RedAgent - Winner of TAC SCM 2003

Philipp W. Keller
pkelle@cs.mcgill.ca

and
Felix-Olivier Duguay
fdugua@cs.mcgill.ca

and
Doina Precup
dprecup@cs.mcgill.ca
School of Computer Science
McGill University

The Supply Chain Management track of the international Trading Agents Competition (TAC SCM) provides a challenging scenario for existing AI decision-making algorithms, due to the high dimensionality and the non-determinism of the environment. Our entry in this competition, RedAgent, is centered around the idea of using an internal market in order to determine what products to focus on and how to allocate the existing resources. This simple design provided a very efficient search mechanism, while at the same time producing price estimates for components, as well as bidding estimates on customer orders.

Categories and Subject Descriptors: K.6.0 [**Electronic Commerce**]: Trading Agents

1. INTRODUCTION

The TAC SCM game scenario poses great challenges, as the agents have to act in a very high-dimensional, non-deterministic environment. A key sub-problem of the game concerns finding a good (if not optimal) allocation of the available resources, especially of the production cycles. Market-based mechanisms have been used successfully for complex resource allocation problems in grid computing (e.g., [Wolski et al. 2001]), computer networks (e.g., [Kuwabara and Ishida 1994]), and scheduling (e.g., [Walsh et al. 1998]). Markets have several advantages over more traditional optimization approaches. First, they offer the possibility of a modular design, in which different agents, which are responsible

Authors' address: 3480 University st., Montreal, QC, H3X2A7, CANADA
This research was supported by NSERC USRA grants to P.W. Keller and F-O. Duguay and by an NSERC operating grant to D. Precup

for different resources, can communicate through the market. Second, markets can 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 offers a value attached to each resource. This is perhaps the most important feature from our point of view, as we will describe below.

Given the complexity of the TAC SCM scenario, our approach was to have 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. This design results in an efficient and flexible decision making process, which helped RedAgent to win the TAC SCM competition.

The paper is structured as follows. Section 2 describes the high-level strategy and the basic design of our agent. In Section 3 we compare our agent to other competitors, based on data from the finals of TAC SCM competition, and discuss avenues for future work.

2. REDAGENT DESIGN

TAC SCM poses a complicated decision making process, because agents have to deal with procurement, production management, as well as bidding for customer orders. However, this 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 in order to deliver 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.

RedAgent is composed of a number of simple, heuristic-based agents, as shown in Figure 1. There are five main 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.

—One *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.

