Controlling a Supply Chain Agent Using Value-Based Decomposition

Christopher Kiekintveld, Jason Miller, Patrick R. Jordan, and Michael P. Wellman
University of Michigan
Artificial Intelligence Laboratory
Ann Arbor, MI 48109 USA
ckiekint@umich.edu

ABSTRACT

We present and evaluate the design of Deep Maize, our entry in the 2005 Trading Agent Competition Supply Chain Management scenario. The central idea is to decompose the problem by estimating the value of key resources in the game. We first create a high-level production schedule that considers cross-cutting constraints and future decisions, but abstracts aways from the details of sales and purchasing. We then make specific sales and purchasing decisions separately, coordinating these decisions with the high-level schedule using resource values derived from the schedule. All of these decisions are made using approximate optimization techniques and make use of explicit predictions about market conditions. Deep Maize was one of the most successful agents in the 2005 tournament, both in overall performance and on specific measures that emphasize coordination.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Design,Experimentation

Keywords

Optimization, Scheduling, Decomposition, Coordination, Trading Agents, Supply Chain Management

1. INTRODUCTION

We present and evaluate the design of Deep Maize, a successful entrant in the 2005 Trading Agent Competition Supply Chain Management scenario (TAC SCM). The main idea of our design is to estimate marginal values for each finished product and component input as accurately as possible, given predictions about market conditions and constraints on production. These values provide a way to decompose the agent's decisions into manageable subproblems while retaining many of the advantages of global optimization. We first perform a centralized, high-level optimization to create a long-term projected production schedule. This optimization accounts for the major constraints that act across markets and over time, but abstracts away from the details of interacting with suppliers and customers. We make specific decisions about actions in the individual markets using a second stage of optimization. To coordinate these second-stage optimizations with the high-level production plan we introduction the notion of resource values. These values derive from the production plan and are used to form a modified objective function for the low-level decisions. They transmit information about global market conditions and constraints to the individual subproblems.

We evaluate our approach in comparison with other TAC SCM agents. These agents represent a diverse set of strategies designed by other researchers, and the ability to benchmark against such a set is one of the important advantages of participating in TAC. Deep Maize was one of the most successful agents in the 2005 competition. However, there are many factors that contribute to overall agent performance. To assess more thoroughly our value-based decomposition methods we present additional comparisons that focus on aspects of agent performance that require coordination between decisions across markets and over time. These include analysis of inventory management, market selection, and market timing. Deep Maize performs strongly on these more specific measures, providing further evidence that value-based decomposition is a very effective design paradigm.

In the next section, we introduce the TAC SCM scenario and discuss in more detail the coordination problem our design addresses. Section 3 presents a background of related work that is useful for understanding the motivation of our design. Section 4 presents an overview of the main elements of the design, and the following section gives more detailed descriptions. Section 6 adds some further discussion of the design in the context of prior work. We end with an evaluation of the design and concluding remarks.

2. DECISIONS ON A SUPPLY CHAIN

2.1 TAC SCM Scenario

In the TAC SCM scenario [7], [1] six agents representing PC (personal computer) manufacturers compete to maximize their profits over a simulated year. There are 220 simulation days, and agents have approximately 14 seconds to make decisions each day. Agents participate simultaneously in markets for supplies (components) and finished PCs. There are 16 different types of PCs (divided into three market segments), defined by the compatible combinations of 10 different component types. Components fall into one of

[1]For complete details of the game rules, see the specification document [4]. This is available at `http://www.sics.se/tac`, along with software and additional information about TAC.

four categories: CPU, motherboard, memory, and hard disk. There are four types of CPUs and two types of all other components; one component from each category is required to produce a PC.

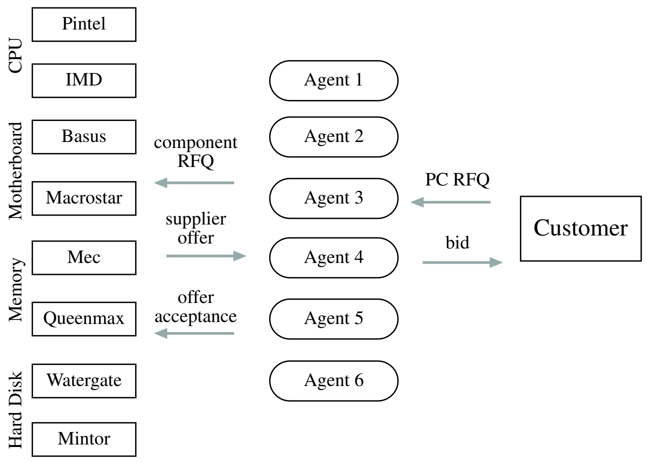

Figure 1: TAC SCM supply chain configuration.

Figure 1 shows the overall configuration of the supply chain. The six manufacturers are in the middle of the chain; these are the agents implemented by tournament participants. They procure components from eight suppliers on the left and sell PCs to customers, shown on the right. Trades at both levels are negotiated through a *request-for-quote* (RFQ) mechanism, which proceeds in three stages:

1. Buyer issues RFQs to one or more sellers.
2. Sellers respond to RFQs with *offers*.
3. Buyers accept or reject offers. An accepted offer becomes an *order*.

The supplier and customer agents implement policies defined by the game specification and implemented in the server. The suppliers have limited production capacity that varies during the game according to a random walk. They make offers and set prices based on their ratio of available capacity. The customer agent generates requests for PCs each day. The number generated is based on a separate random walk with a trend parameter for each market segment. If a manufacturer does not deliver the order on time, it pays a penalty specified in the request. Manufacturers face substantial uncertainty in both markets. The underlying supplier capacities, customer demand parameters, and local state of other manufacturers are not directly observable, so agents must estimate these from other sources of information. There is also strategic uncertainty, since agents do not know the exact strategies employed by their competitors.

Each manufacturer is endowed with an identical factory that has limited production capacity, measured in "cycles." Each PC type requires a different number of cycles to produce. Agents pay storage costs for all components and PCs held in inventory each day, and are charged (or paid) interest on bank balances. At the end of the game agents are evaluated based on total profit, and any remaining inventory is worthless.

