bundles using hill-climbing, starting from the state accepting the maximal offer set. Deep Maize terminates the search when it finds a local minimum, or runs out of time.

Reference Zone Policy: Sufficient short-term inventory means that there is no need to generate short-term RFQs. Any long-term deficit in inventory for a component will cause RFQs to be generated for that component. The resulting offers will be accepted only if they are priced at less than the projected equilibrium price.

Short-Term Deficit Zone Policy: For short-term deficits, the procurement module generates customer-order and baseline inventory RFQs. If there is long-term deficit as well, then RFQs for expected component utilization are also generated with lower priority. The offer acceptance algorithm places high values on offers that can fill customer-orders or baseline inventory needs, accepting them regardless of price. Offers that fill long-term needs will be accepted if they are priced at less than projected equilibrium prices.

Long-Term Surplus Zone Policy: RFQs for future expected component utilization will not be generated in this zone. Any simultaneous short-term deficit is dealt with by the policy for that case.

Sales Module

Each day, the sales module receives a set of customer RFQs, each specifying a PC type, quantity, due date, late penalty, and reserve price. It decides which customer RFQs to bid on, and how much to bid on each one. The customer receives bids from all the agents and accepts the lowest offer, as long as it is below the reserve price. The sales module acts to maintain the reference inventory trajectory by bidding under the assumption that components used to serve any new customer orders must be replaced. It associates with each order a *component replacement cost*, calculated as the sum of equilibrium prices for the components necessary to fulfill the order.

We simplify the parallel-bidding problem faced, by treating the auctions as strategically independent. We model each RFQ as a first-price sealed-bid (FPSB) auction with independent private values (negative costs), symmetric and uniformly distributed (Krishna 2002)). According to the standard FPSB analysis, the equilibrium bidding strategy is

$$b(c) = c + \frac{1}{N}(B - c), \tag{9}$$

where c is the agent's own cost, B is the upper bound on the cost distribution, and N is the number of agents. We set B to the summed base price (an upper bound on actual component prices), and select $2 \leq N \leq 6$ by assessing the *effective* number of bidding agents (the maximum in low- or moderate-demand situations, tapering off as overall demand increases). Under idealized conditions (violated by interdependencies between auctions in the actual TAC/SCM environment), Eq. (9) maximizes expected profits, selecting the optimal markup over costs given the competitive bidding environment assumed.

The sales module bids on any RFQ that can be fulfilled, taking into account current customer orders, current inventory and pending component deliveries, and factory capacity.[4] RFQs are considered one-by-one. Successive considerations assume that any bids placed on RFQs will be won, so their associated resources are considered unavailable.

Reference Zone Policy: In this zone, short-term inventory is not a binding constraint; the agent can bid on as many customer RFQs as factory capacity allows. Prices offered to customers are based on replacement costs, with profits taken according to the second term in the bidding formula (9).

[3] Deep Maize maintains an assessment of each supplier's available capacity profile, based on the offers we have seen and the specified supplier pricing function (1). This yields an estimated upper-bound on price for each possible date. We then select probabilistically favoring lower priced dates.

[4] In the tournament Deep Maize placed additional "aggressive" bids beyond these constraints for premium prices. Discussion of these bids is omitted for brevity.

Short-Term Deficit Zone Policy: The customer module will bid on only as many PCs as can be produced with the current inventory and pending component deliveries. This helps to reduce the risk of paying large penalties if components cannot be acquired in time to fill the customer RFQs.
Long-Term Surplus Zone Policy: The sales module tries to increases its probability of winning customer orders by *discounting* bid prices. In this zone, it is not presumed necessary to replace all components used, and so the effective cost c in Eq. (9) can be reduced. Deep Maize employs a discounted replacement cost of ce^{-kx}, where x is the number of surplus components and k is a scaling parameter. In the limit, as surplus increases, component costs are treated as fully sunk.

Factory Module

The factory module decides which finished PCs to ship to which customers, and how many of each type of PC to assemble on the following day. These are interesting scheduling problems in their own right, but our discussion is brief because they do not directly affect the overall inventory trajectory. Given stochastic models of customer orders and component arrivals, we could use dynamic programming to solve these problems. Instead, for computational efficiency, we optimize over a three-day horizon, considering only the components that are already in inventory or due to arrive the next day. This turns out to be an integer linear programming problem, which we solve using CPLEX.

Using the local policies described above, the three Deep Maize modules are able to coordinate their activies by communicating only information related to the reference inventory trajectory: PC orders currently held, projected customer demand, current inventory, and anticipated component deliveries. The local policies employed by the three modules are designed to be profit maximizing when the TAC/SCM market is in equilibrium and the agent is opertating in the reference inventory zone. The procurement and sales modules deal with tumultuous conditions by using their joint capabilities to force Deep Maize back into the profitable reference zone as quickly as possible.

Empirical Analysis

We evaluate Deep Maize's performance in the tournament and in post-tournament controlled experiments.