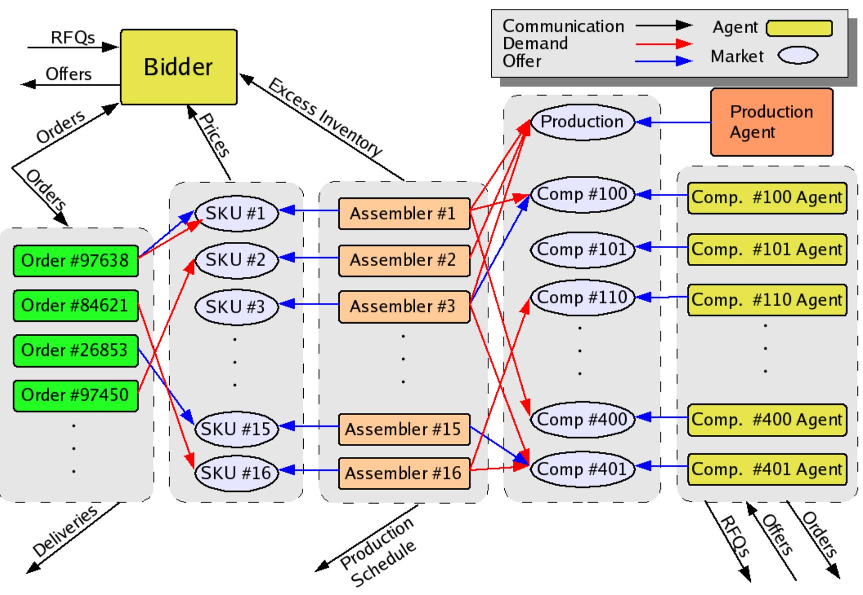

Fig. 1. Overview of RedAgent's components

Markets provide the main communication and exchange links between the different agents. RedAgent has one internal market for each component type, each type of PC and for the production cycles. The goal of the markets is twofold. On one hand, they are used as a search mechanism for the best actions to take internally: which PCs to build, how to allocate the PC inventory to the existing orders, what supplier orders and RFQs to issue. On the other hand, markets also provide value estimates for the PCs that RedAgent produces. This is a very valuable side effect, as the PC price estimates are used by the bidding agent to formulate offers for customers.

In all of these markets, trading is performed through a *sequence of auctions*. The idea of using sequential auctions for allocating complementary resources has been explored before in [Boutilier et al. 1999]. In our case, the mechanism of sequential auctions is desirable, compared to combinatorial auctions, because it allows goods to be purchased from different sellers. As we will see below, this feature is important in our resource markets. However, our work differs from [Boutilier et al. 1999] in that the bids in each auction are not computed by dynamic programming; instead, we use simple heuristics to guide the bidding process, as described below. The main reason for using heuristics is efficiency: because decisions have to be made in a limited amount of time, and because we will run many auctions each day, each involving several agents, we need to ensure that each bid computation is as efficient as possible. The idea of using auctions at different levels of a supply-chain has also been explored before, and an auction protocol designed for this purpose has been proposed in [Babaioff and Nisan 2001]. The main difference in our approach is that we do not propagate information back and forth between the demand side and the supply side of the chain; instead, we estimate the future demand based on current demand. Although using only the current demand can be more imprecise, it allows for faster computation. In the current version of the agent, using current demand proved to be sufficient, but we are considering adding a more complex prediction mechanism in the future.

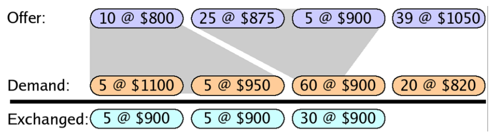

Fig. 2. Example of market exchange

All the markets use a variation of sequential, sealed-bid double auctions. In each auction, buyers and sellers submit secret offer and demand bids for the items traded in that particular auction (components, PCs or production cycles). The offer and demand profiles obtained in this way are matched, and a single exchange price is determined, which will be used for all the 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 a market exchange is given in Figure 2. 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 participating agents work towards a common goal.

The markets for resources hold several rounds of auctions each day, with these auctions taking place in a predefined order: production cycles (usually the most contentious resource), CPUs (the most expensive resource), motherboards, memory, hard drives. In each auction, the main buyers are the assembler agents (which participate only in the markets trading resources they need), and the main seller is the corresponding component agent or production agent. However, assembler agents also have the option of selling excess resources. The goal of repeating the auctions during a day is to allow the participating agents to adjust their valuations based on the current prices and on the availability of other necessary 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 sell the excess production cycles in the next round of auctions, to an assembly agent which can use them more productively.

After an auction closes, the ordered list of demand and offer bids is 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 in order to adjust prices, but we plan to use this information more extensively in the future.

All the component agents in this distributed architecture are based on simple heuristics. Due to lack of space, we cannot describe all these heuristics here, but we refer the interested reader to [Keller et al. 2004]. Here we will just mention the basic ideas of these heuristics.

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. The bid that they place is determined as a sum of three components: the base price of the material necessary for manufacturing the PC, the estimated discounted profit for the order (computed based on the order price and the current component price estimate), and the penalties suffered by not delivering the order. A typical bidding profile, as a function of the day around the order date, is shown in Figure 2. This bidding profile has an important influence on the PC market. If a lot of PCs are being offered by the assembler agents, the order agents buy early, at a low price, and the PC market closing price is close to the total price of the components necessary to produce the PCs. If, on the other hand, not enough PCs are being produced, then there will be a backlog of orders and order agents get their PCs later. This, in turn determines an increase of the closing price.

