RedAgent-2003: An autonomous, market-based supply-chain management agent

Philipp W. Keller  
School of Computer Science  
McGill University  
Montreal, QC, Canada  
pkelle@cs.mcgill.ca

Felix-Olivier Duguay  
School of Computer Science  
McGill University  
Montreal, QC, Canada  
fdugua@cs.mcgill.ca

Doina Precup  
School of Computer Science  
McGill University  
Montreal, QC, Canada  
dprecup@cs.mcgill.ca

Abstract

The Supply Chain Management track of the international Trading Agents Competition (TAC SCM) was introduced in 2003 as a test-bed for researchers interested in building autonomous agents that act in dynamic supply chains. TAC SCM provides a challenging scenario for existing AI decision-making algorithms, due to the high dimensionality and the non-determinism of the environment, as well as the combinatorial nature of the problem. In this paper we present RedAgent, the winner of the first TAC SCM competition. RedAgent is based on a multi-agent design, in which many simple, heuristic agents manage tasks such as fulfilling customer orders or procuring particular resources. The key idea is to use internal markets as the main decision mechanism, in order to determine what products to focus on and how to allocate the existing resources. The internal markets ensure the coordination of the individual agents, but at the same time provide price estimates for the goods that RedAgent has to sell and purchase, a key feature in this domain. We describe RedAgent's architecture and analyze its behavior based on data from the competition.

1. Introduction

Supply-chain management is a key economic task, as it impacts directly the ability of an enterprise to adapt to changing market demands. The Trading Agents Competition Supply Chain Management scenario (TAC SCM), designed by researchers at SICS and CMU [6], provides a challenging test-bed for automated trading agents acting in dynamic supply chains. In TAC SCM, agents representing personal computer (PC) manufacturers compete in markets for components (supplies), as buyers, as well as in markets for finished goods (as sellers), with the purpose of maximizing profits over a simulated year. Several aspects of TAC SCM make it particularly challenging. The agents have to solve a very complex combinatorial optimization problem, regarding which supplies to purchase and how to allocate the existing resources optimally. Moreover, they have to act in the face of tremendous uncertainty regarding the behavior of suppliers, clients, and competitor agents. For instance, agents have to decide which components to buy before they know how much demand there will be in the market. And, of course, this is a very competitive setting.

In this paper we present RedAgent, the winner of the first TAC SCM competition. RedAgent has a highly distributed architecture, in which simple, heuristic-based agents are assigned to deal with individual aspects of the game, such as component procurement and production of customer orders. These agents communicate through a market mechanism in order to determine, collectively, which components to purchase, which types of PCs to produce, how to allocate the available components and production cycles, and what offers to send to customers.

Market-based mechanisms have been used successfully for complex resource allocation problems in grid computing (e.g., [8]), computer networks (e.g., [5]), and scheduling (e.g., [7]). Markets have several advantages over more traditional optimization approaches. First, they offer the possibility of a modular design, in which different agents, responsible for different resources, can communicate through the market. Second, markets provide a more efficient search mechanism than traditional approaches, since alternative courses of action are not considered explicitly. Third, in addition to an allocation, the market also estimates a value attached to each resource. This is perhaps the most important feature from our point of view, as we will describe below.

The paper is structured as follows. In Section 2, we give an overview of the TAC SCM scenario and the possible high-level strategies. Section 3 describes RedAgent’s architecture. Sections 4, 5, 6, 7 and 8 detail the main components of our agent. In Sections 9 and 10 we analyze various aspects of RedAgent’s performance based on data from the final round of the TAC SCM competition.

2. Overview of TAC SCM

In TAC SCM, six agents representing PC assemblers operate in a common market environment, simulated over a period of one year (220 days). The agents are involved in a supply chain, in the sense that they compete both in markets for components (CPUs, motherboards, memory and hard drives) and in the market for finished PCs. These agents can assemble
16 different types of PCs, each using different combinations of components and different numbers of production cycles. Trades with the suppliers, as well as with the customers, are negotiated through a request-for-quotes (RFQ) mechanism. In this mechanism, the buyer issues RFQs to one or more sellers. The sellers respond with offers, which are then accepted or rejected by the buyer. An accepted offer becomes an order, which the supplier has to deliver. The suppliers and customers in the TAC SCM market follow fixed strategies, as described in the game specification [1]. The goal of the PC manufacturers is to obtain the highest profit (i.e., have the highest bank balance at the end of the game).