Figure 2 shows the decisions each agent makes during a game and the negotiation procedures for interacting with suppliers and customers. On each day the agent may receive offers and component deliveries from suppliers, and RFQs and orders from customers. The agent must then make five decisions:

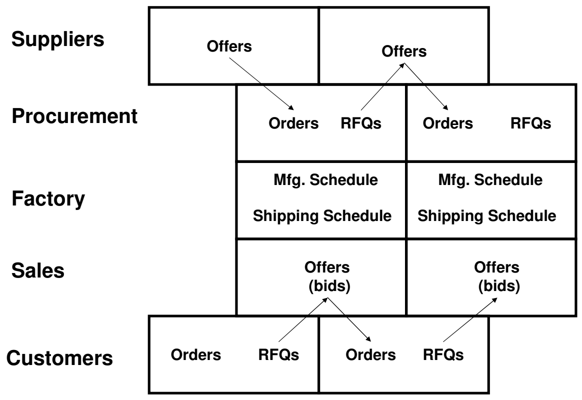

Figure 2: Decisions and SCM agent makes during a game.

1. What RFQs to issue to component suppliers.
2. Of the offers received from suppliers (based on 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 and with what offers.

Customer and supplier decisions are interleaved with manufacturer decisions.

2.2 Coordination Problem

The TAC SCM scenario is challenging for many reasons including computational complexity [1], dealing with uncertainty [3, 13], and strategic interactions [23]. A particularly challenging aspect of the scenario is the problem of coordinating decisions at different levels of the supply chain over time. Agents must integrate decisions about factory operations, supply purchasing, and customer sales. Each of these decisions has a different information structure and requires different types of reasoning and interaction with other agents, but they interact through cross-cutting constraints (e.g., factory capacity). Agents must also plan ahead to ensure that they have inventory when it is needed.

To make the coordination problem more tangible, consider the following examples of how decisions might interact. First, suppose that profit margins are high and the manufacturer is operating at full capacity.The agent should try to determine the optimal mix of products, which requires assessing information about prices in both the customer and supplier markets to determine which PC types offer the highest profits. It also requires the agent to change customer prices to induce the desired sales distribution and purchase the correct numbers of each type of component so that it does not run out of key components or overstock others. Now, suppose that earlier in the game margins were much lower, and the agent was able to predict that prices would improve (perhaps by detecting a trend towards greater customer demand). In this case the agent would be able to improve profits by producing extra PC inventory and saving it until prices in the customer market improved. This requires the agent to purchase extra components for future use, while simultaneously raising prices in the customer market to reduce the number of sales and build inventory.

These qualitative examples are intended to illustrate the types of coordination and reasoning an agent may need to carry out. Situations in the actual game require agents to make quantitative decisions, and may present ambiguous or contradictory circumstances that require difficult tradeoffs.

3. RELATED WORK

The operations research literature contains various techniques for decomposing Linear Programming (LP) problems. Of these, the various price-directive decompositions are most closely related to our approach. The basic form of this technique is the well-known Dantzig-Wolfe decomposition [6]. The idea is to break the problem into a master LP and multiple subproblems, all of which are simpler and easier to solve than the original "monolithic" LP. The master problem takes into account cross-cutting constraints, while the subproblems contain additional local constraints. A solution is found by iteratively solving the master problem and subproblems, passing information in both directions on each iteration. The master uses estimates of the demand for each resource (i.e., variable) from the subproblems to find a solution. It then sends shadow prices for each resource to each of the subproblems, which find new solutions and update their respective demand estimates. Eventually, the process converges to an optimal solution to the original problem.

These decompositions were originally developed as an efficient way to find solutions to LP problems with certain properties, but they have also been proposed as a natural framework for organizational design [10]. Within the literature on supply chain management, rigorous optimization techniques including decomposition methods have been promoted as a useful tool for analyzing decisions [20]. We note two successful applications of decomposition techniques in supply chain settings. Shapiro and White employed a hybrid price and resource decomposition approach to study the U.S. coal market [21]. Leachman et al. [16] used a heuristic decomposition to formulate an optimization approach for a real semiconductor production system.

Our design for Deep Maize also builds on previous designs for the TAC SCM game. The winner of the first TAC SCM competition in 2003, RedAgent-03 [11, 12], implemented an interesting design that used internal markets to allocate resources (e.g., components, PCs, factory cycles) based on bids by agents representing individual customer orders, PC assemblers, component purchasers, and a seller of production cycles. The clearing prices of these markets can be interpreted as valuations for the resources and used to make external decisions about sales and purchasing. A successful implementation of this approach requires attention to market design and the bidding strategies of the individual agents, since the quality of the values will depend on both of these factors.

Two other agents that make extensive use of optimization techniques to solve portions of the TAC SCM problem are TacTex-03 [18] and Botticelli [2]. TacTex-03 uses a search procedure to optimize the combined customer bidding and factory scheduling problem. It searches over possible sets of customer bids, using a very fast greedy factory scheduler to estimate the profitability of the bids with respect to estimated replacement costs. It accounts for future orders by including predicted customer requests for the next several days in its bid optimization process. The updated version of this agent, TacTex-05, uses similar optimization approaches for decision-making [19], in addition to making predictions of market conditions. Botticelli optimizes a linear programming model of the combined customer bidding and production problems. The Botticelli team has also investigated stochastic programming approaches to model more effectively the uncertainty in the SCM game [3].

There are alternative approaches to TAC SCM that do not rely explicitly on optimization tools. For instance, several agents have taken knowledge-based approaches where developers build up a base of (possibly fuzzy) rules for how to act in various situations. Examples of this approach are PSUTAC [22] and Southampton-SCM [9].