The closing price in the PC market is very important because it is used by the bidding

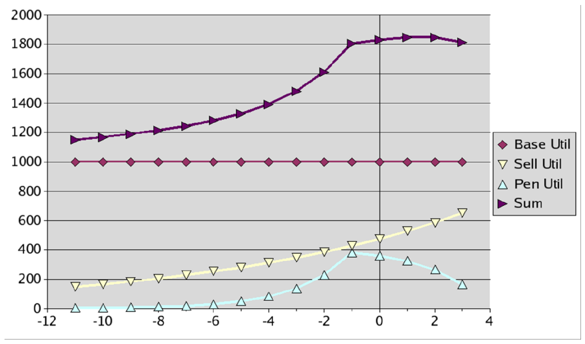

Fig. 3. Typical bidding profile for an order agent, as a function of day number around the order date.

agent in order to determine what to offer in response to customer RFQs. During the competition, we used an adaptive margin bidder. It computes price offers by taking a running average of the closing price from the PC market and adding a margin for each necessary production cycle. The margin is decreased when a lot of production cycles are available, and increased if production is at capacity. By allowing a *margin per production cycle*, rather than per PC, we aim to maximize the profit per cycle, which is important because manufacturing is the main bottleneck in the system.

The assembler agents are responsible for purchasing components and production cycles and then selling assembled PCs to the order agents. 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. 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, to avoid “panicked” purchasing. A similar linear decrease is performed at the end of the game 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 $0.7b_a$ and b_a. All other computers are offered at $0.7b_a$. Hence, if and assembler starts selling computers from its safety buffer, the price it offers will increase in the PC market; similarly, if an assembler has a lot of stock, the PC market price will fall. Hence, this information is transmitted to the bidder using 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. Based on the current resource market prices, the assembler eliminates unprofitable bids. Then it adds a series of “fake” bids, so as to maintain its target inventory. This profile is then translated into profiles for bidding on each component.

Component agents use the same principle of maintaining a desired inventory, but the

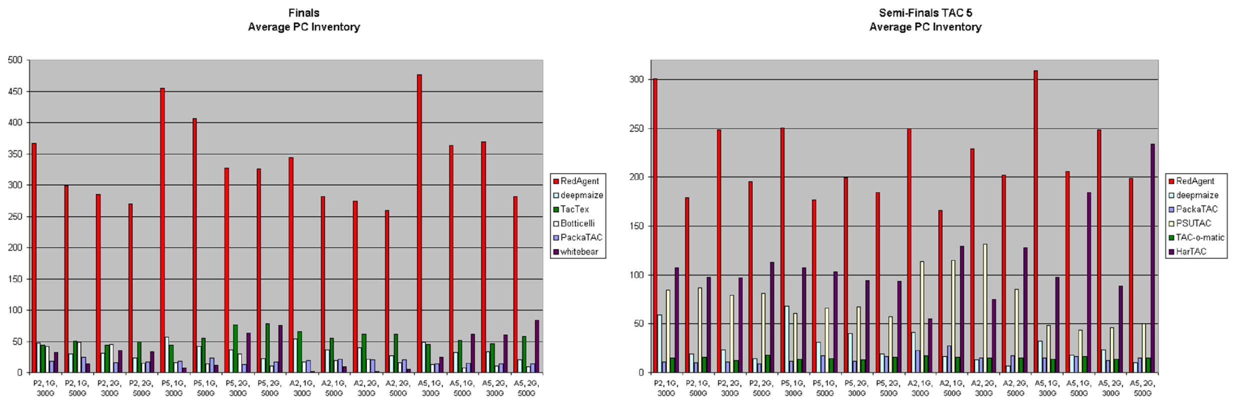

Fig. 4. Average PC inventory during the finals (left) and semi-finals (right)

size of the component buffer depends on the day in the game, as well as the average demand, estimated from the component market. Details for both the assembler agents and the component agents are given in [Keller et al. 2004].

