To appear in Gedas Adomavicius and Alok Gupta, editors, *Handbooks in Information Systems Series: Business Computing*, Elsevier, 2007.

An Autonomous Agent for Supply Chain Management

David Pardoe, Peter Stone

Department of Computer Sciences

The University of Texas at Austin

Abstract

Supply Chain Management involves planning for the procurement of materials, assembly of finished products from these materials, and distribution of products to customers. The Trading Agent Competition Supply Chain Management scenario (TAC SCM) provides a competitive benchmarking environment for developing and testing agent-based solutions to supply chain management. Autonomous software agents must perform the above tasks while competing against each other as computer manufacturers: each agent must purchase components such as memory and hard drives from suppliers, manage a factory where computers are assembled, and negotiate with customers to sell computers. In this chapter, we describe TacTex-06, the winning agent in the 2006 TAC SCM competition. TacTex-06 operates by making predictions about the future of the economy, such as the prices that will be offered by component suppliers and the level of customer demand, and then planning its future actions in order to maximize profits. A key component of TacTex-06 is the ability to adapt these predictions based on the observed behavior of other agents. Although the agent is described in full, particular emphasis is given to agent components that differ from the previous year's winner, TacTex-05, and the importance of these components is demonstrated through controlled experiments.

1 Introduction

In today's industrial world, supply chains are ubiquitous in the manufacturing of many complex products. Traditionally, supply chains have been created through the interactions of human representatives of the various companies involved. However, recent advances in autonomous agent technologies have

Email addresses: dpardoe@cs.utexas.edu (David Pardoe), pstone@cs.utexas.edu (Peter Stone).

Preprint submitted to Elsevier Science *28 May 2007*

sparked an interest, both in academia and in industry, in automating the process (Kumar, 2001) (Sadeh *et al.*, 2001) (Chen *et al.*, 1999). Creating a fully autonomous agent for supply chain management is difficult due to the large number of tasks such an agent must perform. In general, the agent must procure resources for, manage the assembly of, and negotiate the sale of a completed product. To perform these tasks intelligently, the agent must be able to plan in the face of uncertainty, schedule the optimal use of its resources, and adapt to changing market conditions.

One barrier to supply chain management research is that it can be difficult to benchmark automated strategies in a live business environment, both due to the proprietary nature of the systems and due to the high cost of errors. The Trading Agent Competition Supply Chain Management (TAC SCM) scenario provides a unique testbed for studying and prototyping supply chain management agents by providing a competitive environment in which independently created agents can be tested against each other over the course of many simulations in an open academic setting. A particularly appealing feature of TAC is that, unlike in many simulation environments, the other agents are real profit-maximizing agents with incentive to perform well, rather than strawman benchmarks.

In a TAC SCM game, each agent acts as an independent computer manufacturer in a simulated economy. The agent must procure components such as CPUs and memory; decide what types of computers to manufacture from these components as constrained by its factory resources; bid for sales contracts with customers; and decide which computers to deliver to whom and by when.

In this chapter, we describe TacTex-06, the winner of the 2006 TAC SCM competition. In particular, we describe the various components that make up the agent and discuss how they are combined to result in an effective supply chain management agent. Emphasis is given to those components that differ from the previous year's winner, TacTex-05, and the importance of these components is demonstrated through controlled experiments. The remainder of the chapter is organized as follows. We first summarize the TAC SCM scenario, and then give an overview of the design of TacTex-06. Next, we describe in detail the individual components: three predictive modules, two decision-making modules that attempt to identify optimal behavior with respect to the predictions, and two methods of adapting to opponent behavior based on past games. Finally, we examine the success of the complete agent, through both analysis of competition results and controlled experiments.

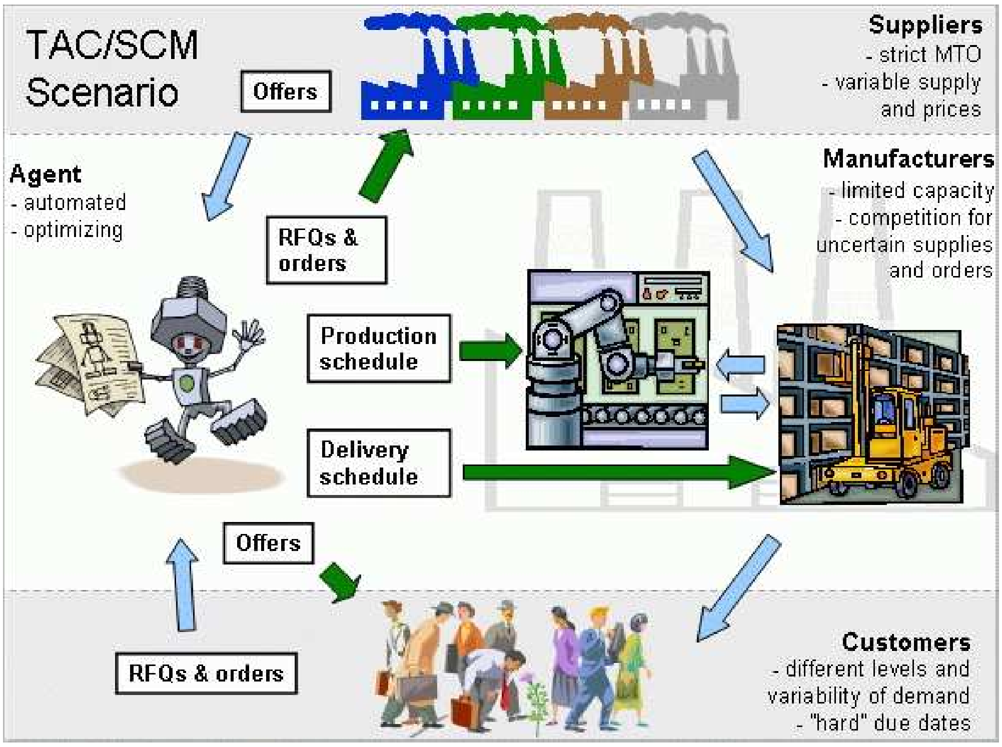

Fig. 1. The TAC SCM Scenario (Collins *et al.*, 2005).

2 The TAC Supply Chain Management Scenario

In this section, we provide a summary of the TAC SCM scenario. Full details are available in the official specification document (Collins *et al.*, 2005).

In a TAC SCM game, six agents act as computer manufacturers in a simulated economy that is managed by a game server. The length of a game is 220 simulated days, with each day lasting 15 seconds of real time. At the beginning of each day, agents receive messages from the game server with information concerning the state of the game, such as the customer requests for quotes (RFQs) for that day, and agents have until the end of the day to send messages to the server indicating their actions for that day, such as making offers to customers. The game can be divided into three parts: i) component procurement, ii) computer sales, and iii) production and delivery as expanded on below and illustrated in Figure 1.

2.1 Component Procurement

The computers are made from four components: CPUs, motherboards, memory, and hard drives, each of which come in multiple varieties. From these components, 16 different computer configurations can be made. Each component has a *base price* that is used as a reference point by suppliers making offers.

Agents wanting to purchase components send requests for quotes (RFQs) to suppliers indicating the *type and quantity* of components desired, the *date* on which they should be delivered, and a *reserve price* stating the maximum amount the agent is willing to pay. Agents are limited to sending at most 5 RFQs per component per supplier per day. Suppliers respond to RFQs the next day by offering a price for the requested components if the request can be satisfied. Agents may then accept or reject the offers.

Suppliers have a *limited capacity* for producing components, and this capacity varies throughout the game according to a random walk. Suppliers base their prices offered in response to RFQs on the fraction of their capacity that is currently free. When determining prices for RFQs for a particular component, a supplier simulates scheduling the production of all components currently ordered plus those components requested in the RFQs as late as possible. From the production schedule, the supplier can determine the remaining free capacity between the current day and any future day. The price offered in response to an RFQ is equal to the base price of the component discounted by an amount proportional to the fraction of the supplier's capacity free before the due date. Agents may send zero-quantity RFQs to serve as price probes. Due to the nature of the supplier pricing model, it is possible for prices to be as low when components are requested at the last minute as when they are requested well in advance. Agents thus face an interesting *tradeoff*: they may either commit to ordering while knowledge of future customer demand is still limited (see below), or wait to order and risk being unable to purchase needed components.

To prevent agents from driving up prices by sending RFQs with no intention of buying, each supplier keeps track of a *reputation* rating for each agent that represents the fraction of offered components that have been accepted by the agent. If this reputation falls below a minimum acceptable purchase ratio (75% for CPU suppliers, and 45% for others), then the prices and availability of components are affected for that agent. Agents must therefore plan component purchases carefully, sending RFQs only when they believe it is likely that they will accept the offers received.