4. DESIGN OVERVIEW

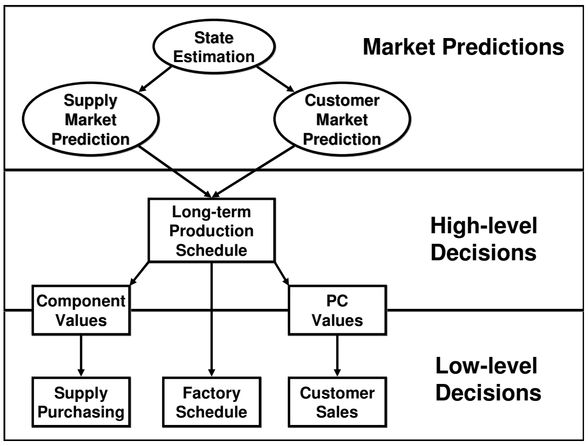

Figure 3: Organization of Deep Maize's decision process on each TAC SCM day.

Deep Maize uses internal estimates of the value of marginal resources to coordinate decisions across all levels of the supply chain and over time. These values provide a natural decomposition of the problem that yields both design modularity and computational benefits. Figure 3 displays a schematic organization of Deep Maize's daily decisions. The agent starts by using any new information it receives to update its beliefs about hidden state in the supply chain. It then makes comprehensive predictions about the conditions it faces in the markets for components and finished PCs, accounting for the behavior of the other manufacturers. These predictions may be based on a variety of data sources, including previous tournament games.

The next stage makes high-level decisions about long-term production scheduling. The *projected production schedule* approximately optimizes the agent's expected profit margin with respect to the market predictions, subject to constraints on factory capacity and component arrival. The schedule is heuristically optimized using a greedy algorithm that orders production according to profit margin per factory cycle. This high-level optimization considers prices in both markets and the important global constraints, but ignores many of the details of actually interacting with suppliers and customers. For example, it does not decide which customer requests to bid on, which suppliers to purchase from, or how far in advance to purchase components. The final stage is a set of optimizations that make specific low-level decisions about customer sales and component purchases. Figure 4 shows the difference in scope between the high- and low-level decisions. The high-level optimization encompasses all of the decisions pictured, out to a horizon many days in the future.[2] In contrast, the shaded boxes represent the low-level sales and purchasing decisions the agent makes each day.

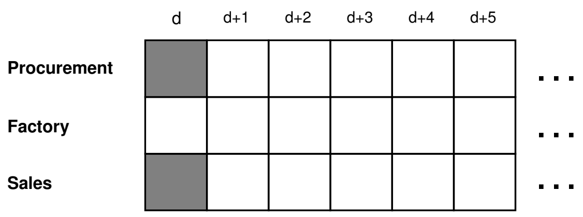

Figure 4: High-level (unshaded) and low-level (shaded) decisions in Deep Maize.

The key question is how to coordinate the low-level decisions with the overall production plan. We accomplish this by deriving values for PCs and components from the production plan and incorporating them into the objective functions for the low-level decisions. Instead of optimizing the overall profit margin, the sales decision optimizes the margin between expected revenue and the value of the PCs sold. Similarly, the purchasing decision optimizes the margin between the value of the components purchased and the total cost. The values condense all of the information contained in the projected production schedule into a form that can be used directly for making low-level decisions.

This design is modular, which offers practical engineering benefits in addition to the computational advantages. The high-level optimization, low-level decisions, and various prediction tasks are all relatively independent, so they can be developed in parallel. The modules are also free to use whatever methods are best suited to the individual structure of the problem, without concern for the specific methods used in the other modules. This is particularly relevant for TAC SCM because the decision structure and available information are very different in the supplier and customer markets.

5. AGENT DESCRIPTION

5.1 Market Predictions

As shown in Figure 3, Deep Maize starts its decision cycle by assessing and forecasting the customer and supplier markets. We sketch the market prediction methods employed for each in turn. Additional details can be found in [17].

For the customer market Deep Maize predicts the *effective demand curve* it will face on each future day. This curve gives the marginal revenue the agent expects to get for each additional PC sold for delivery on the particular day. The exact set of requests is known for the current day, so this amounts to predicting the probability of winning each order given a bid. For future days the curve also reflects predictions about the customer requests that will be generated. Deep Maize estimates the underlying level of demand and the trend using a Bayesian model of the stochastic demand process (see [14]). It employs machine learning techniques based on k-Nearest Neighbors to predict the effective demand curves from historical data, data from self-play games, and information from the current game. The features it uses to find similar situations include indicators of recent market activity, estimates of the underlying customer demand and supplier capacities, and estimates of the aggregate inventory state in the supply chain. The agent scores its predictions during a game and finds affine transformations to correct for systematic errors.

For the supply market Deep Maize predicts the current capacity of each supplier product line and current price each supplier would offer for a component due on any day in the future.[3] The production capacity of each product line follows a mean-reverting random walk; the current capacity is unobservable by the manufacturers. However, agents receive periodic market reports that contain the average capacity for each line over the previous period. Deep Maize adjusts this average to account for the mean-reversion and uses this as a prediction of the capacity of the line. To predict prices the agent uses a weighted average of the prices it was offered over the previous three days. Since the agent can make a limited number of requests each day, the offers do not cover all possible due dates. Between data points the agent predicts using a linear interpolation, and after the last data point prices are decayed to the minimum possible value.

5.2 Projecting Production

Each day Deep Maize projects a future production schedule using a greedy algorithm to quickly approximate an optimal solution. The algorithm orders PC types according to marginal profit per factory cycle. The marginal profit for producing and delivering an additional PC is the difference between the expected *marginal revenue* and *marginal replacement costs*. Both of these factors are updated continually during the scheduling process. The expected marginal revenue R_d for delivering an additional PC of a given type on day d is provided directly by the customer market predictor. For component purchases, we account for the agent's existing inventory commitments in addition to the predicted availability and prices for additional purchases. The replacement cost C_d for a component used for production on day d is defined to be the minimum cost (including holding costs) to acquire an additional component before the day when the projected inventory would fall below a desired buffer level B_d. To determine when a replacement component is needed we maintain a projection of the cumulative number of components expected to be on hand for each future day, denoted I_d. I_d accounts for current inventory, expected future deliveries of components, and components scheduled for production.