Results from the TAC/SCM Tournament

Through a series of qualifying, seeding, and semifinal rounds, the original pool of 20 competing in TAC/SCM was whittled down to six finalists. The final round—held on 13 August 2003 in Acapulco, in conjunction with IJCAI-03—featured a series of 16 games among the finalist group. Deep Maize emerged with the second highest score.

There are many factors bearing on overall score, and so gross tournament results provide only suggestive evidence for the efficacy of Deep Maize's strategy and approach. To assess the specific performance of our distributed feedback-control algorithms, we measured the short-term deficit and long-term surplus deviations from the reference inventory trajectory in the tournament games. Figures 4 and 5 present deviation measurement results for a single component from two different finals games.

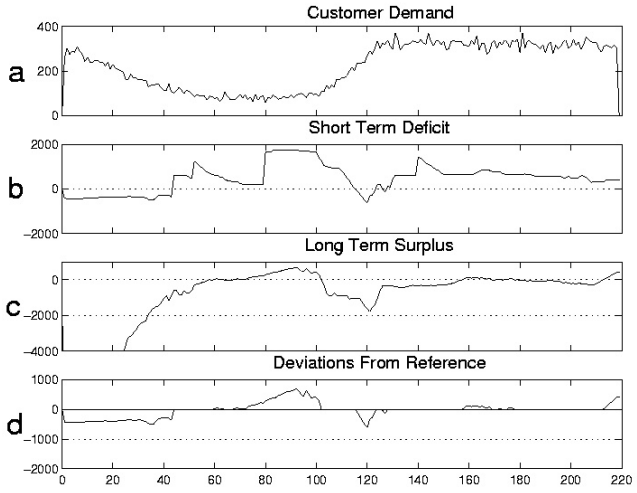

Figure 4: Deviation analysis for a tournament game. For the deficit graph, only negative regions represent deviations, and for the suplus graph only the positive regions are deviating.

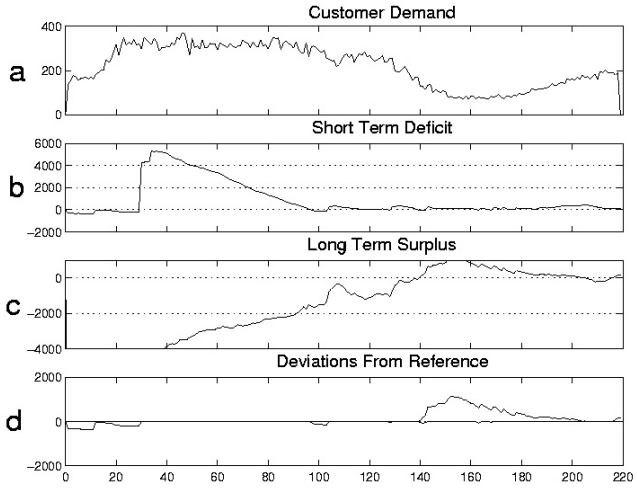

Figure 5: Results from another tournament game.

The x-axes in these diagrams represent the days of the TAC/SCM year. Panel (a) depicts the time series of customer demand, which as seen here, can swing from extreme to extreme fairly quickly. Panel (b) displays the short-term deficit, based on normalizing the reference inventory trajectory to zero for all days. Panel (c) shows the long-term surplus, again based on a normalized reference. The effects of feedback control are apparent in both plots, as perturbations are suppressed again and again. Panel (d) plots deviations only: the curve above zero corresponds to long-term surplus and the curve below to short-term deficits.

We can observe other interesting effects in these games.

In Figure 4, between day 80 and day 100 as the customer demand bottoms, we accumulate both short-term and long-term surplus. Shortly after day 100, demand starts trending higher, leading quite quickly to a short-term and long-term deficit. Figure 5(b) reflects a very large delivery of components around day 30, a perturbation from which it takes Deep Maize over 50 days to recover. Around day 140, the game shifts from high to low demand, and we see on Panel (d) a brief instance of the interesting (and difficult) situation where there exists both short-term deficit and long-term surplus.

Figures 4 and 5 are representative of the kinds of deviation plots we obtained from the games of the tournament. The tournament data indicate that Deep Maize was effective in suppressing deviations from the reference inventory trajectory. What the tournament data results cannot show, however, is how effective or ineffective different pieces of our feedback control strategy were. For that we turned to controlled experiments we conducted with a local copy of the TAC/SCM game server since the tournament ended.