During each simulated day (which lasts for 15 seconds), the PC manufacturer has to decide which RFQs to issue for components, which of the offers from suppliers to accept, which PCs to manufacture with the available resources, which of the customer orders to ship, and what offers to send to customers in response to their RFQs. This complex decision making process can be simplified by considering two possible high-level strategies: buy-to-build and build-to-order. In the buy-to-build strategy, an agent stocks up on components, and starts producing PCs without necessarily having orders for all the production. In the build-to-order strategy, the first concern of a PC maker is to secure orders from customers; then, PCs are mostly built with the purpose of delivering these existing orders. The buy-to-build strategy has the advantage of ensuring a large stock, which can then be “dumped” on the market at any time. If the other competing manufacturers have low stocks, this has the added benefit of being able to obtain a lot of customer orders at a high profit margin. On the other hand, in a low-demand market, this strategy can be detrimental, given that sales would be low, and profits may not be high enough to cover the cost of the unsold PCs. Even given this potential pitfall, we adopted the buy-to-build strategy, because the absence of costs for maintaining inventory in the TAC SCM scenario made it appealing.

3. Architecture

The TAC SCM problem can be decomposed, at a high level, into three complex, interacting tasks: the procurement of supplies, the allocation of these supplies and (most importantly) of the production cycles to different goods, and the competition for market share. Hence, it seems natural to tackle these problems separately. However, the strong interdependence between these tasks means that the different modules need to exchange a lot of information. RedAgent proposes a unified solution to all these tasks: the use of internal auctions. We initially explored the idea of using auctions to solve the resource allocation problem. However, from a different perspective, auctions provide a relatively simple protocol for the agents to exchange information about their needs and valuations. Most importantly, auctions are a great way of assigning meaningful values to components, as well as finished products.

RedAgent is composed of a number of simple, heuristic-based agents communicating and exchanging resources almost exclusively through auctions, as shown in Figure 1. There are five types of agents:

- An Order Agent (OA) is created for each received order; its goal is to obtain the PCs needed to fill the order and ship them to the customer.
- A Component Agent (CA) is assigned for each of the 10 types of components; these agents provide RFQs and orders for the suppliers.
- A Production Agent (PA) provides production cycles; this is a bottleneck resource, given the fact that only a fixed production capacity is available per day.
- An Assembler Agent (AA) is assigned for each of the 16 types of PC; it obtains components from the CAs and production cycles from the PA, then delivers finished products to the OAs.
- The Bidder sends offers to customers in response to RFQs.

Markets provide the main communication and exchange links between the different agents. A market is simply a predefined series of auctions taking place on each simulated day. RedAgent has one internal market for each type of PC (denoted SKUi in Figure 1), for each component type, and for the production cycles.

4. Internal markets and auctions

All the markets use sequential, sealed-bid double auctions. In each auction, buyers and sellers submit secret offer and demand bids for the type of resource (components, PCs or production cycles) being traded. The offer and demand profiles are the lists of offer and demand bids, in ascending and descending order of bid price respectively. These two profiles are matched, and a single exchange price is determined, which
will be used for all trades. In the current implementation, we use the midpoint between the highest satisfied offer and the lowest satisfied demand as the exchange price. An example of such an auction closing is given in Figure 2. In this case, the top three demands are satisfied, and the exchange prices are all set at $900, given that both the lowest demand partly satisfied and the highest accepted offer are of this value. This mechanism is designed to maximize the overall utility gain, assuming that the bids are indicative of the true valuations of the buyers and sellers. This assumption is justified in our case because this is an internal market in which the participants work towards a common goal. Note that in a general setting with self-interested agents the strategy of bidding true valuations may not be dominant.

After an auction closes, the offer and demand profiles are made available to all the participating agents, in order to allow them to adjust their future bids. In our implementation, only the sellers make use of the demand profiles to adjust their offer prices, but we plan to use this information more extensively in the future.

The auctions take place in a fixed order on each simulated game day. First, the auctions for each type of PC close. These auctions can be considered orthogonal: no agent participates in more than one PC auction, since each order agent only bids for the PCs in its assigned order, and each assembler only offers a single type of PC. The PC auctions provide fresh demand information, which the assembler agents can use to determine how to bid in the resource markets.