To determine the minimum cost to acquire an additional component by day d we compute the *projected purchase price* from raw predictions of supplier prices and holding costs. The supplier price predictor provides estimates of the price to purchase a component from a given supplier on a given day, $P_{d,s}$. The projected price P'_d is the price from the best supplier on the best day, including holding costs:

$$P'_d = \min_{s \in suppliers} \min_{d' \leq d} (P_{d',s} + HC \cdot (d - d')).$$

The daily holding cost HC accounts for both storage costs (exactly) and interest rates (approximately). The exact interest rate

[2]During the tournament, Deep Maize projected the production schedule out to a horizon of 34 days.

[3]It also predicts late deliveries and potential changes in prices, but we do not describe these predictions here.

cost depends on the purchase price, but we simplify using a conservative lower bound. In the case where our supplier predictions indicate that there is very little uncommitted supplier capacity available before the desired day we define $P_d = \infty$ to represent this as a hard constraint.

The replacement cost for a component used on day d, C_d, is defined for three cases:

- If the component needs to be replaced but there is a hard constraint on supplier capacity, the price is infinite. If $I_{d'} \leq 0$ for any day $d' \geq d$ and $P'_{d'} = \infty$, then $C_d = \infty$.
- If the component never needs to be replaced, the replacement cost is the negative of the holding costs until the end of the game. If $I_{d'} > B_{d'}$ for all $d' \geq d$, then $C_d = -HC \cdot (d_{last} - d)$.
- If the component needs to be replaced and can be replaced, the replacement cost is the projected purchase price for the day it is needed, less holding costs until that day.[4] Let $d' \geq d$ be the first day when $I_{d'} \leq B_{d'}$ and $P'_{d'} < \infty$. Then $C_d = P'_{d'} - HC \cdot (d' - d)$.

Deep Maize uses a greedy ordering to schedule production, tracking the factory cycles utilized on each possible production day. On each iteration it selects the delivery day $d_{deliver}$ and PC type with the maximum marginal profit per factory cycle. The marginal profit for a potential delivery day $M_{d_{deliver}}$ is defined to be $-\infty$ if factory cycles are not available to produce the PC in time for delivery. Otherwise, $M_{d_{deliver}}$ is the difference between marginal revenue for the delivery day and the sum of marginal component replacement costs on the latest feasible production day, $d_{produce}$, less holding costs from production to delivery:

$$M_{d_{deliver}} = R_{d_{deliver}} - \sum C_{d_{produce}} - HC \cdot (d_{deliver} - d_{produce}).$$

Once delivery and production days have been selected, Deep Maize updates its state description to prepare for the next iteration. Factory cycles are reserved for manufacturing. The marginal revenue R_d is updated to reflect the additional quantity scheduled to be sold. The inventory trajectory is updated to reflect the components scheduled for manufacturing and any replacement components scheduled for purchase. Component replacement costs are updated to reflect the new inventory trajectory. Deep Maize is now ready to select another PC type and delivery day.

Since marginal revenue is monotone decreasing on each iteration and marginal replacement costs are monotone increasing, the total margin is monotone decreasing on successive iterations. The algorithm terminates when the profit margin per cycle falls below a threshold.

Before initiating the main scheduling algorithm, Deep Maize accounts for outstanding customer orders and current PC inventory. Resources necessary to fill outstanding customer orders are reserved first, using finished PC inventory if possible and factory cycles and components otherwise. The orders are sorted first by due date and then base price per cycle. Scheduling for existing orders first reflects an assumption that it would never be more profitable to leave these orders unfilled in favor of new orders. In TAC SCM this is unlikely because agents must pay relatively high penalties on late or unfilled orders and the additional revenue from a new order is unlikely to outweigh this penalty. Inventory of PCs that is not allocated to fill existing outstanding orders is scheduled for use after the primary scheduling algorithm, using the highest remaining marginal revenues from any future day.[5]

[4]Holding costs are subtracted because they would need to be paid if the current component were held instead of being replaced later.

Deep Maize uses the first day of the projected manufacturing schedule as the manufacturing decisions for the current day. The shipping decision is made separately using simple heuristics. As many outstanding orders as possible are filled using available PC inventory. The ordering is first by due date and then penalty per factory cycle.

5.3 Customer Sales

The customer sales decision for the current day requires the agent to decide which customer requests to bid on and how much to bid on each one. We use a gradient descent search algorithm (with random restarts) to find a set of bids that approximately optimizes the expected value of the resulting set of orders. The expected value of a single order is given by

$$\mathbb{E}[o] = \rho_o(b_o - v_o)q_o,$$

where b_o is the bid level, ρ_o the probability of win given bid b_o, q_o the quantity of PCs demanded, and v_o the value of PCs requested in o. The total expected value over all orders O is

$$\mathbb{E}[O] = \sum_i \mathbb{E}[o_i] = \sum_i \rho_i(b_i - v_i)q_i.$$

The value of PCs term in the customer sales optimization represents the marginal value of the PCs necessary to fill the order, given the projected production schedule and expected orders from the rest of the bid schedule. This value functions as a lower bound on the price the agent will bid on the request.

We approximate the value of PCs as follows. First, note that the projected production schedule explicitly accounts only for future sales, and does not consider potential orders from the current day. Any orders from the current day must either be produced in addition to this production schedule or replace future sales. If the agent produces additional PCs, the effective cost is the replacement cost for the components. If the agent must forgo future sales, the effective cost is the expected revenue from the sale. We define the marginal value of a PC to be the lesser of these two costs. The value of a PC used in the optimization is taken to be the value after scheduling all other PCs in the current set of expected orders. We recompute the values for each day/PC type combination many times during the search process as the current set of expected orders changes.

Information from the projected production schedule is used to determine the costs of additional production and expected revenue from foregone sales. The replacement costs for additional production are determined exactly as they are in the creation of the projected production schedule. The state to perform these calculations (replacement costs, inventory trajectories, factory cycles, etc.) is saved and restored as necessary. To compute the effective cost of replacing future sales we store the marginal revenue from all future sales during the creation of the projected production schedule, as well as the day the PC was produced (or the current day, if the sale is from existing PC inventory). The effective cost is the minimum of these values that meets the constraint that the PC must be produced before the order due date.