2.2 Computer Sales

Customers wishing to buy computers send the agents RFQs consisting of the *type and quantity* of computer desired, the *due date*, a *reserve price* indicating the maximum amount the customer is willing to pay per computer, and a *penalty* that must be paid for each day the delivery is late. Agents respond to the RFQs by bidding in a first-price auction: the agent offering the lowest price on each RFQ wins the order. Agents are unable to see the prices offered by other agents or even the winning prices, but they do receive a report each day indicating the highest and lowest price at which each type of computer sold on the previous day.

Each RFQ is for between 1 and 20 computers, with due dates ranging from 3 to 12 days in the future, and reserve prices ranging from 75% to 125% of the base price of the requested computer type. (The base price of a computer is equal to the sum of the base prices of its parts.)

The number of RFQs sent by customers each day depends on the level of customer demand, which fluctuates throughout the game. Demand is broken into three segments, each containing about one third of the 16 computer types: high, mid, and low range. Each range has its own level of demand. The total number of RFQs per day ranges between roughly 80 and 320, all of which can be bid upon by all six agents. It is *possible for demand levels to change rapidly*, limiting the ability of agents to plan for the future with confidence.

2.3 Production and Delivery

Each agent manages a factory where computers are assembled. Factory operation is constrained by both the components in inventory and assembly cycles. Factories are limited to producing roughly 360 computers per day (depending on their types). Each day an agent must send a production schedule and a delivery schedule to the server indicating its actions for the next day. The production schedule specifies how many of each computer will be assembled by the factory, while the delivery schedule indicates which customer orders will be filled from the completed computers in inventory. Agents are required to pay a small daily storage fee for all components in inventory at the factory. This cost is sufficiently high to discourage agents from holding large inventories of components for long periods.

3 Overview of TacTex-06

Given the detail and complexity of the TAC SCM scenario, creating an effective agent requires the development of tightly coupled modules for interacting with suppliers, customers, and the factory. The fact that each day's decisions must be made in less than 15 seconds constrains the set of possible approaches.

TacTex-06 is a fully implemented agent that operates within the TAC SCM scenario. We present a high-level overview of the agent in this section, and full details in the sections that follow.

3.1 Agent Components

Figure 2 illustrates the basic components of TacTex-06 and their interaction. There are five basic tasks a TAC SCM agent must perform:

(1) Sending RFQs to suppliers to request components;
(2) Deciding which offers from suppliers to accept;
(3) Bidding on RFQs from customers requesting computers;
(4) Sending the daily production schedule to the factory;
(5) Delivering completed computers.

We assign the first two tasks to a *Supply Manager* module, and the last three to a *Demand Manager* module. The Supply Manager handles all planning related to component inventories and purchases, and requires no information about computer production except for a projection of future component use, which is provided by the Demand Manager. The Demand Manager, in turn, handles all planning related to computer sales and production. The only information about components required by the Demand Manager is a projection of the current inventory and future component deliveries, along with an estimated replacement cost for each component used. This information is provided by the Supply Manager.

We view the tasks to be performed by these two managers as optimization tasks: the Supply Manager tries to minimize the cost of obtaining the components required by the Demand Manager, while the Demand Manager seeks to maximize the profits from computer sales subject to the information provided by the Supply Manager. In order to perform these tasks, the two managers need to be able to make predictions about the results of their actions and the future of the economy. TacTex-06 uses three predictive models to assist the managers with these predictions: a predictive *Supplier Model*, a predictive *Demand Model*, and an *Offer Acceptance Predictor*.

The Supplier Model keeps track of all information available about each supplier, such as TacTex-06 's outstanding orders and the prices that have been offered in response to RFQs. Using this information, the Supplier Model can assist the Supply Manager by making predictions concerning future component availability and prices.

The Demand Model tracks the customer demand in each of the three market segments, and tries to estimate the underlying demand parameters in each

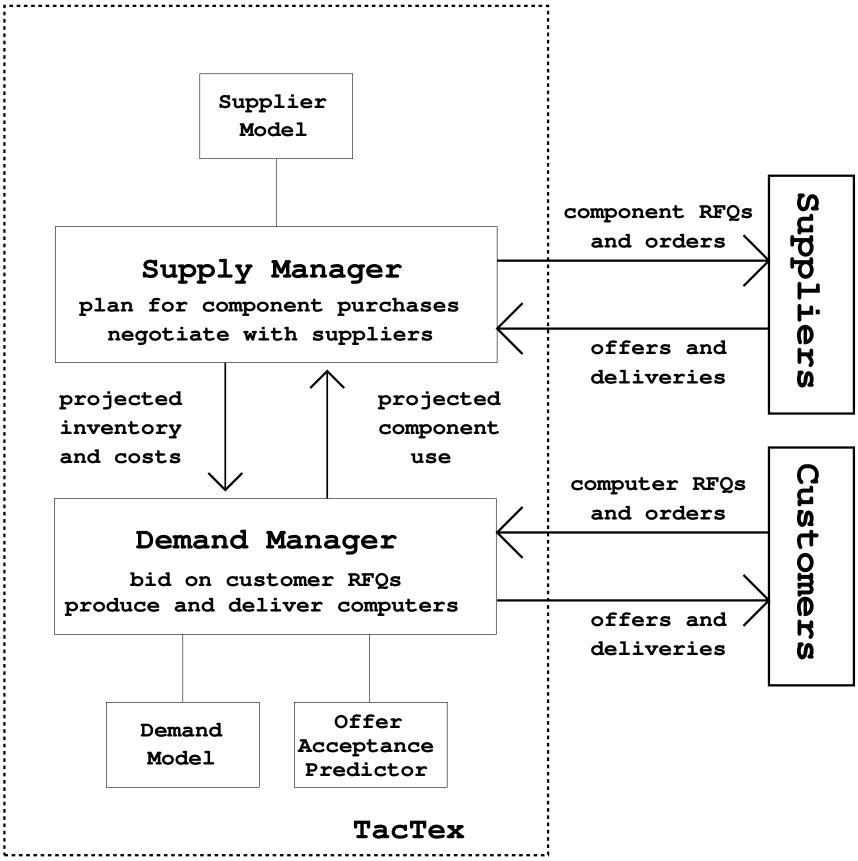

Fig. 2. An overview of the main agent components

segment. With these estimates, it is possible to predict the number of RFQs that will be received on any future day. The Demand Manager can then use these predictions to plan for future production.

When deciding what bids to make in response to customer RFQs, the Demand Manager needs to be able to estimate the probability of a particular bid being accepted (which depends on the bidding behavior of the other agents). This prediction is handled by the Offer Acceptance Predictor. Based on past bidding results, the Offer Acceptance Predictor produces a function for each RFQ that maps bid prices to the predicted probability of winning the order.

The steps taken each day by TacTex-06 as it performs the five tasks described previously are presented in Table 1.

4 The Demand Manager

The Demand Manager handles all computation related to computer sales and production. This section describes the Demand Manager, along with the De-

- Record information received from the server and update prediction modules
- The Supply Manager takes the supplier offers as input and performs the following:
  · decide which offers to accept
  · update projected future inventory
  · update replacement costs
- The Demand Manager takes customer RFQs, current orders, projected inventory, and replacement costs as input and performs the following:
  · predict future customer demand using the Demand Model
  · use the Offer Acceptance Predictor to generate acceptance functions for RFQs
  · schedule production several days into the future
  · extract the current day's production, delivery, and bids from the schedule
  · update projected future component use
- The Supply Manager takes the projected future component use as input and performs the following:
  · determine the future deliveries needed to maintain a threshold inventory
  · use the Supplier Model to predict future component prices
  · decide what RFQs need to be sent on the current day

Table 1
Overview of the steps taken each day by TacTex-06.

mand Predictor and the Offer Acceptance Predictor upon which it relies.

4.1 Demand Model

When planning for future computer production, the Demand Manager needs to be able to make predictions about future demand in each market segment. For example, if more RFQs are expected for high range than low range computers, the planned production should reflect this fact. The Demand Model is responsible for making these predictions.

In order to explain its operation, further detail is required about the customer demand model. The state of each demand segment (high, mid, and low range computers) is represented by parameters Q_d and τ_d (both of which are internal to the game server). Q_d represents the expected number of RFQs on day d, and τ_d is the trend in demand (increasing or decreasing) on day d. The actual number of RFQs is generated randomly from a Poisson distribution with Q_d as its mean. The next day's demand, Q_{d+1}, is set to $Q_d\tau_d$, and τ_{d+1} is determined from τ_d according to a random walk.