The markets for resources hold 5 rounds of auctions each day, with the auctions taking place in predefined order: production cycles (usually the most contentious resource), CPUs (the most expensive resource), motherboards, memory, hard drives. This order reflects the importance of each type of resource (usually the most contentious resource), motherboards, memory, hard drives. This order reflects the importance of each type of resource for the manufacturing process. In each auction, the buyers are the assembler agents (which participate only in the markets for the resources they need), and the main seller is the corresponding component agent or the production agent. However, assembler agents also have the option of selling excess resources. For instance, an assembler agent may purchase a lot of production cycles but then realize that it cannot acquire enough CPUs to sustain its production. In this case, it will attempt to sell the excess production cycles in the next round of auctions, to an assembler agent which can use them more productively. The bidding strategy used by the assemblers is described in detail in section 6.

Since we have a setting where auction participants need to obtain a bundle of goods, it would seem natural to consider using combinatorial auctions. However, resolving combinatorial auctions is quite complicated, and they do not necessarily assign a meaningful value to each individual good. Sequential auctions on the other hand can be resolved quickly, are much simpler to implement, and allow the participating agents to gain more information about the market by observing the successive closing prices and bid profiles. The idea of using sequential auctions for allocating complementary resources has been explored before in [2], but in the context of bids computed by dynamic programming. As we will see shortly, we use simple heuristic bidding.

5. Order Agents

The order agents are created for each customer order, and their goal is to deliver the requested PCs, by acquiring them from the assembler agents. By participating in the PC market, these agents collectively determine how to allocate the available PCs to existing orders. A crucial effect of their bidding, however, is to provide an estimate of the true valuations of different types of PCs. These valuations are used by the Bidder agent when responding to customer RFQs, as described in Section 8. Another important effect is that the PC market provides estimates for the assembler agents of the types, amounts and valuations of the PCs that are needed. Because of the need to produce accurate estimates, good bidding on the part of the order agents is very important for RedAgent’s performance.

Suppose that an order agent is bidding for one PC unit, ordered for a price $p$, and let $d$ be the index of the day relative to the due date of the order (e.g., $d = -1$ one day before the order is due). The bid is computed as a sum of 3 terms: the base price of the PC, the estimated discounted profit for this unit, and the discounted penalties that would be avoided by shipping the PC now. The base price accounts for the components necessary to produce the PC. For each such component $i$, let $c_i$ be the average price that was paid for the units still in stock. Then $c = \sum c_i$ is the estimated material cost of the PC, and the base price is computed as $b(p, c) = \min(p, c)$. Note that if the material cost is lower than the order price, the agent is bid-
Assembling lower, in order to push down the price for PCs of this type in the internal market.

The estimated discounted profit for the PC is computed by subtracting the base price from the order price, and discounting it in a standard fashion, based on the number of days left until the order “expires”:

\[ s(p, c, d) = (p - b(p, c))D^{3-d}_{sale}. \]

In our implementation, \( D_{sale} = 0.9 \). Finally, the penalty-avoidance term represents the money saved by avoiding late deliveries. In the TAC SCM rules, a daily penalty \( q \) is charged for 5 days after the due date if the order has not been delivered, after which the order is canceled. Because of their fairly large amount (between 5% and 15% of the order price), penalties play a significant role in the final profits, so they must be carefully considered when assigning available PCs to orders. The penalty avoidance term is computed as:

\[ r(q, d) = \sum_{i=\max(d, -1)}^{3} D^{3-d}_{pen}q, \]

where \( D_{pen} \) is a discount factor used to model the uncertainty in the possibility of shipping at a later date. In our implementation, \( D_{pen} = 0.7 \).

The bid placed by the order agent is computed as \( b(c, p) + s(p, c, d) + r(q, d) \). A typical bidding profile, as a function of the day around the due date, is shown in Figure 3. Figure 4 illustrates two interesting kinds of bidding curves. The black line corresponds to an “average” profile. The blue curve corresponds to a high-penalty order, and it dominates the other curves around the due date. The sharp drop at the end is due to the fact that all penalty has now been incurred. The red curve corresponds to a high margin order, which should also be assigned high priority. We have to note here that our discount factors have been chosen such that orders get “prioritized” in this way, but RedAgent is very robust to the actual values of these factors. All that matters is that \( D_{sale} \) is close to 1 and \( D_{pen} \) is smaller.

Order agents generally submit a demand bid in the corresponding PC market on each day (according to their profile) until they obtain the needed PCs, but they do not ship as soon as possible. Instead, after they have acquired the PCs, they offer to sell them at their utility value, thus allowing other agents who need them more to obtain their inventory. This is reasonable from a cooperative perspective, but also if the order agents were self-interested, since they are compensated. The larger number of bids generated in this way is also helpful in ensuring that the closing prices of the auctions will be representative of the cost of taking on new orders. Note that if a PC type is in plentiful supply, the order agents obtain it long before the due dates at a lower price, whereas a shortage will force them to wait until their profile shows maximum valuation.