5.4 Purchasing Components

[5]This was done to ensure that the lowest values were the most flexible in the PC value calculation. Unfortunately, scheduling these values last ignores the profit margin threshold and may lead to overproduction in some cases. We intend to change how we account for PC inventory in future versions of Deep Maize.

Purchasing components requires action in two stages: RFQ generation and offer acceptance. Both stages attempt to maximize the total value of components ordered less the total purchase price, with component values subject to arrival constraints. We estimate component values from the projected production schedule by storing a value for each replacement component required. This value is determined by dividing the marginal revenue between the necessary components in the same ratio as their expected contribution to the total cost. The values are stored in a structure that encodes the arrival constraints on the values. We can determine the total value for a set of components arriving on different days by finding the maximum set of marginal values that can be achieved without violating arrival constraints. A similar representation was used for procurement in earlier versions of Deep Maize [15].

To generate its set of RFQs, Deep Maize first creates potential RFQs for each future day that maximize the total value of components purchased less the predicted price for the given day. These RFQs are heuristically merged to meet the limit of five requests per supplier per product line. For offer acceptance, the agent determines an initial order set by considering each potential offer individually and ordering those with positive values (or the greatest value, if more than one offer was received for a single request). From this starting point the agent performs a hill-climbing search to improve the value of the order set. This is essentially the same approach employed in prior versions [15].

6. DISCUSSION

The value-based decomposition approach evolved from earlier versions of Deep Maize that used a distributed feedback control mechanism to coordinate decisions between sales and procurement modules [14]. The feedback design defined a reference inventory trajectory based on a desired buffer inventory, current PC orders, and projected future PC orders. Separate sales and procurement modules implemented feedback mechanisms to maintain the actual inventory trajectory within the desired region. The feedback mechanism for the procurement module assigned values to the components in the desired inventory trajectory and optimized purchasing decisions based on these values [15]. The value-based design of Deep Maize 2005 extends this idea to all of the decisions made by the agent. It also replaces the reference trajectory and feedback mechanisms with more precise concepts derived from a principled optimization. The projected production schedule is analogous to the reference inventory trajectory. The PC and component values fill the role of the feedback mechanism; they provide an explicit measure of how much each possible deviation from the reference would cost the agent, allowing for better decision-making over a broader range of conditions.

Our decomposition techniques are related to the price-directive decompositions discussed in Section 3, but it is instructive to consider some ways in which our methods depart from the classic approach. The first is that we do not iterate the process of finding solutions to the individual optimizations to improve the overall results. Our high-level optimization is solved only once each day, using estimates of the outcomes of low-level decisions (i.e., market predictions). We solve the low-level decisions that are needed for taking actions only on the current day. Neglecting to consider future decisions at the detailed level represents a very large simplification of the actual problem. However, we expect that the benefits of explicitly optimizing the future subproblems and iterating the solution procedure would be relatively low due to the significant uncertainty present in projections of future market conditions.

Another departure is that the values we use are not shadow prices and do not have the same theoretical guarantees. Part of the issue here is that our formulation is an integer programming problem, and the concept of shadow prices from linear programming does not translate easily to integer programming [8]. However, we also introduce a further approximation by defining the values based on marginal values with respect to incomplete production schedules. This definition leaves us with values that are most meaningful for small deviations from the final projected production schedule, which is exactly where we expect most of the decisions to be made. Intuitively, some decisions are obvious and do not need careful consideration, whereas the decisions on the margin (close variants of the final projected schedule) are more difficult.

These departures represent potential ways to improve our methods, if we are willing to pay the additional costs of model complexity and computation time. For instance, we could likely improve our solutions by introducing some form of iteration between the high- and low-level solutions. There has also been recent work investigating new pricing concepts for integer programs that have many of the same properties as shadow price in linear programs (for example, average shadow prices [5]). Using one of these pricing concepts could potential improve our value estimates. Finally, we directly solve the nonlinear optimization problem using a heuristic optimization. An alternative approach that would allow application of linear programming solution techniques is to approximate the objective function with a linearizion (this is the approach taken by Botticelli [2]).

7. EVALUATION

Evaluating trading agent designs in TAC SCM is challenging because there are many factors that contribute to the performance of each agent. We must also rely on comparisons with other agents as benchmarks in the absence of a known optimal strategy. The primary measure agent performance is average profit. However, we can improve our understanding of more specific differences between the agents by considering additional metrics. Here we focus on measures that relate to to central feature of our design, the ability to coordinate decisions through resource valuations. We consider only data from the tournament, but note that the addition of a repository of agent binaries offers interesting opportunities for future controlled studies [24].

7.1 Average Profits

The TAC SCM tournament is played in a series of rounds. Qualifying and seeding rounds (used for debugging) last two weeks each. The final three rounds were held on three successive days in conjunction with IJCAI-05. The qualifying round started 32 teams, and 24 qualified for the quarter-finals. Agents played in heats of 6, with the bottom three from each heat eliminated in each round (leaving 12 for the semi-finals and 6 for the finals). The quarter and semi-final rounds had 8 games for each heat; the finals had 16 games.

Deep Maize advanced to the final round. The results from each heat it competed in are in Table 1. In the quarter-final heat one agent did not participate in any games and another missed two games. This distorts the game significantly, so we omit further analysis of this round. In the final round, the University of Michigan experienced network problems that caused Deep Maize to miss a substantial part of two games.[6] The "adjusted" column for the finals shows the scores and standings recomputed without these two games. All of the additional analysis of the finals presented in this section is done excluding these two games, since they were flawed for known reasons unrelated to agent strategy.

[6]Games number 3718 and 4259.

Table 1: Scores and rankings from the rounds Deep Maize played in during the 2005 SCM tournament. All scores are in millions of dollars.