To predict future demand, the Demand Manager estimates the values of Q_d and τ_d for each segment using an approach first used by the agent DeepMaize in 2003 (Kiekintveld *et al.*, 2004). Basically, this is a Bayesian approach that involves maintaining a probability distribution over (Q_d, τ_d) pairs for each segment. The number of RFQs received each day from the segment represents information that can be used to update this distribution, and the distribution over (Q_{d+1}, τ_{d+1}) pairs can then be generated based on the game's demand model. By repeating this last step, the expected value of Q_i can be determined for any future day i and used as the number of RFQs predicted on that day. Full details of the approach are available in (Kiekintveld *et al.*, 2004). [1]

4.2 Offer Acceptance Predictor[2]

In order to bid on customer RFQs, the Demand Manager needs to be able to predict the orders that will result from the offers it makes. A simple method of prediction would be to estimate the winning price for each RFQ, and assume that any bid below this price would result in an order. Alternatively, for each RFQ the probability of winning the order could be estimated as a function of the current bid. This latter approach is the one implemented by the Offer Acceptance Predictor. For each customer RFQ received, the Offer Acceptance Predictor generates a function mapping the possible bid prices to the probability of acceptance. (The function can thus be viewed as a cumulative distribution function.) This approach involves three components: a particle filter used to generate initial predictions, an adaptive means of revising the predictions to account for the impact of an RFQ's due date, and a learned predictor that predicts how the prices of computers will change in the future.

A visual inspection of each day's winning prices for each type of computer in a typical completed game suggests that these prices tend to follow a normal distribution. To estimate these distributions during a game, the Offer Acceptance Predictor makes use of a separate particle filter (specifically a Sampling Importance Resampling filter (Arulampalam *et al.*, 2002)) for each computer type. A particle filter is a sequential Monte Carlo method that tracks the changing state of a system by using a set of weighted samples (called particles) to estimate a posterior density function over the possible states. The weight of each particle represents its relative probability, and particles and weights are revised each time an observation (conditioned on the current state) is received. In this case, each of the 100 particles used per filter represents a normal distribution (indicating the probability that a given price will be the winning price on the computer) with a particular mean and variance. At the beginning of each game, weights are set equally and each particle is assigned a mean and

[1] The DeepMaize team has released their code for this approach: `http://www.eecs.umich.edu/~ckiekint/downloads/DeepMaize_CustomerDemand_Release.tar.gz`

[2] This section presents a significant addition to the previous agent, TacTex-05

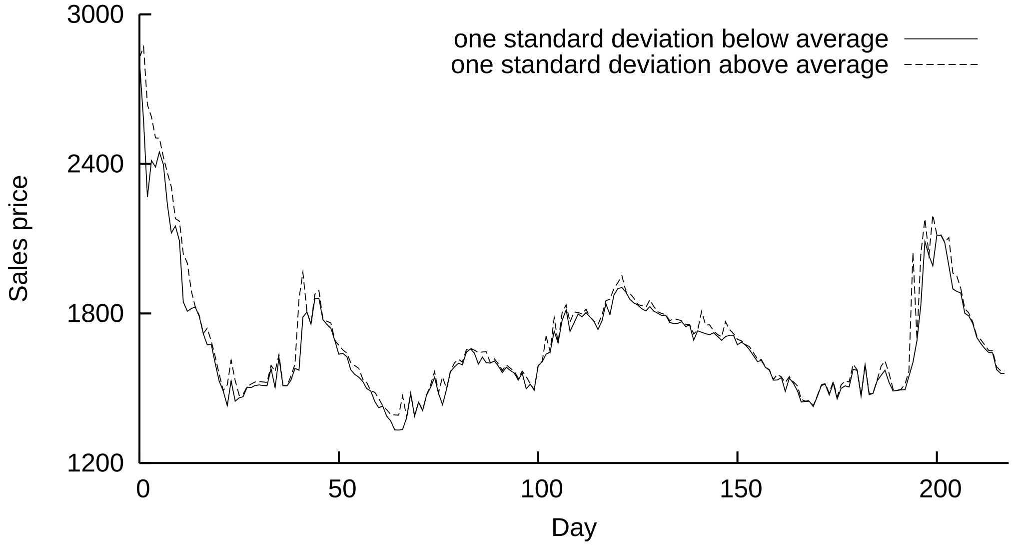

Fig. 3. Average prices at which one type of computer sold during one game of the 2006 finals. One standard deviation above and below the average is shown.

variance drawn randomly from a distribution that is generated by analyzing the first day prices from a large data set of past games. (The source of this data set will be described below.) Each succeeding day, a new set of particles is generated from the old. For each new particle to be generated, an old particle is selected at random based on weight, and the new particle's estimate of mean and variance are set to those of the old particle plus small changes, drawn randomly from the distribution of day-to-day changes seen in the data set of past games. The new particles are then reweighted, with the weight of each particle set to the probability of the previous day's price-related observations occurring according to the distribution represented. These observations consist of the reported highest and lowest winning prices and the acceptance or rejection of each offer made to a customer for the given type of computer. Finally, the weights are normalized to sum to one. The distribution of winning prices predicted by the particle filter is simply the weighted sum of the individual particles' distributions, and from this distribution the function mapping each possible bid price to a probability of acceptance can be determined.

These functions are then modified using values we call *day factors*, which are designed to measure the effect of the due date on offer acceptance. The due dates for RFQs range from 3 to 12 days in the future, and a separate day factor is learned for each day in this range. Each day factor is set to the ratio of actual orders received to orders expected based on the linear heuristic, for all recent offers made. When an offer is made on an RFQ, the Offer Acceptance Predictor computes the probability of an order by multiplying the initial prediction by the corresponding day factor. The day factors therefore serve both as a means of gauging the impact of due dates on computer prices and as a mechanism for ensuring that the number of orders received is roughly the number expected.

In order to maximize revenue from the computers sold, the Demand Manager needs to consider not only the prices it will offer in response to the current day's RFQs, but also what computers it will wish to sell on future days. In fact, the Demand Manager plans ahead for 10 days and considers future as well as current RFQs when making offers, as will be described in the next section. It is therefore important for the Offer Acceptance Predictor to be able to predict future changes in computer prices. To illustrate why this is important, Figure 3 shows the prices at which one type of computer sold during a single game of the 2006 finals. For each day, points representing one standard deviation above and below the average price are plotted. On most days, there is clearly little variance between the winning prices, but prices often change drastically over the course of a few days. This fact suggests that it may be even more valuable to be able to predict future changes in price than to predict the distribution of winning prices on a single day. By simply selling a computer a few days earlier or later, it might be possible for the Demand Manager to significantly increase the price it obtains.

To make these predictions of price changes, the Offer Acceptance Predictor performs machine learning on data from past games. Each training instance consists of 31 features representing data available to the agent during the game, such as the date, estimated levels of customer demand and demand trend, and current and recent computer prices. The label for each instance is the amount by which the average price changes in ten days. Once the Offer Acceptance Predictor has learned to predict this quantity, it can predict the change in average price for any day between zero and ten days in the future through linear interpolation. No effort is made to predict changes in the shape of the distribution, i.e., the variance. Thus, to generate an offer acceptance function for a future RFQ, the Offer Acceptance Predictor simply shifts the predicted distribution over winning prices up or down depending on the predicted change in average price, and bases the acceptance function on this modified distribution.

In order to train the price change predictor, a learning algorithm and source of training data must be chosen. After experimenting with various algorithms from the WEKA machine learning package (Witten & Frank, 1999), we selected additive regression with decision stumps, an iterative method in which a decision stump is repeatedly fit to the residual from the previous step. (M5 regression trees gave nearly identical performance, but the models generated were significantly larger.) For training data, we could have used data from games in the competition, but instead we ran a large number of games of our own using both variations of TacTex-06 and other agents taken from the TAC Agent Repository[3], a collection of agents provided by the teams involved in the competition. Doing so allowed us to generate separate training and testing

[3] http://www.sics.se/tac/showagents.php

data sets for various combinations of six agents, which we then used to test whether predictors trained on data from games with one set of agents would generalize to games involving a different set of agents. In particular, for four different groups of six agents, we ran 40 games, and we generated training data using 30 games and testing data with the other 10. We then trained a separate predictor on each training set. Fortunately, generalization was good: for each of the four testing data sets, all four predictors were reasonably accurate. In other words, in order to predict price changes in a game with a particular group of agents, it was not absolutely necessary to have trained on data specific to those agents. We thus chose to train a single predictor on the entire set of data from these games, and use the same predictor throughout the competition [4].