6. Assembler Agents

The 16 assembler agents, one for each PC model, deal with the task of scheduling production. They compete in the component and production cycle auctions in order to acquire the resources necessary for assembling PCs, which they sell in the PC auctions.

The assemblers construct their bids with the goal of maintaining a target inventory, which can be viewed as a buffer to counter spikes in PC demand or temporary shortages of components/production capacity. The target inventory is computed as the number of PCs expected to be needed for 10 days of operation, truncated between static lower and upper bounds (set to 40 and 400 respectively). The expected demand is computed as a running average (over 20 days) of the number of profitable PCs requested by the order agents. The target inventory is linearly increased from 0 in the beginning of the game (over the first 40 days), to avoid “panicked” purchasing. A similar linear decrease is performed at the end of the game (during the last 60 days) to eliminate excess inventory.

Assemblers place offers in the PC market to sell their inventory. In order to form an offer, the base price, \( b_a \), for a given PC is the sum of the component and production costs, estimated using the latest closing prices of the corresponding resource auctions. The whole PC inventory is divided into three batches. The first two batches are equal to the target buffer. The computers in the first batch are priced linearly between \( b_a \) and \( 1.3b_a \), in order to increase the likelihood that this stock is kept. The second batch is priced linearly between 0.7\( b_a \) and \( b_a \). All other computers are offered at 0.7\( b_a \). Hence, if an assembler starts selling computers from its safety buffer, the price it demands for the corresponding type of PC will increase, and vice versa. This information reaches the bidder agent through the PC market.

The auctions on the supply-side are used as a search method for finding a good allocation of resources (components and production capacity) among the 16 assembler agents. To determine what bids to place, the assembler takes the demand profile from the most recent PC auction. This is a sorted list of demand bids, and each bid is a price-quantity pair. An example is shown in Figure 5. Based on the current resource market prices, the assembler eliminates unprofitable bids. Then it adds a series of “fake” bids, so as to maintain
5. Example of constructing resource bids

Figure 5. Example of constructing resource bids

7. Component and Production Agents

The 10 component agents also aim to maintain their inventory in a desired target region. Hence, they need to estimated the rate at which their inventory is getting depleted. We use for this estimate a running average of the daily sales to the assemblers, \( sale_c \). The lower bound \( L \) of the target inventory region is determined so as to maintain enough inventory to operate for 10 game days, assuming that the daily sales will all be equal to \( sale_c \). This buffer is increased from 0 over 40 days in the beginning of the game, and decreased back to 0 at the end of the game. The upper bound \( U \) for the target inventory region on a given day is simply \( L \) plus the expected sales over the rest of the game, based on \( sale_c \).

The bids of a component agent are directly determined as a function of these upper and lower bounds, in a very similar fashion to the bids of the assembler agents. The base price \( b_c \) for the sale of components is the true average cost, the average of the price paid for all the components ordered (but not necessarily received) to date. Then, the entire inventory is offered for sale follows:

\[
\begin{align*}
\text{Inventory portion} & \quad \text{Offer price} \\
[0, L] & \quad \{1.0b_c, 2.5b_c\} \\
[L, U] & \quad b_c \\
[U, U + L] & \quad \{0.5b_c, 1.0b_c\} \\
[U, \infty] & \quad 0.5b_c
\end{align*}
\]

The part of the inventory acting as a buffer is priced at a very high premium in order to quickly push up the price of a resource if it becomes scarce. We allow for a similar buffer over the upper bound to avoid abruptly discounting the price if we have a slight excess in inventory.

The main role of a component agent is sending RFQs to suppliers and responding to returned offers. Determining which RFQs to send is done by projecting the inventory assuming that sales in \( d \) days are given by \( (0.98)^d \cdot sale_c \). When-
ever the projected inventory falls below \( L \), an RFQ is submitted to all potential suppliers, with the due date on that day \( d \). The quantity requested is the quantity expected to be used from \( d \) until the end of the game, capped by a constant (3000 components per order) and by a function of the number of days before the order is due (no more than 200 for every day until the due date). If multiple suppliers can fill the order, half the amount is requested from each instead of requesting the full order twice. A single component agent can send multiple RFQs on a day (with some caveats, described below). If several component agents send RFQs to the same supplier, their order in the RFQ bundle is chosen randomly.