Quarter-Finals D		Semi-Finals 2		Finals		Finals (adjusted)	
Deep Maize	17.49	Deep Maize	3.68	TacTex-05	4.74	TacTex-05	5.18
CMieux	15.03	TacTex-05	3.57	SouthamptonSCM	1.60	Deep Maize	2.06
RationalSCM	14.61	MinneTAC	2.27	Mertacor	0.55	SouthamptonSCM	1.55
CrocodileAgent	11.64	RationalSCM	-2.28	Deep Maize	-0.22	Mertacor	0.53
Cylinder	5.79	CMieux	-2.33	MinneTAC	-0.31	MinneTAC	0.33
optimiSCM	0	PhantAgent	-6.64	Maxon	-1.99	Maxon	-1.74

Deep Maize performed very well overall. It made the final round and placed fourth overall, out of 32 original teams. Unofficially, Deep Maize did even better and placed second, disregarding games with network problems. Deep Maize also demonstrated strong performance in earlier rounds, placing first in both of them. This provides some evidence that Deep Maize performs well in a variety of competitive environments.

7.2 Inventory Management

If the sales and purchasing strategies are not well coordinated inventory can quickly become unbalanced, leading to over- or undersupply of certain components. We look at three specific inventory management issues. The first is how well agents sold out inventory at the end of the game. Unsold inventory is worthless, and large unsold inventories are an indication of overstocking. We also look at the storage cost incurred by the agents, which correspond exactly to the average daily value of inventory held during the game. Finally, we look for instances where agents had very low inventory of a component. These are undesirable because they may lead to late deliveries or missed sales opportunities.

Tables 2 and 3 give the average value (base price) of unsold inventory for each agent in the semi-final and final rounds, respectively. Average revenue and material cost figures for each agent are also provided for calibration. Deep Maize had the lowest levels of unsold inventory in both rounds; TacTex-05 was also very good at selling off inventory. The tables also give the average storage costs paid by each agent. It is desirable to minimize these costs, but lower levels of inventory carry an increased risk that the agent will run out of components or need to pay high prices to acquire components that are in short supply. Deep Maize attempts to maintain a fairly large buffer inventory during the game, and we have not carefully tuned the size of this buffer to balance the risks against the storage costs. Deep Maize paid storage costs that were on the high side, but not excessive. TacTex-05 and MinneTAC maintained noticeably lower inventories than the other finalists.

Finally, we consider instances where agents had very low inventory of a component during the middle portion of the game. We count the number of instances between days 20 and 200 where each agents had less than 100 total components of any type in stock (including components PC inventory). We exclude the start and end of the game because these are regions where agents will naturally have instances of very low inventory. Deep Maize had the lowest number of instances of low inventory in both rounds, with Maxon and TacTex-05 very close in the finals.

7.3 Market Selection

Agents should direct their production activities to more profitable markets (i.e., to PC types with higher margins). Determining which markets are most profitable requires the agent to synthesize information about prices in both the component and PC markets. Altering the distribution of PCs produced requires significant coordination between the purchasing and sales decisions; adjusting customer prices to alter the sales distribution does not make sense unless the agent also purchases the right combination of extra components (at relatively high prices, if need be).

To measure how well agents were able to adjust their product mix to more profitable activities we looked at the correlation between margins in a particular PC market and the fraction of an agent's sales devoted to the market. We calculate margins from average selling prices (ASPs) over 5-day periods in the customer and supplier markets. Since supply purchases typically lead customer sales, we also consider a second measure of the margin that uses the supplier ASPs over a 20 day period up to and including the 5-day customer sales period. We calculate the fraction of an agent's total sales (by quantity) that are made in the given market over the same 5-day period. The distribution question is particularly interesting when agents are constrained by limited factory cycles since there are stronger interactions between PC types in this case. Component availability can also be a constraint, but our experience suggests that this is less common (the data about instances of low inventory presented in section 7.2 support this). To emphasize the effect of factory cycles, we also present the results for the margin per factory cycle.

Tables 4 and 5 show the correlations between production fraction and the different measures of profit margin for each of the agents in the semi-final and final rounds. In computing these correlations, we disregard any instances where no PCs of any type were sold by the agent over the 5-day period. Higher correlations correspond to more PCs being sold in markets with higher margins, and are therefore desirable. There is a natural negative correlation due to the demand curve; as an agent sells larger quantities customer prices fall and supplier prices rise, decreasing the margin. In both rounds, Deep Maize had the highest correlation, regardless of the particular margin calculation. Deep Maize also appears to have the strongest bias towards per cycle margins over of raw margins. This is likely a result of using the margin per cycle as a heuristic for the greedy optimization algorithm instead of the raw margin. In addition to the production fraction results presented here we also looked at correlations with raw quantities and market share. The general pattern of results is the same in both cases.

7.4 Market Timing

Another interesting aspect of agent performance is market timing. Ideally, agents should buy, sell, and produce at the most profitable times during the game. For instance, if customer market prices are currently high and predicted to weaken, the agent should sell any excess inventory quickly to take advantage of the current market. Conversely, improving prices may make it advantageous to build extra inventory. We develop a measure of agent efficiency in timing customer sales, holding the agent's purchasing and production activities fixed. The measure compares the agent's achieved ASPs with the optimal ASPs the agent could have achieved with

Table 2: Statistics from 2005 tournament, semi-final round heat 1. Unsold is the average total base price of inventory unsold at the end of the game, and low inventory is the average number of instances where inventory for a component was less that 100 during the middle 180 days of the simulation (out of 1800 possible). All numbers except low inventory are in millions of dollars.

Agent	Revenue	Material	Storage	Unsold	Low Inv.
Deep Maize	110.4	102.7	2.38	0.81	25
TacTex-05	96.3	89.8	1.59	1.56	76
MinneTAC	77.7	73.3	1.59	2.86	224
RationalSCM	78.8	77.1	2.68	1.48	77
CMieux	93.6	91.5	2.98	2.64	211
PhantAgent	65.4	67.6	3.22	5.06	62

Table 3: Statistics from 2005 tournament, final round. Unsold is the average total base price of inventory unsold at the end of the game, and low inv is the number of instances where inventory for a component was less that 100 during the middle 180 days of the simulation (out of 1800 possible). All numbers except low inventory are in millions of dollars.