4.3 Demand Manager

The Demand Manager is responsible for bidding on customer RFQs, producing computers, and delivering them to customers. All three tasks can be performed using the same production scheduling algorithm. As these tasks compete for the same resources (components, completed computers, and factory cycles), the Demand Manager begins by planning to satisfy existing orders, and then uses the remaining resources in planning for RFQs. The latest possible due date for an RFQ received on the current day is 12 days in the future, meaning the production schedule for the needed computers must be sent within the next 10 days. The Demand Manager thus always plans for the next 10 days of production. Each day, the Demand Manager i) schedules production of existing orders, ii) schedules production of predicted future orders, and then iii) extracts the next day's production and delivery schedule from the result. The production scheduling algorithm, these three steps, and the means of predicting production beyond 10 days are described in the following sections.

4.3.1 Production Scheduling Algorithm

The goal of the production scheduler is to to take a set of orders and determine the 10-day production schedule that maximizes profit, subject to the available resources. The resources provided are:

[4] In our post-competition analysis, we found that this was a reasonable decision given the limited number of games that would have been available during the competition to use for training. In more recent work, however, we explore methods of making use of both sources of data (games from the competition and games run on our own) and show that improvements in predictor accuracy are possible (Pardoe & Stone, 2007).

- A fixed number of factory cycles per day;
- The components in inventory;
- The components projected to be delivered; and
- Completed computers in inventory.

The profit for each order is equal to its price (if it could be delivered) minus any penalties for late delivery and the replacement costs for the components involved as specified by the Supply Manager.

The scheduling algorithm used by the Demand Manager is a greedy algorithm that attempts to produce each order as late as possible. Orders are sorted by profit, and the scheduler tries to produce each order using cycles and components from the latest possible dates. If any part of the order cannot be produced, the needed computers will be taken from the existing inventory of completed computers, if possible. The purpose of scheduling production as late as possible is to preserve resources that might be needed by orders with earlier due dates. A record is kept of what production took place on each day and how each order was filled.

It should be noted that the scheduling problem at hand lends itself to the use of linear programming to determine an optimal solution. We initially experimented with this approach, using a linear program similar to one designed for a slightly simplified scenario by (Benisch *et al.*, 2004a). However, due to the game's time constraints (15s allowed per simulated day), the need to use the scheduler multiple times per day (and in a modified fashion for bidding on customer RFQs, as described below), and the fact that the greedy approach is nearly optimal (observed in our own experiments and confirmed by (Benisch *et al.*, 2006a)), we chose to use the greedy approach.

4.3.2 Handling Existing Orders

The Demand Manager plans for the production of existing orders in two steps. Before starting, the production resources are initialized using the values provided by the Supply Manager. Then the production scheduler is applied to the set of orders due in one day or less. All orders that can be taken from inventory (hopefully be all of them to avoid penalties) are scheduled for delivery the next day. The production scheduler is next applied to the remaining orders. No deliveries are scheduled at this time, because there is no reward for early delivery.

4.3.3 Bidding on RFQs and Handling Predicted Orders

The goal of the Demand Manager is now to identify the set of bids in response to customer RFQs that will maximize the expected profit from using the remaining production resources for the next 10 days, and to schedule production of the resulting predicted orders. The profit depends not only on the RFQs being bid on on the current day, but also on RFQs that will be received on later days for computers due during the period. If these future RFQs were ignored when selecting the current day's bids, the Demand Manager might plan to use up all available production resources on the current RFQs, leaving it unable to bid on future RFQs. One way to address this issue would be to restrict the resources available to the agent for production of the computers being bid on (as in Benisch *et al.*, 2004a). Instead, the Demand Manager generates a predicted set of all RFQs, using the levels of customer demand predicted by the Demand Model, that will be received for computers due during the period, and chooses bids for these RFQs at the same time as the actual RFQs from the current day.

Once the predicted RFQs are generated, the Offer Acceptance Predictor is used to generate an acceptance prediction function for every RFQ, both real and predicted. The acceptance prediction functions for predicted RFQs are shifted based on the price changes predicted, as described in Section 4.2. The Demand Manager then considers the production resources remaining, the set of RFQs, and the set of acceptance prediction functions and simultaneously generates a set of bids on RFQs and a production schedule that produces the expected resulting orders, using the following modification of the greedy scheduler.

If we were considering only a single RFQ and had no resource constraints, the expected profit resulting from a particular bid price would be:

$$Expected\ profit = P(order|price) * (price - cost) \tag{1}$$

The optimal bid would be the value that maximized this quantity.

Computing the expected profit from a set of bids when resource constraints are considered is much more difficult, however, because the profit from each RFQ cannot be computed independently. For each possible set of orders in which it is not possible to fill all orders, the profit obtained depends on the agent's production and delivery strategy. For any nontrivial production and delivery strategy, precise calculation of the expected profit would require separate consideration of a number of possible outcomes that is exponential in the number of RFQs. If we were guaranteed that we would be able to fill all orders, we would not have this problem. The expected profit from each RFQ could be computed independently, and we would have:

$$Expected\ profit = \sum_{i\ \epsilon\ all\ RFQs} P(order_i|price_i) * (price_i - cost_i) \tag{2}$$

Our bidding heuristic is based on the assumption that the expected number of computers ordered for each RFQ will be the actual number ordered. In other words, we pretend that it is possible to win a part of an order, so that instead of winning an entire order with probability p, we win a fraction p of an order with probability 1. This assumption greatly simplifies the consideration of filling orders, since we now have only one set of orders to consider, while leaving the formulation of expected profit unchanged. As long as it is possible to fill the partial orders, (2) will hold, where the probability term now refers to the fraction of the order won. It would appear that this approach could lead to unfilled orders when the agent wins more orders than expected, but in practice, this is not generally a problem. Most of the RFQs being bid on are the predicted RFQs that will be received on future days, and so the agent can modify its future bidding behavior to correct for an unexpectedly high number of orders resulting from the current day's RFQs. TacTex-06 indeed tends to have very few late or missed deliveries using this bidding strategy.

By using this notion of partial orders, we can transform the problem of bid selection into the problem of finding the most profitable set of partial orders that can be filled with the resources available, and we can solve this problem using the greedy production scheduler. All bids are initially set to be just above the reserve price, which means we begin with no orders. The scheduler then chooses an RFQ and an amount by which its bid will be lowered, resulting in an increased partial order for that RFQ. The scheduler simulates filling this increase by scheduling its production as described previously. This process is repeated until no more production is possible or no bid can be reduced without reducing the expected profit.

Because we are working with resource constraints, the goal of the greedy production scheduler at each step is to obtain the largest possible increase in profit using the fewest possible production resources. At each step, the scheduler considers each RFQ and determines the bid reduction that will produce the largest increase in profit per additional computer. The scheduler then selects the RFQ for which this value is the largest. In many cases, however, the most limited resource is production cycles, and not components. In such cases, the increase in profit per cycle used is a better measure of the desirability of a partial order than the increase in profit per additional computer, so we divide the latter quantity by the number of cycles required to produce the type of computer requested by the RFQ and use the resulting values to choose which RFQ should be considered next. We consider cycles to be the limiting factor whenever the previous day's production used more than 90% of the available cycles.

The range of possible bid prices is discretized for the sake of efficiency. Even with fairly fine granularity, this bidding heuristic produces a set of bids in significantly less time than the 15 seconds allowed per simulated game day. The complete bidding heuristic is summarized in Table 2.

- For each RFQ, compute both the probability of winning and the expected profit as a function of price
- Set the bid for each RFQ to be just above the reserve price
- Repeat until no RFQs are left in the list of RFQs to be considered:
  · For each RFQ, find the bid lower than the current bid that produces the largest increase in profit per additional computer ordered (or per additional cycle required during periods of high factory utilization)
  · Choose the RFQ and bid that produce the largest increase.
  · Try to schedule production of the partial order resulting from lowering the bid. If it cannot be scheduled, remove the RFQ from the list.
  · If the production was scheduled, but no further decrease in the bid will lead to an increase in profit, remove the RFQ from the list.
- Return the final bid for each RFQ.

Table 2
The bidding heuristic.

4.3.4 Completing Production and Delivery

After applying the production scheduler to the current orders and RFQs, the Demand Manager is left with a 10-day production schedule, a record of how each order was filled, and a set of bids for the actual and predicted RFQs. The bids on actual RFQs can be sent directly to customers in their current form, and computers scheduled for delivery can be shipped. The Demand Manager then considers modifications to the production schedule to send to the factory for the next day. If there are no cycles remaining on the first day of the 10-day production schedule, the first day can be sent unchanged to the factory. Otherwise, the Delivery Manager shifts production from future days into the first day so as to utilize all cycles, if possible.