Results from Controlled Experiments

Our goal in these experiments was to measure the effectiveness of specific feedback control loops by disabling them and comparing performance with the full Deep Maize agent. Performance can be sensitive to the strategies of other agents in the game and there is no unique or correct choice for these strategies. Experience showed that the agents' initial procurement policy is a particularly important strategic variable (to be analyzed in a separate report), and so we paid specific attention to this factor. Specifically, five of the agents in our test environment use an initial procurement policy selected randomly each game from a representative sample of the policies actually used by the other tournament finalists. One agent uses the Deep Maize policy. Except for this strategic decision, the behavior of all agents is based on Deep Maize, which provides a reasonable level of overall market activity.

	no discount	no premium
Profit	-21.9% ✓	+17.5% ✓
Customer Orders	-17.9% ✓	-4.5% ✓
Short-term Deficit (CPUs)	-7.3% ✓	-1.1%
Long-term Surplus (CPUs)	+91.8% ✓	+2.4%
Short-term Deficit (other)	-3.6%	+12.3% ✓
Long-term Surplus (other)	+170.5% ✓	-5.2%

Table 1: Table of results for controlled experiments, expressed as percentage differences from an unmodified version of Deep Maize.

We tested the performance of two modified versions of Deep Maize. In *No-Discounting*, the sales module refrains from discounting its customer bids, regardless of the long-term surplus. In *No-Premiums*, the procurement module adds no premium to the value assigned to baseline inventory. Both versions used the randomly selected initial procurement policy, as did the other unmodified agents. We ran 38 games with these two modified agents, and measured various aspects of agent performance against the average performance of the unmodified agents. The results are reported in Table 1. Each column reports the percentage difference between the mean measurement for the modified version and the full Deep Maize agents. Entries with check marks are statistically significant at the 0.05 level, using a paired t-test.

First, we analyze the results for the agent with no sales module discounting. Our hypothesis is that failing to discount should increase long-term surplus, decrease customer orders, and decrease score. The data show strong support for all of these effects. Long-term surplus more than doubles for non-CPU components, and both customer orders and profits fall off significantly. Interestingly, not discounting also has the indirect effect of decreasing the short-term deficit. One possible explanation for this is that the extra orders won by discounting may use up short-term inventory stocks, causing deficits until additional components arrive. Overall, the experimental data indicates that discounting is both consequential and effective in achieving the reference trajectory.

Next, we analyze the results for the no-premiums agent, which are mixed. The hypothesis for this case is that removing premiums should increase the short-term deficit, decrease customer orders, and decrease score. The expected differences did show up in customer orders, and in the short-term deficit for non-CPU components. The short-term deficit for CPU components did not show a significant difference. We suspect that the fact that each CPU type is available from a sole supplier makes CPU procurement more noisy than buying other components.

More unexpected was the increase in profit obtained by the no-premiums agent. Coupled with the difference in orders, this implies that the agents using premiums succeed in producing more but that this additional production is unprofitable. Follow-on experiments will investigate whether the control can be fixed by decreasing the baseline level or premium, or whether some alternative mechanism would be better. It would also be useful to analyze the effects of different configurations of other agents (we have checked some small variations without observing qualitatively different results). Regardless, it seems clear that future versions of Deep Maize could benefit from better policies for handling short-term deficits in the procurement module.

Conclusion

Deep Maize makes decisions on the TAC/SCM supply chain using a distributed feedback control algorithm, tracking a reference inventory trajectory defined by aggregate market analysis. The market analysis factors strategic interactions out of the game by summarizing their expected effects. One interaction not covered by this analysis is mitigated by a special-case strategy.

The reference inventory trajectory is used to coordinate Deep Maize's procurement and sales activities, which are controlled by separate functional modules. Maintaining the reference zone globally also restricts the problems faced by the individual modules. These problems are substantially easier than the full SCM problem, and invite the use of additional methods and analytic tools that may not be applicable to all situations faced by Deep Maize. We believe that

feedback control designs could be applied to other complex domains to aid in problem decomposition and limit the effective scope of subproblems to instances that can be solved effectively using known techniques.

Evaluation of the results from the TAC-03 tournament confirmed that feedback control was effective in Deep Maize. The design led to profitable production through many rounds of tournament play, and achieved the second highest score in the finals. The robustness of the design was apparent in solid overall performance in many qualitatively different tournament environments, and at a finer level in the agent's ability to consistently recover from reference inventory deviations. Controlled experiments support a higher fidelity analysis, indicating that some of Deep Maize's control actions should be modified for competition in next year's TAC/SCM tournament.

Acknowledgments

We gratefully acknowledge the help of a great 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 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. Technical report, SICS. Draft Version 0.62.

Astrom, K. J., and Wittenmark, B. 1994. *Adaptive Control*. Addison-Wesley, second edition.

Carr, S. C., and Duenyas, I. 2000. Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Operations Research* 48:709–720.

Chen, F. 1999. Decentralized supply chains subject to information delays. *Management Science* 45:1076–1090.

Chopra, S., and Meindl, P. 2003. *Supply Chain Management*. Prentice Hall, second edition.

Krishna, V. 2002. *Auction Theory*. Academic Press.

Lucking-Reiley, D., and Spulber, D. F. 2001. Business-to-business electronic commerce. *Journal of Economic Perspectives* 15(1):55–68.

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.

Simchi-Levi, D.; Kaminsky, P.; and Simchi-Levi, E. 2002. *Designing and Managing the Supply Chain*. McGraw-Hill, second edition.