Agent	Revenue	Material	Storage	Unsold	Low Inv.
TacTex-05	109.3	100.8	2.08	1.00	40
Deep Maize	110.3	103.5	2.82	0.82	24
SouthamptonSCM	108.9	103.0	2.91	4.16	177
Mertacor	74.2	71.2	1.97	2.67	242
MinneTAC	83.2	80.4	1.76	5.18	121
Maxon	68.8	68.7	3.53	3.84	25

Table 4: Correlations between prevailing margins for a particular PC type and the fraction of each agent's sales devoted to the market. Semi-Final round, heat 1.

Agent	Margin (5)	Margin/cycles (5)	Margin (20)	Margin/cycles (20)
Deep Maize	-0.056	-0.013	-0.059	-0.008
TacTex-05	-0.160	-0.147	-0.156	-0.141
MinneTAC	-0.111	-0.134	-0.099	-0.094
RationalSCM	-0.096	-0.091	-0.099	-0.094
CMieux	-0.175	-0.159	-0.196	-0.178
PhantAgent	-0.125	-0.113	-0.104	-0.090

Table 5: Correlations between prevailing margins for a particular PC type and the fraction of each agent's sales devoted to the market. Final round.

Agent	Margin (5)	Margin/cycles (5)	Margin (20)	Margin/cycles (20)
TacTex-05	-0.118	-0.112	-0.121	-0.111
Deep Maize	-0.091	-0.059	-0.105	-0.067
SouthamptonSCM	-0.120	-0.125	-0.138	-0.141
Mertacor	-0.168	-0.159	-0.162	-0.148
MinneTAC	-0.116	-0.129	-0.142	-0.156
Maxon	-0.101	-0.077	-0.124	-0.095

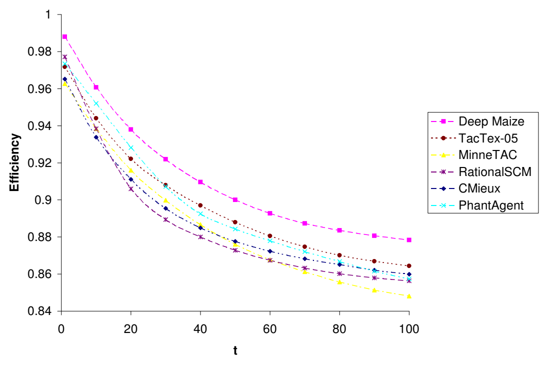

Figure 5: Average market timing efficiencies for the semi-final round. The x-axis represents varying degrees of freedom to change the timing of sales.

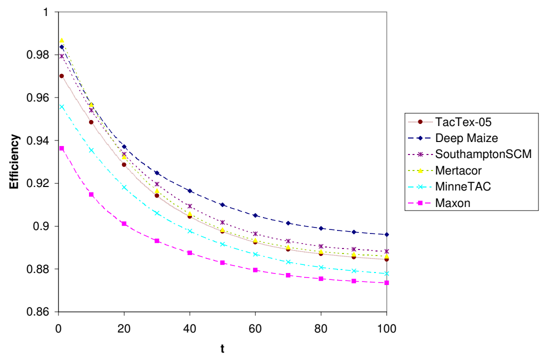

Figure 6: Average market timing efficiencies for the final round. The x-axis represents varying degrees of freedom to change the timing of sales.

perfect information of market prices, under varying degrees of flexibility to alter the sales schedule.

Define *optimal same day selling price* (OSDSP) to be the highest ASP an agent could have achieved by selling the same number of PCs for each due date as it actually sold in a game, but possibly for higher prices or in response to different requests with the same due date. We allow partial fulfillment of customer requests so that it is possible for agents to switch to different requests without violating the constraint on the quantity sold. Next, we define the *optimal t-variable day selling price* (t-OVDSP) to be the highest ASP the agent could have achieved selling each PC for delivery at most t days after the day it was actually delivered in the game and no earlier than it could have been delivered under the actual production schedule. PCs are labeled according to a FIFO queuing policy. For simplicity, we also disregard changes in the storage and interest costs from changing the sales schedule (the effects are typically small). Finally, we define the *t-day timing efficiency* to be the ratio of OSDSP and t-OVDSP. Normalizing by the same-day optimal value factors out the effects of bidding policy (pricing and RFQ selection), leaving the effects of deciding which day to sell on. The parameter t allows us to investigate a range of schedules with different levels of flexibility.

Figures 5 and 6 show the results of applying this measure to the semi-final and final round data. Deep Maize performs well on this measure of market timing in both rounds, and particularly for larger values of t. This is encouraging, but we must be careful not to draw too much from these comparisons. In particular, fixing the production and purchasing decisions introduces a bias into the measure; we would expect some methods of production to inherently allow more flexibility in sales than others (e.g. a strict make-to-stock strategy versus a make-to-order strategy). This bias is not as problematic when comparing different versions of the same agent with identical purchasing and production strategies, or when using the timing efficiency to look for specific instances where an agent was selling too early or too late. In the future we hope to augment this measure with additional measures that allow modifications to the production and purchasing schedules; this should provide a more complete means of assessing agent performance in timing markets.

7.5 Simulation Results

We present some final evidence that our decision-making architecture has the ability to exploit better predictions about market conditions. Table 6 gives simulation results for different versions of Deep Maize that use less accurate predictors. All results are based on approximately 30 simulation games, and contain two instances of TacTex-05, two instances of MinneTAC, and two different versions of Deep Maize. One version of Deep Maize uses the standard predictor, and the other uses the variation listed in the table. Results are presented as the difference in the scores between the two variations of Deep Maize. The "Noisy" predictors add uniform random noise to the base predictions. The supplier noise is an additive fraction of the base price, while the customer noise is an exponential mapping of the predicted distribution. This results in approximately a 10% distortion in predictions for the setting of 25. The details of these predictors are not crucial; the important point we wish to make is that reductions in prediction accuracy have a substantial negative impact on Deep Maize's overall performance. More detailed analysis can be found in [17].

8. CONCLUSION

We have presented and evaluated the design of our entry in the 2005 TAC SCM competition, Deep Maize. Our design is based on a unified approach to coordinating decisions across the supply

Table 6: Simulation results using different predictors.