4.3.5 Production Beyond 10 Days

The components purchased by the Supply Manager depend on the component use projected by the Demand Manager. If we want to allow the possibility of ordering components more than 10 days in advance, the Demand Manager must be able to project its component use beyond the 10-day period for which it plans production. One possibility we considered was to extend this period and predict RFQs farther into the future. Another was to predict future computer and component prices by estimating our opponents' inventories and predicting their future behavior. Neither method provided accurate predictions of the future, and both resulted in large swings in projected component use from one day to the next. The Demand Manager thus uses a simple and conservative prediction of future component use.

The Demand Manager attempts to predict its component use for the period between 11 and 40 days in the future. Before 11 days, the components used in the 10-day production schedule are used as the prediction, and situations in which it is advantageous to order components more than 40 days in advance appear to be rare. The Demand Model is used to predict customer demand during this period, and the Demand Manager assumes that it will win, and thus need to produce, some fraction of this demand. This fraction ranges from zero during times of low demand to 1/6 during times of moderate or high demand, although the Demand Manager will not predict a higher level of component use than is possible given the available factory cycles. While this method of projecting component use yields reasonable results, improving the prediction is a significant area for future work.

5 The Supply Manager

The Supply Manager is responsible for purchasing components from suppliers based on the projection of future component use provided by the Demand Manager, and for informing the Demand Manager of expected component deliveries and replacement costs. In order to be effective, the Supply Manager must be able to predict future component availability and prices. The Supplier Model assists in these predictions.

5.1 Supplier Model

The Supplier Model keeps track of all information sent to and received from suppliers. This information is used to model the state of each supplier, allowing predictions to be made. The Supplier Model performs three main tasks: predicting component prices, tracking reputation, and generating probe RFQs to improve its models.

5.1.1 Price Prediction

To assist the Supply Manager in choosing which RFQs to send to suppliers, the Supplier Model predicts the price that a supplier will offer in response to an RFQ with a given quantity and due date. The Supplier Model requires an estimate of each supplier's existing commitments in order to make this prediction.

Recall that the price offered in response to an RFQ requesting delivery on a given day is determined entirely by the fraction of the supplier's capacity that is committed through that day. As a result, the Supplier Model can compute this fraction from the price offered. If two offers with different due dates are available, the fraction of the supplier's capacity that is committed in the period between the first and second date can be determined by subtracting the total capacity committed before the first date from that committed before the second. With enough offers, the Supplier Model can form a reasonable estimate of the fraction of capacity committed by a supplier on any single day.

For each supplier and supply line, the Supply Manager maintains an estimate of free capacity, and updates this estimate daily based on offers received. Using this estimate, the Supplier Model is able to make predictions on the price a supplier will offer for a particular RFQ.

5.1.2 Reputation

When deciding which RFQs to send, the Supply Manager needs to be careful to maintain a good reputation with suppliers. Each supplier has a minimum acceptable purchase ratio, and the Supply Manager tries to keep this ratio above the minimum. The Supplier Model tracks the offers accepted from each supplier and informs the Supply Manager of the quantity of offered components that can be rejected from each supplier before the ratio falls below the minimum.

5.1.3 Price Probes

The Supply Manager will often not need to use the full five RFQs allowed each day per supplier line. In these cases, the remaining RFQs can be used as zero-quantity price probes to improve the Supplier Model's estimate of a supplier's committed capacity. For each supplier line, the Supplier Model records the last time each future day has been the due date for an offer received. Each day, the Supply Manager informs the Supplier Model of the number of RFQs available per supplier line to be used as probes. The Supplier Model chooses the due dates for these RFQs by finding dates that have been used as due dates least recently.

5.2 Supply Manager

The Supply Manager's goal is to obtain the components that the Demand Manager projects it will use at the lowest possible cost. This process is divided into two steps: first the Supply Manager decides *what* components will need to be delivered, and then it decides *how* best to ensure the delivery of these components. These two steps are described below, along with an alternative means of obtaining components.

5.2.1 *Deciding What to Order*

The Supply Manager seeks to keep the inventory of each component above a certain threshold. This threshold (determined experimentally) is 800, or 400 in the case of CPUs, and decreases linearly to zero between days 195 and 215. Each day the Supply Manager determines the deliveries that will be needed to maintain the threshold on each day in the future. Starting with the current component inventory, the Supply Manager moves through each future day, adding the deliveries from suppliers expected for that day, subtracting the amount projected to be used by the Demand Manager for that day, and making a note of any new deliveries needed to maintain the threshold. The result is a list of needed deliveries that we will call *intended deliveries*. When informing the Demand Manager of the expected future component deliveries, the Supply Manager will add these intended deliveries to the actual deliveries expected from previously placed component orders. The idea is that although the Supply Manager has not yet placed the orders guaranteeing these deliveries, it intends to, and is willing to make a commitment to the Demand Manager to have these components available.

Because prices offered in response to short term RFQs can be very unpredictable, the Supply Manager never makes plans to send RFQs requesting delivery in less than five days. (One exception is discussed later.) As discussed previously, no component use is projected beyond 40 days in the future, meaning that the intended deliveries fall in the period between five and 40 days in the future.

5.2.2 *Deciding How to Order*

Once the Supply Manager has determined the intended deliveries, it must decide how to ensure their delivery at the lowest possible cost. We simplify this task by requiring that for each component and day, that day's intended delivery will be supplied by a single order with that day as the due date. Thus, the only decisions left for the Supply Manager are when to send the RFQ and which supplier to send it to. For each individual intended delivery, the Supply Manager predicts whether sending the RFQ immediately will result in a lower offered price than waiting for some future day, and sends the RFQ if this is the case.

In order to make this prediction correctly, the Supply Manager would need to know the prices that would be offered by a supplier on any future day. Although this information is clearly not available, the Supplier Model does have the ability to predict the prices that would be offered by a supplier for any RFQ sent on the current day. To enable the Supply Manager to extend these predictions into the future, we make the simplifying assumption that the price pattern predicted on the current day will remain the same on all future days. In other words, if an RFQ sent on the current day due in i days would result in a certain price, then sending an RFQ on any future day d due on day $d+i$ would result in the same price. This assumption is not entirely unrealistic due to the fact that agents tend to order components a certain number of days in advance, and this number generally changes slowly. Essentially, we are saying, "Given the current ordering pattern of other agents, prices are lowest when RFQs are sent x days in advance of the due date, so plan to send all RFQs x days in advance."

The resulting procedure followed by the Supply Manager is as follows. For each intended delivery, the Supplier Model is asked to predict the prices that would result from sending RFQs today with various due dates requesting the needed quantity. A price is predicted for each due date between 5 and 40 days in the future. (Each price is then modified slightly according to a heuristic that will be presented in the next section.) If there are two suppliers, the lower price is used. If the intended delivery is needed in i days, and the price for ordering i days in advance is lower than that of any smaller number of days, the Supply Manager will send the RFQ. Any spare RFQs will be offered to the Supplier Model to use as probes.

The final step is to predict the replacement cost of each component. The Supply Manager assumes that any need for additional components that results from the decisions of the Demand Manager will be felt on the first day on which components are currently needed, i.e., the day with the first intended delivery. Therefore, for each component's replacement cost, the Supply Manager uses the lowest price found when considering the first intended delivery of that component, even if no RFQ was sent.

For each RFQ, a reserve price somewhat higher than the expected offer price is used. Because the Supply Manager believes that the RFQs it sends are the ones that will result in the lowest possible prices, all offers are accepted. If the reserve price cannot be met, the Supplier Model's predictions will be updated accordingly and the Supply Manager will try again the next day.

5.2.3 Waiting to Order in Certain Cases[5]

When prices are lower for long term orders than short term orders, the Supply Manager faces an interesting tradeoff. Waiting to order an intended delivery in the short term is expected to increase costs, but by waiting the agent might

[5] This section presents a significant addition to the previous agent, TacTex-05

gain a clearer picture of its true component needs. For example, if customer demand suddenly drops, the agent may be better off if it has waited to order and can avoid unnecessary purchases, even if prices are somewhat higher for those components the agent does purchase. Using the ordering strategy of the previous section, however, the Supply Manager would always choose to place long term orders no matter how small the expected increase in cost would be if it waited.

A number of experiments using the previous version of the agent, TacTex-05, suggest that agent performance would improve if the Supply Manager were to postpone ordering in such situations (Pardoe & Stone, 2006). One possible way of ensuring this behavior would be to modify the current strategy so that instead of sending a request as soon as the predicted price is at its lowest point, the request is only sent when it is believed to be unlikely that a reasonably close price can still be obtained. In TacTex-06, the Supply Manager implements an approximation of this strategy using a straightforward heuristic: predictions of offer prices are increased by an amount proportional to the distance of the requested due date. In particular, the predicted price for a requested due date d days away, $5 \leq d \leq 40$, is multiplied by $1 + x_d$, where $x_d = 0.1 * (d - 5)/35$. Predicted prices are thus increased between zero and ten percent, values chosen through experimentation. As a result, the Supply Manager will wait to order when long term prices are only slightly lower than short term prices.