When offers are received in response to these RFQs, they are sorted by price, and the inventory projection is repeated. For every day when components are required, the lowest priced satisfying offer is accepted. It should be noted that if a supplier cannot deliver the requested quantity on the date requested, it sends an earliest-complete offer as well as a partial offer with the requested delivery date. Hence there is no risk in sending too large RFQs, except that it may push up the price of subsequent RFQs for the same component if the supplier allocated production capacity to the first RFQ.

The production agent’s role is somewhat similar to that of the component agents in that it sells resources to the assemblers, but since factory capacity is fixed in the TAC SCM scenario, it offers all the production cycles for a fixed base price. The price paid in the production auction is entirely determined by the bids of the assembler agents.

As the preliminary and seeding rounds of TAC SCM 2003 progressed, most agents started ordering their components on the first day of the game. A detailed discussion of this phenomenon is given in [3]. Essentially, due to the game design, all RFQs sent to suppliers on the first day generated an offer with the lowest possible price, so there was a strong incentive to order large amounts of components on the first day. However, the more manufacturers sent large early orders, the more the suppliers’ capacity tended to be saturated, and the delays for obtaining components became long. We modified the component agent especially for this peculiarity of the game scenario.

On the first day of the game, each component agent sends 5 RFQs to every supplier, requesting the total amount of components needed for an “average” game (as assessed from prior games). The proportions of the RFQs were \((0.55, 0.30, 0.10, 0.035, 0.015)\). These RFQs are sent with a due date of 0, in order to receive earliest-complete offers. We expected that most participants would adopt similar first-day strategies and that RedAgent could only hope to receive the first, and possibly the second, of these orders within a reasonable delay. A simple mechanism was added to avoid accepting any of these large first-day offers for which more than 15% was expected to go unused. The smaller RFQs were left in the eventuality that it was possible to receive some cheap components near the end of the game, even if the larger RFQs were unusable.

### Table 1. Average change in daily prices after the five auctions

<table>
<thead>
<tr>
<th>Round</th>
<th>Production</th>
<th>CPUs</th>
<th>All other components</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26.1 (22%)</td>
<td>78.2 (12%)</td>
<td>17.1 (7.3%)</td>
</tr>
<tr>
<td>2</td>
<td>13.8 (16%)</td>
<td>39.7 (5.2%)</td>
<td>12.4 (5.8%)</td>
</tr>
<tr>
<td>3</td>
<td>12.2 (14%)</td>
<td>27.0 (3.6%)</td>
<td>11.1 (5.7%)</td>
</tr>
<tr>
<td>4</td>
<td>11.0 (11%)</td>
<td>15.0 (2.1%)</td>
<td>8.1 (4.0%)</td>
</tr>
<tr>
<td>5</td>
<td>9.0 (8.1%)</td>
<td>8.9 (1.2%)</td>
<td>6.2 (2.9%)</td>
</tr>
</tbody>
</table>

### 8. Bidder Agent

The bidder’s task is to respond to customer RFQs with offers. Unlike the other internal agents, the bidder does not directly participate in auctions. However, it relies heavily on the closing prices in the PC auctions in order to compute the offer prices.

Initially we attempted to use machine learning techniques in order to estimate the winning bid distributions for the RFQs; we were unsuccessful due to the highly stochastic nature of the bidding process and the changing competitors. Therefore, starting in the seeding rounds, we used an adaptive-margin bidder. The basic idea is to compute the offer price for a PC as the closing price of the corresponding internal market, incremented by an absolute margin per production cycle needed to produce the PC.

To compensate for the noise in the daily market closing prices, we used a running average over four days. A shorter averaging period would cause price oscillations in the customer bidding process because of overcorrections, which would be detrimental to all competitors. On the other hand, the longer the averaging period is, the slower RedAgent reacts to internal changes (shortages, overstocks, etc.).

The per-cycle margin is meant to ensure that RedAgent does not receive more orders than it can produce PCs for, given its limited production capacity. At the same time, it aims to maximize the profit per cycle, since production capacity tends to be the bottleneck resource. The margin increases if RedAgent is receiving too many orders, the total number of RFQs increases, or the orders already do saturate production capacity around the due date of the RFQ being considered. It decreases in the opposite situations.

The resulting strategy is essentially to bid on all RFQs by a certain margin above the internal market value of the PCs being sold, and to adjust this margin so as to use all production capacity after observing the success rate of our offers.