Predictor	Score Difference
Heuristic Customer	-9.1 M
Noisy (supplier 0, customer 25)	-1.8 M
Noisy (supplier 0, customer 50)	-4.5 M
Noisy (supplier 10, customer 0)	-0.5 M
Noisy (supplier 10, customer 25)	-1.8 M
Noisy (supplier 20, customer 0)	-1.9 M
Noisy (supllier 20, customer 25)	-4.3 M

chain using estimated resource valuations. These valuations allow us to decompose the problem into manageable subproblems, while retaining the ability to do high-level planning. The approach is grounded in similar theoretical decompositions used to solve linear programs, but the demands of the TAC SCM scenario force us to use approximations to achieve computationally feasible models.

Deep Maize performed very well in the TAC SCM tournament, making the final round and placing 4th overall (2nd if we disregard two games with network connectivity problems). We also considered several additional measures of agent performance that relate more specifically to the ability of agents to coordinate decisions across markets over time. Deep Maize performed strongly on all of the measures we considered. We believe that estimating resource values and using these values to coordinate decisions is a powerful design paradigm for building agents that act in complex environments. Our success with Deep Maize shows that the values do not need to be perfect to transmit useful information; even approximations exhibit strong performance compared to other methods.

9. ACKNOWLEDGMENTS

We would like to thank the TAC SCM organizers for their efforts in running a successful tournament, and the SCM participants for providing a strong field of competitors. We would also like to thank Kevin O'Malley for his role in the development of Deep Maize. This research was supported in part by the U.S. National Science Foundation under IGERT grant 0114368 and grant IIS-0205435, and also by the DARPA REAL strategic reasoning program.

10. REFERENCES

[1] R. Arunachalam and N. M. Sadeh. The supply chain trading agent competition. *Electronic Commerce Research and Applications*, 4:63–81, 2005.

[2] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. Tschantz. Botticelli: A supply chain management agent. In *Third International Conference on Autonomous Agents and Multiagent Systems*, pages 1174–1181, New York, 2004.

[3] M. Benisch, A. Greenwald, V. Naroditskiy, and M. Tschantz. A stochastic programming approach to scheduling in TAC SCM. In *Fifth ACM Conference on Electronic Commerce*, pages 152–159, New York, 2004.

[4] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Janson. The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-04-139, Carnegie Mellon University, 2004.

[5] A. Crema. Average shadow price in a mixed integer linear programming problem. *European Journal of Operational Research*, 85:625–635, 1995.

[6] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[7] J. Eriksson, N. Finne, and S. Janson. Evolution of a supply chain management game for the Trading Agent Competition. *AI Communications*, 9:1–12, 2006.

[8] R. E. Gomory and W. J. Baumol. Integer programming and pricing. *Econometrica*, 28(3):521–550, 1960.

[9] M. He, A. Rogers, E. David, and N. R. Jennings. Designing and evaluating an adaptive trading agent for supply chain management applications. In *IJCAI-05 Workshop on Trading Agent Design and Analysis*, 2005.

[10] K. Holmberg. Primal and dual decomposition as organizational design: price and/or resource directive decomposition. Technical report, Linkoping Institute of Technology, Sweden, 1996.

[11] P. Keller, F.-O. Duguay, and D. Precup. RedAgent-2003: An autonomous market-based supply-chain management agent. In *Third International Conference on Autonomous Agents and Multiagent Systems*, pages 1182–1189, New York, 2004.

[12] P. W. Keller, F.-O. Duguay, and D. Precup. RedAgent: Winner of TAC SCM 2003. *SIGecom Exchanges*, 4(3):1–8, 2004.

[13] W. Ketter, J. Collins, M. Gini, A. Gupta, and P. Schrater. Identifying and forecasting economic regimes in TAC SCM. In *IJCAI-05 Workshop on Trading Agent Design and Analysis*, 2005.

[14] C. Kiekintveld, M. P. Wellman, S. Singh, J. Estelle, Y. Vorobeychik, V. Soni, and M. Rudary. Distributed feedback control for decision making on supply chains. In *Fourteenth International Conference on Automated Planning and Scheduling*, pages 384–392, Whistler, BC, 2004.

[15] C. Kiekintveld, M. P. Wellman, S. Singh, and V. Soni. Value-driven procurement in the TAC supply chain game. *SIGecom Exchanges*, 4(3):9–18, 2004.

[16] R. C. Leachman, R. F. Benson, C. Liu, and D. J. Raar. IMPReSS: An automated production planning and delivery quotation system at Harris Corporation – semicondutor sector. *Interfaces*, 26:6–37, 1996.

[17] J. Miller, C. Kiekintveld, P. R. Jordan, and M. P. Wellman. Forecasting market prices in a supply chain game. Technical report, University of Michigan, 2006.

[18] D. Pardoe and P. Stone. TacTex-03: A supply chain management agent. *SIGecom Exchanges*, 4(3):19–28, 2004.

[19] D. Pardoe and P. Stone. Predictive planning for supply chain management. In *Proceedings of the International Conference on Automated Planning and Scheduling*, June 2006.

[20] J. F. Shapiro. *Modeling the Supply Chain*. Duxbury, 2001.

[21] J. F. Shapiro and D. E. White. A hybrid decomposition method for integrating coal supply and demand models. *Operations Research*, 30(5):887–906, 1982.

[22] S. Sun, V. Avasarala, T. Mullen, and J. Yen. PSUTAC: A trading agent designed from heuristics to knowledge. In *AAMAS-04 Workshop on Trading Agent Design and Analysis*, 2004.

[23] M. P. Wellman, J. Estelle, S. Singh, Y. Vorobeychik, C. Kiekintveld, and V. Soni. Strategic interactions in a supply chain game. *Computational Intelligence*, 21:1–26, 2005.

[24] M. P. Wellman, P. R. Jordan, C. Kiekintveld, J. Miller, and D. M. Reeves. Empirical game-theoretic analysis of the TAC market games. In *AAMAS-06 Workshop on Game-Theoretic and Decision-Theoretic Agents*, 2006.