5.2.4 2-Day RFQs

As mentioned previously, the prices offered in response to RFQs requesting near-immediate delivery are very unpredictable. If the Supply Manager were to wait until the last minute to send RFQs in hopes of low prices, it might frequently end up paying more than expected or be unable to buy the components at all. To allow for the possibility of getting low priced short-term orders without risk, the Supply Manager sends RFQs due in 2 days, the minimum possible, for small quantities in addition to what is required by the intended deliveries. If the prices offered are lower than those expected from the normal RFQs, the offers will be accepted.

The size of each 2-day RFQ depends on the need for components, the reputation with the supplier, and the success of past 2-day RFQs. Because the Supply Manager may reject many of the offers resulting from 2-day RFQs, it is possible for the agent's reputation with a supplier to fall below the acceptable purchase ratio. The Supplier Model determines the maximum amount from each supplier that can be rejected before this happens, and the quantity requested is kept below this amount.

The Supply Manager decides whether to accept an offer resulting from a 2-day

RFQ by comparing the price to the replacement cost and the prices in offers resulting from normal RFQs for that component. If the offer's price is lower than any of these other prices, the offer is accepted. If the quantity in another, more expensive offer is smaller than the quantity of the 2-day RFQ, then that offer may safely be rejected.

The 2-day RFQs enable the agent to be opportunistic in taking advantage of short-term bargains on components without being dependent on the availability of such bargains.

6 Adaptation over a Series of Games

The predictions made by the predictive modules as described above are based only on observations from the current game. Another source of information that could be useful in making predictions is the events of past games, made available in log files kept by the game server. During the final rounds of the TAC SCM competition, agents are divided into brackets of six and play a number of games (16 on the final day of competition) against the same set of opponents. When facing the same opponents repeatedly, it makes sense to consider adapting predictions in response to completed games. TacTex-06 makes use of information from these games in its decisions during two phases of the game: buying components at the beginning of the game (impacting mainly the behavior described in Section 5.2), and selling computers at the end of the game (impacting the behavior in Section 4.2). In both cases, only past games within a bracket are considered, and default strategies are used when no game logs are yet available. We chose to focus on these areas for two reasons. Behavior during these two phases varies significantly from one agent to another, possibly due to the fact that these phases are difficult to reason about in general and may thus be handled using special-case heuristic strategies by many agents. At the same time, each agent's behavior remains somewhat consistent from game to game (e.g. many agents order the same components at the beginning of each game). This fact is critical to the success of an adaptive strategy – the limited number of games played means that it must be possible to learn an effective response from only a few past games.

6.1 Initial Component Orders

At the beginning of each game, many agents place relatively large component orders (when compared to the rest of the game) to ensure that they will be able to produce computers during the early part of the game. Prices for some components may also be lower on the first day than they will be afterwards, depending on the due date requested. Determining the optimal initial orders to place is difficult, because no information is made available on the first day of the game, and prices depend heavily on the orders of other agents.

TacTex-06 addresses this issue by analyzing component costs from past games and deciding what components need to be requested on the first two days in order to ensure a sufficient supply of components early in the game and to take advantage of low prices. The process is very similar to the one described in Section 5.2, except that predictions of prices offered by suppliers are based on past games. First, the components needed are identified, then the decision of which components should be requested is made, and finally the RFQs are generated.

The Supply Manager begins by deciding what components will be needed. On the first day, when no demand information is available (customers begin sending RFQs on the second day), the Supply Manager assumes that it will be producing an equal number of each type of computer, and projects the components needed to sustain full factory utilization for 80 days. On the second day, the Supply Manager projects future customer demand as before and assumes it will receive orders for some fraction of RFQs over each of the next 80 days. The projected component use is converted into a list of intended deliveries as before. (The Supply Manager makes no projections beyond the first 80 days because we have not observed instances where it would be worthwhile to order components so far in advance.)

Next, the Supply Manager must decide which components should be requested on the current day (the first or second day of the game). As in Section 5.2.2, the Supply Manager must determine which intended deliveries will be cheapest if they are requested immediately. At the beginning of the game, the Supplier Model will have no information to use in predicting prices, and so information from past games is used. By analyzing the log from a past game and modeling the state of each supplier, it is possible to determine the exact price that would have been offered in response to any possible RFQ. Predictions for the current game can be made by averaging the results from all past games. When modeling the states of suppliers, RFQs and orders from TacTex-06 are omitted to prevent the agent from trying to adapt to its own behavior. If the initial component purchasing strategies of opponents remain the same from game to game, these average values provide a reasonable means of estimating prices.

At the beginning of the game, the Supply Manager reads in a table from a file that gives the average price for each component for each pair of request date and due date. Using this table, the Supply Manager can determine which intended deliveries will cost less if requested on the current day than on any later day. Intended deliveries due within the first 20 days are always requested on the first day, however, to avoid the possibility that they will be unavailable later. If opponents request many components on the first day of the game but few on the second, the prices offered in response to RFQs sent on the second day will be about the same as if the RFQs had been sent on the first day. Since information about customer demand is available on the second day of the game but not the first, it might be beneficial to wait until the second day to send RFQs. For this reason, the Supply Manager will not send a request for an intended delivery if the price expected on the second day is less than 3% more than the price expected on the first.

Once the Supply Manager has decided which intended deliveries to request, it must decide how to combine these requests into the available number of RFQs (five, or ten if there are two suppliers). In Section 5.2.2, this problem did not arise, because there were typically few requests per day. On the first two days, it is possible for the number of intended deliveries requested to be much larger than the number of RFQs available. Intended deliveries will therefore need to be combined into groups, with delivery on the earliest group member's delivery date. The choice of grouping can have a large impact on the prices offered. When there is only one supplier, the Supply Manager begins by dividing the 80 day period into five intervals, defined by six interval endpoints, with a roughly equal number of intended deliveries in each interval. Each interval represents a group of intended deliveries that will have delivery requested on the first day of the interval. One at a time, each endpoint is adjusted to minimize the sum of expected prices plus storage costs for those components delivered early. When no more adjustments will reduce the cost, the Supply Manager sends the resulting RFQs. When there are two suppliers, ten intervals are used, and intervals alternate between suppliers.

6.2 Endgame Sales

Near the end of each game, some agents tend to run out of inventory and stop bidding on computers, while other agents tend to have surplus computers, possibly by design, that they attempt to sell up until the last possible day. As a result, computer prices on the last few days of the game are often either very high or very low. When end-game prices will be high, it can be beneficial to hold on to inventory so as to sell it at a premium during the last days. When prices will be low, the agent should deplete its inventory earlier in the game. TacTex-06 adapts in response to the behavior of its competitors in past games by adjusting the predictions of the Offer Acceptance Predictor (Section 4.2) during the last few days of each game.

TacTex-06's endgame strategy is essentially to reserve only as many computers for the final few days as it expects to be able to sell at high prices. In particular, from day 215 to 217, the Demand Manager will always respond to a customer RFQ (if it chooses to respond) by offering a price slightly below the reserve. For RFQs received on these days, the probability predicted by the Offer Acceptance Predictor is set to the fraction of computers that would have sold at the reserve price on that day in past games. When the Demand Manager plans for a period of production that includes one of these days, these acceptance probabilities will hopefully result in an appropriate number of computers being saved for these three days.

7 2006 Competition Results

Out of 21 teams that participated in the final round of the 2006 TAC SCM competition, held over three days at AAMAS 2006, six advanced to the final day of competition. After 16 games between these agents, TacTex-06 had the highest average score, \$5.9 million, followed closely by PhantAgent with \$4.1 million and DeepMaize with \$3.6 million [6]. Both PhantAgent and DeepMaize were much improved over their 2005 counterparts, and would very likely have beaten the previous year's champion, TacTex-05, if it had competed unchanged. It thus appears that the improvements present in TacTex-06 were an important part of its victory. Although it is difficult to assign credit for an agent's performance in the competition to particular components, we can make some observations that support this hypothesis.