### 9. Internal behavior

The complex setting of TAC SCM makes it difficult to analyze quantitatively the merit of the different components of RedAgent. In this section, we illustrate the merit of the markets both as resource allocation tools, as well as for the purpose of communication and price estimation.
In terms of resource allocation, we hoped that markets would provide a good answer in a fairly timely manner. Table 1 shows the absolute and relative change in closing prices after each of the 5 daily auctions, averaged over all days except the first and last 10 days (which are subject to large inventory variations). The first auction has much higher variations on average than all the other auctions, because the bids take into account new data from the PC markets and suppliers. The changes decrease in further auctions. The auctions for production cycles are the most volatile because cycles are a bottleneck resource, and the assembler agents often divert a lot of funds to purchase them. The CPU auctions show high absolute changes because CPUs are expensive, but they converge the most rapidly. All other components behave very similarly. It is also important to note that in almost all games, there are no price variations over the last 2-3 auctions of the day. Hence, markets converge very quickly to a valuation.

In order to understand better the quality of the allocation produced by the markets we looked at what RedAgent chooses to produce. Figure 6 shows the average number of PCs produced daily, during the 31 final games, as a function of the ratio of material cost vs. production cycles needed for each PC type. As shown, RedAgent has a strong preference for PC types with a large ratio. These are either PCs that can be built quickly, or PCs with high component costs, for which it can apply a large margin.

Figure 7 illustrates the role of the internal markets in ensuring communication between the different agents during a typical game with variable demand. The top graph shows the total demand during the game, computed as the total number of customer RFQs, to all players. The second panel shows the external market price for PCs of a given type, computed as the average of the winning bids on the given day. The third panel shows the internal market closing price for PCs of this type. The external market prices are always high in the beginning of a game because few agents have the necessary components, then decreases later. The spikes in the internal component price represent shortages, while the dips are temporary surpluses. The internal market price is clearly influenced both by the external market and by the component price. It approximates quite successfully the external market price, but spikes when there are component shortages, as well as at the end, when we have depleted our stock. Note that internal market price is effectively a lower bound on any offer sent to customers.

Figure 8 shows the evolution of the margin, production utilization and stock. The margin mostly follows the demand curve, ensuring steady sales. However, when the production is not utilized, the margin also drops in order to ensure that the production cycles are filled up. Note that the margin increases at the end of the game because the internal prices are dropping, but also because other competitors typically run out of stock and stop bidding. Our stock is also decreasing at this time, as RedAgent dumps PCs on the market. In games ending with low demand, the margin decreases to near 0 to sell off the stock.

10. Competition results

In this section, we present results comparing RedAgent to other competitors in the 16 final games of the TAC SCM competition. A more detailed comparison is presented in [4]. In all the following graphs, the agents are ordered from left to right, in the order in which they ranked during the competition.

RedAgent’s buy-to-build strategy is quite apparent from the PC inventory graphs, presented in Figure 9, in which it has by far the largest stock. The buy-to-build strategy has two important effects. On one hand, RedAgent is able to respond very well to customer RFQs, given that there are almost always PCs in stock. Figure 10 shows the ratio of offers sent to customers, divided by the number of RFQs received. As can be seen, RedAgent sends significantly more bids to cus-
tomers than the other teams. We also note that both of the top two teams, RedAgent and DeepMaize, have higher response rates than the rest of the competitors.

A second important effect is that, because RedAgent has large stocks, it can sell when other manufacturers are out of stock, thus obtaining very good prices. Figure 11 shows the average price obtained per PC. RedAgent managed to obtain the best prices in all PC categories, both in the finals (shown here) and in the semi-finals.

11. Discussion and future work

RedAgent uses cooperative agents acting in very simple markets to achieve its goals. We strongly believe that the use of markets in supply-chain management problems is very promising. The use of markets in other domains in which good valuations for resources are needed should also be explored more. RedAgent implements a very simple market scenario. In the future, we plan to explore more complex negotiation and auction protocols in order to communicate more precise information. Another avenue of research his the use of self-interested agents; this would greatly simplify the design of the individual agents, because they would not need to concern themselves with the welfare of the entire system. In the SCM scenario, each manufacturer has a one-step production process. It would be interesting to attempt to apply a market-based approach to a more complex agent, for instance one with multiple stages of production. Finally, we plan to integrate machine learning into RedAgent, in order to track better the behavior of our competitors.

References