Figure 4 shows the average, over all 16 games on the final day of the competition, of the profit earned per game day for the top three agents. Daily profit is computed by determining what computers were delivered to customers each day and which components in inventory went into those computers, and then subtracting costs from revenue. TacTex-06 clearly had the highest daily profits over the first 70 days of the game, and after this point profits were roughly equal for all three agents. The difference in profits appears to be accounted for by higher revenue per computer. During the first 70 days of each game, TacTex-06 sold about as many computers as PhantAgent and DeepMaize while paying roughly the same costs for components, but TacTex-06 almost always had a much higher average sales price for each type of computer. After day 70, TacTex-06 still had somewhat higher average computer prices, but these were offset by higher component costs than the other two agents paid.

The ability of TacTex-06 to sell computers at higher prices appears to be due to its attempt to predict future changes in computer prices and react accordingly. During the competition, TacTex-06 could often be seen building up its inventory of completed computers before prices rose or selling off its inventory as prices peaked, while such behavior among other agents was less visible.

[6] Competition scores are available at `http://www.sics.se/tac/scmserver`

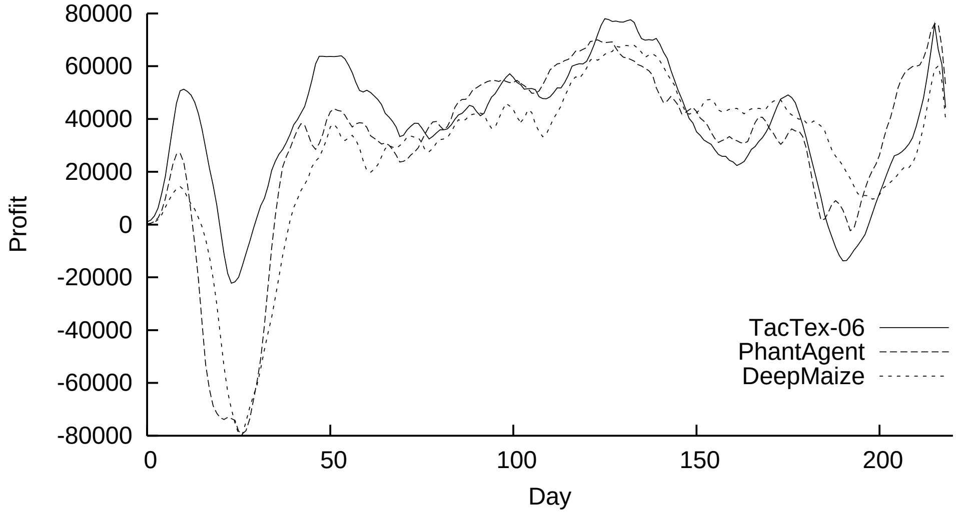

Fig. 4. Daily profits for the top three agents on the final day of the 2006 competition, averaged over all 16 games.

This behavior can explain not only the fact that TacTex-06 sold computers at higher prices, but also the fact that the advantage was especially large in the first portion of each game. To see why, consider Figure 3. For this particular game and computer type, prices began very high, then fell rapidly before recovering somewhat. This pattern is actually very common. Agents begin with no components or computers in inventory, and the supply of computers is thus much smaller than the demand in the beginning of each game. As agents obtain components and begin selling computers, prices usually drop rapidly. Due to the rapid changes in computer prices and the predictability of this pattern, the attempts by TacTex-06 to predict and exploit changes in prices are particularly effective in this period of the game.

To get a clearer picture of how the improvements in TacTex-06 contribute to its performance, we perform a series of controlled experiments in the following section.

8 Experiments

We now present the results of controlled experiments designed to measure the impact of individual components of TacTex-06 on its overall performance. In each experiment, two versions of TacTex-06 compete: one unaltered agent that matches the description provided previously, and one agent that has been modified in a specific way. Each experiment involves 30 games. The other four agents competing — Mertacor, DeepMaize, MinneTAC, and PhantAgent (all versions from 2005) — are taken from the TAC Agent Repository. (Experiments against different combinations of agents appear to produce qualitatively similar results.)

Experimental results are shown in Table 3. Each experiment is labeled with a number. The columns represent the averages over the 30 games of the total score (profit), percent of factory utilization over the game (which is closely correlated with the number of computers sold), revenue from selling computers to customers, component costs, and the percentage of games in which the altered agent outscored the unaltered agent. In every experiment, the difference between the altered and unaltered agent is statistically significant with 99% confidence according to a paired t-test.

The first row, experiment 0, is provided to give perspective to the results of other experiments. In experiment 0, two unaltered agents are used, and all numbers represent the actual results obtained. In all other rows, the numbers represent the *differences* between the results of the altered agent and the unaltered agent (from that experiment, not from experiment 0). In general, the results of the unaltered agents are close to those in experiment 0, but there is some variation due to differences between games (e.g. customer demand), and due to the effects of the altered agent on the economy.

8.1 Supply Price Prediction Modification

As described in Section 5.2.3, the Supply Manager slightly increases the predictions of prices that will be offered for components by an amount proportional to the number of days before the requested due date. This addition to TacTex-06 is designed to cause the agent to favor short term component orders over long term orders if the difference in price is small. In experiment 1, an agent that does not use this technique is tested. Compared to the unaltered agent, this agent has increased component purchases and factory utilization, but the increase in revenue is not enough to offset the higher costs, and the final score is lower than that of the unaltered agent. It appears that the unaltered agent is able to avoid purchasing unprofitable components in some cases by waiting longer to place its orders.

8.2 Offer Acceptance Predictor

We now consider the impact of the improvements to the Offer Acceptance Predictor described in Section 4.2. In experiment 2, the altered agent always predicts that future computer prices will remain unchanged. Not surprisingly, the result is a large decrease in revenue and score. The decrease in score is almost twice as large as the margin of victory for TacTex-06 in the 2006 competition ($1.8 million), adding more weight to the claim of Section 7 that the

Exp. #	*Description*	*Score*	*Util.*	*Revenue*	*Costs*	*Win %*
0	no changes	$7.28M	83%	$104.7M	$94.5M	-
1	no component price prediction increase	-1.42	+3%	+3.51	+4.79	23%
2	no computer price change prediction	-3.51	-1%	-4.50M	-.70M	0%
3	no particle filter	-1.97	-7%	-10.05M	-8.03M	0%
4	no particle filter or prediction	-3.93	-6%	-10.99M	-6.83M	0%
5	heuristic price change prediction	-1.74	0%	-1.14M	-.64M	13%

Table 3
Experimental results. In each experiment, one altered version of TacTex-06 and one unaltered version compete in 30 games, along with four additional agents. Columns represent the total score, percent of factory utilization, revenue from customers, component costs, and how often the altered agent outscored the unaltered agent. Numbers represent millions of dollars. In experiment 0, provided to place other experiments' results in perspective, no alteration is made to TacTex-06, and numbers represent the actual results. In all other experiments, numbers represent the difference between the altered and unaltered agent. In each experiment, the difference between the altered and unaltered agent is statistically significant with 99% confidence according to a paired t-test.

prediction of future price changes played a large role in the winning performance.

In experiment 3, the particle filter used to generate predictions of offer acceptance is replaced with a simpler heuristic that was used in TacTex-05. This heuristic used linear regression over the results of the past five days' offers to generate a linear function used for offer acceptance predictions and was originally used by the agent Botticelli in 2003 (Benisch *et al.*, 2004a). The experiment shows that the particle filter approach is an improvement over this heuristic. The large drop in factory utilization in the altered agent is surprising. Experiment 4 shows the result when the changes of experiments 2 and 3 are combined: the agent makes no predictions of future price changes and uses the linear heuristic instead of the particle filter. The score is only slightly worse than in experiment 2, suggesting that the benefits of using the particle filter are more pronounced when price changes are predicted. It is possible that the more detailed and precise predictions of offer acceptance generated from the particle filter are necessary for the agent to effectively make use of the predictions of future price changes.

In experiment 5, the learned predictor of price changes is replaced with a heuristic that performs linear regression on the average computer price over the last ten days, and extrapolates the trend seen into the future to predict price changes. Although the heuristic's predictions are reasonably accurate, the performance of the altered agent is about midway between that of the unaltered agent and that of the agent from experiment 2 that makes no predictions at all, demonstrating the value of learning an accurate predictor.

9 Related Work

Outside of TAC SCM, much of the work on agent-based supply chain management has focused on the design of architectures for distributed systems in which multiple agents throughout the supply chain must be able to communicate and coordinate (Sadeh *et al.*, 2001) (Fox, Barbuceanu, & Teigen, 2000). These systems may involve a static supply chain or allow for the dynamic formation of supply chains through agent negotiation (Chen *et al.*, 1999). Other work has focused on general solutions to specific subproblems such as procurement or delivery. TAC SCM appears to be unique in that it represents a concrete domain in which individual agents must manage a complete supply chain in a competitive setting.

A number of agent descriptions for TAC SCM have been published presenting various solutions to the problem. At a high level, many of these agents are similar in design to TacTex-06: they divide the full problem into a number of smaller tasks and generally solve these tasks using decision theoretic approaches based on maximizing utility given estimates of various values and prices. The key differences are the specific methods used to solve these tasks.

The problem of bidding on customer RFQs has been addressed with a wide variety of solutions. SouthamptonSCM (He *et al.*, 2006) takes a fuzzy reasoning approach in which a rule base is developed containing fuzzy rules that specify how to bid in various situations. PSUTAC (Sun *et al.*, 2004) takes a similar knowledge-based approach. DeepMaize (Kiekintveld *et al.*, 2004) performs a game-theoretic analysis of the economy to decide which bids to place. RedAgent (Keller, Duguay, & Precup, 2004) uses a simulated internal market to allocate resources and determine their values, identifying bid prices in the process. The approach described in this chapter, where probabilities of offer acceptance are predicted and then used in an optimization routine, is also used in various forms by several other agents. CMieux (Benisch *et al.*, 2006b) makes predictions using a form of regression tree that is trained on data from past games, Foreseer (Burke *et al.*, 2006) uses a form of online learning to learn multipliers (similar to the day factors used in TacTex-06) indicating the impact of various RFQ properties on prices, and Botticelli (Benisch *et al.*, 2004a) uses the heuristic described in Section 8.2.

Like TacTex-06, many agents use some form of greedy production scheduling, but other, more sophisticated approaches have been studied. These include a stochastic programming approach, in which expected profit is maximized through the use of samples generated from a probabilistic model of possible customer orders (Benisch *et al.*, 2004b), and an approach treating the bidding and scheduling problems as a continuous knapsack problem (Benisch *et al.*, 2006a). In the latter case, an ϵ-optimal solution is presented which is shown to produce results similar to the greedy approach of TacTex-06 but in significantly less time for large problems.

Attention has also been paid to the problem of component procurement, although much of it has focused on an unintended feature of the game rules (eliminated in 2005) that caused many agents to purchase the majority of their components at the very beginning of the game (Kiekintveld, Vorobeychik, & Wellman, 2005). Most agents now employ approaches that involve predictions of future component needs and prices and are somewhat similar to the approach described in this chapter. These approaches are often heuristic in nature, although there are some exceptions; NaRC (Buffett & Scott, 2004) models the procurement problem as a Markov decision process and uses dynamic programming to identify optimal actions.

Although several agents make efforts to adapt to changing conditions *during* a single game, such as MinneTAC (Ketter *et al.*, 2005) and Southampton-SCM (He *et al.*, 2005), to our knowledge methods of adaptation to a set of opponents over a series of games in TAC SCM have not been reported on by any other agent. (Such adaptation has been used in the TAC Travel competition, however, both during a round of competition (Stone *et al.*, 2001), and in response to hundreds of previous games (Stone *et al.*, 2003).)

10 Conclusions and Future Work

In this chapter we described TacTex-06, a supply chain management agent consisting of predictive, optimizing, and adaptive components. We analyzed its winning performance in the 2006 TAC SCM competition, and found evidence that the strategy of exploiting predicted changes in computer prices to increase revenue played a significant role in this performance. Controlled experiments verified the value of a number of improvements made to TacTex-05, the previous winner.

A number of areas remain open for future work. There is room for improvement in many of the predictions, possibly though additional uses of learning. Also, by looking farther ahead when planning offers to customers, it may be possible for the agent to better take advantage of the predicted changes in future prices. In addition, there is the question of what would happen if several agents attempted to utilize such a strategy for responding to price changes, and what the proper response to this situation would be.

The most important area for improvement, in both TacTex-06 and other TAC SCM agents, is likely increasing the degree to which agents are adaptive to ensure robust performance regardless of market conditions. While developing

TacTex-06, we had the opportunity to carefully tune agent parameters (such as inventory thresholds) and to test various agent modifications during several rounds of competition and in our own experiments with the available agent binaries. In addition, we were able to implement learning-based approaches that took advantage of data from past games. When developing agents for real-world supply chains, such sources of feedback and experience would be reduced in quantity or unavailable. Although it would still be possible to test agents in simulation, the market conditions encountered upon deployment might differ significantly from the simulated conditions. Designing agents that can adapt quickly given limited experience is therefore a significant part of our future research agenda.

Ultimately, this research drives both towards understanding the implications and challenges of deploying autonomous agents in supply chain management scenarios, and towards developing new machine-learning-based complete autonomous agents in dynamic multiagent domains.

Acknowledgments

We would like to thank Jan Ulrich and Mark VanMiddlesworth for contributing to the development of TacTex, the SICS team for developing the game server, and all teams that have contributed to the agent repository. This research was supported in part by NSF CAREER award IIS-0237699.

References

Arulampalam, S.; Maskell, S.; Gordon, N.; and Clapp, T. 2002. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing* 50(2):174–188.

Benisch, M.; Greenwald, A.; Grypari, I.; Lederman, R.; Naroditskiy, V.; and Tschantz, M. 2004a. Botticelli: A supply chain management agent. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, 1174–1181.

Benisch, M.; Greenwald, A.; Naroditskiy, V.; and Tschantz, M. 2004b. A stochastic programming approach to scheduling in TAC SCM. In *Fifth ACM Conference on Electronic Commerce*, 152–159.

Benisch, M.; Andrews, J.; ; and Sadeh, N. 2006a. Pricing for customers with probabilistic valuations as a continuous knapsack problem. In *Eighth International Conference on Electronic Commerce.*

Benisch, M.; Sardinha, A.; Andrews, J.; and Sadeh, N. 2006b. Cmieux: Adaptive strategies for competitive supply chain trading. In *Eighth International Conference on Electronic Commerce*.

Buffett, S., and Scott, N. 2004. An algorithm for procurement in supply chain management. In *AAMAS 2004 Workshop on Trading Agent Design and Analysis*.

Burke, D. A.; Brown, K. N.; Hnich, B.; and Tarim, A. 2006. Learning market prices for a real-time supply chain management trading agent. In *AAMAS 2006 Workshop on Trading Agent Design and Analysis / Agent Mediated Electronic Commerce*.

Chen, Y.; Peng, Y.; Finin, T.; Labrou, Y.; and Cost, S. 1999. A negotiation-based multi-agent system for supply chain management. *In Workshop on Agent-Based Decision Support in Managing the Internet-Enabled Supply-Chain, at Agents '99*.

Collins, J.; Arunachalam, R.; Sadeh, N.; Eriksson, J.; Finne, N.; and Janson, S. 2005. The Supply Chain Management game for the 2006 Trading Agent Competition. Technical report. Available from http://www.sics.se/tac/tac06scmspec_v16.pdf.

Fox, M. S.; Barbuceanu, M.; and Teigen, R. 2000. Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems* 12:165–188.

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

He, M.; Rogers, A.; Luo, X.; ; and Jennings, N. R. 2006. Designing a successful trading agent for supply chain management. In *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 1159–1166.

Keller, P. W.; Duguay, F.-O.; and Precup, D. 2004. RedAgent - winner of TAC SCM 2003. *SIGecom Exchanges: Special Issue on Trading Agent Design and Analysis* 4(3):1–8.

Ketter, W.; Collins, J.; Gini, M.; Gupta, A.; and Schrater, P. 2005. Identifying and forecasting economic regimes in TAC SCM. In *IJCAI 2005 Workshop on Trading Agent Design and Analysis*, 53–60.

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

Kiekintveld, C.; Vorobeychik, Y.; and Wellman, M. P. 2005. An analysis of the 2004 supply chain management trading agent competition. In *IJCAI 2005 Workshop on Trading Agent Design and Analysis*.

Kumar, K. 2001. Technology for supporting supply-chain management. *Communications of the ACM* 44(6):58–61.

Pardoe, D., and Stone, P. 2006. Predictive planning for supply chain management. In *Sixteenth International Conference on Automated Planning and Scheduling*.

Pardoe, D., and Stone, P. 2007. Adapting price predictions in TAC SCM. In

AAMAS 2007 Workshop on Agent Mediated Electronic Commerce.

Sadeh, N.; Hildum, D.; Kjenstad, D.; and Tseng, A. 2001. MASCOT: an agent-based architecture for dynamic supply chain creation and coordination in the Internet economy. *Journal of Production, Planning and Control* 12(3):211–223.

Stone, P.; Littman, M. L.; Singh, S.; and Kearns, M. 2001. ATTac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research* 15:189–206.

Stone, P.; Schapire, R. E.; Littman, M. L.; Csirik, J. A.; and McAllester, D. 2003. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research* 19:209–242.

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

Witten, I. H., and Frank, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.