3. COMPETITION PERFORMANCE

In this section, we try to provide some insight into the strategy of RedAgent, as well as that of other competitors. The data comes from analyzing the semi-finals and finals of the competition. 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 4. RedAgent has by far the highest stock of PCs in the final round. During the semi-finals, only one other team, HarTAC, had similar inventory levels for some systems. We note that HarTAC came last during the semi-final round due to technical networking problems, which caused one catastrophic game. Otherwise, their performance would have been much better.

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 always PCs in stock. Figure 5 shows the ratio of offers sent to customers, divided by the number of RFQs received. As can be seen, RedAgent has a significantly response rate 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 very similar profile was seen in the semi-final round, where only HarTAC had similar RFQ response rates.

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 6 shows the average price obtained per PC. RedAgent manages to obtain the best prices in *all PC categories*, both in the semi-finals and in the finals.

A dual perspective on performance is offered by the inventory and price paid for parts, shown in Figure 7 and Figure 8 respectively. As can be seen, RedAgent keeps a fairly large inventory (in order to support its production), but not too large, compared to other competitors. It is also able to secure components at prices that are lower or very similar to those of the competition, thus ensuring higher profit margins than those of other agents. It is worth mentioning here that the design of the game favored agents which buy their

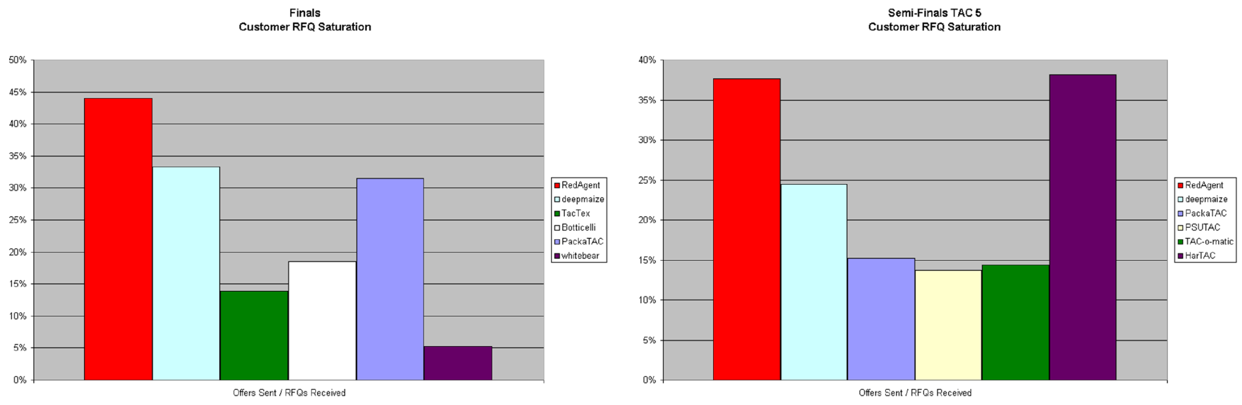

Fig. 5. Percentage of customer RFQs for which offers were sent

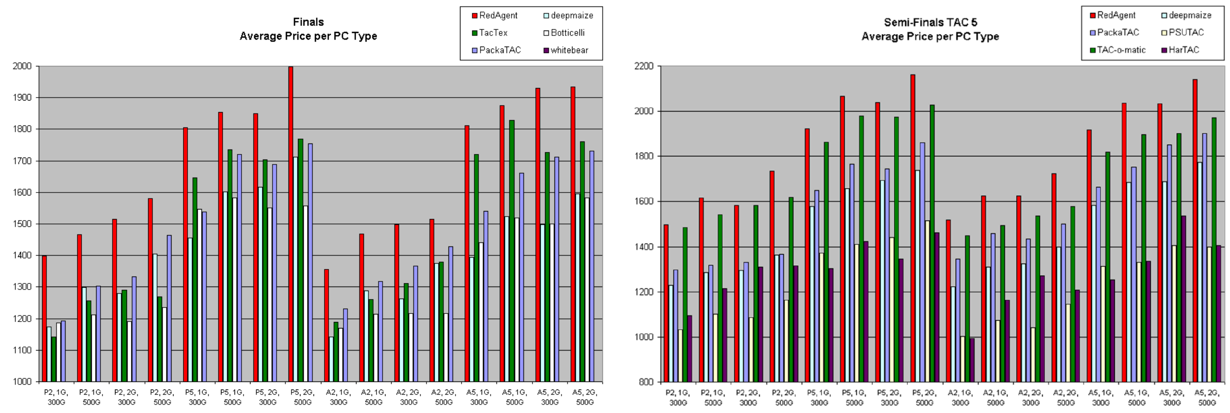

Fig. 6. Average price per PC type

components at the beginning of the game. Some agents (e.g., whitebear) relied exclusively on first-day supplies. RedAgent orders most of the supplies in the beginning, but continues to order supplies later on, as needed. We anticipate that this aspect, which had a high impact on this year's competition [Estelle et al. tted], will be removed next year.

In conclusion, the use of internal markets has provided RedAgent with a very efficient search mechanism for resource allocation, while at the same time ensuring the communication between different agents in our distributed architecture. It is worth noting that RedAgent uses significantly fewer computational resources than other competitors [Kiekintveld et al. 2004]. However, the reliance on markets has the disadvantage of losing a lot of information that could be captured in a richer communication protocol. We plan to explore this issue in future work.

REFERENCES

BABAIOFF, M. AND NISAN, N. 2001. Concurrent auctions across the supply chain. In *The Proceedings of the Third ACM Conference on Electronic Commerce*. 1–10.

BOUTILIER, C., GOLDSZMIDT, M., AND SABATA, B. 1999. Sequential auctions for the allocation of resources with complementarities. In *IJCAI*. 527–523.

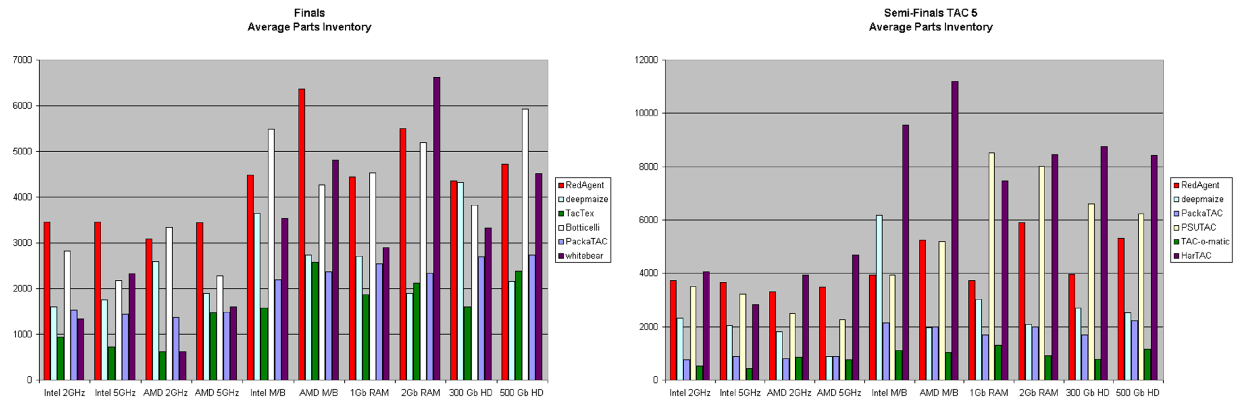

Fig. 7. Average inventory maintained for the different components during the finals (left) and semi-finals (right).

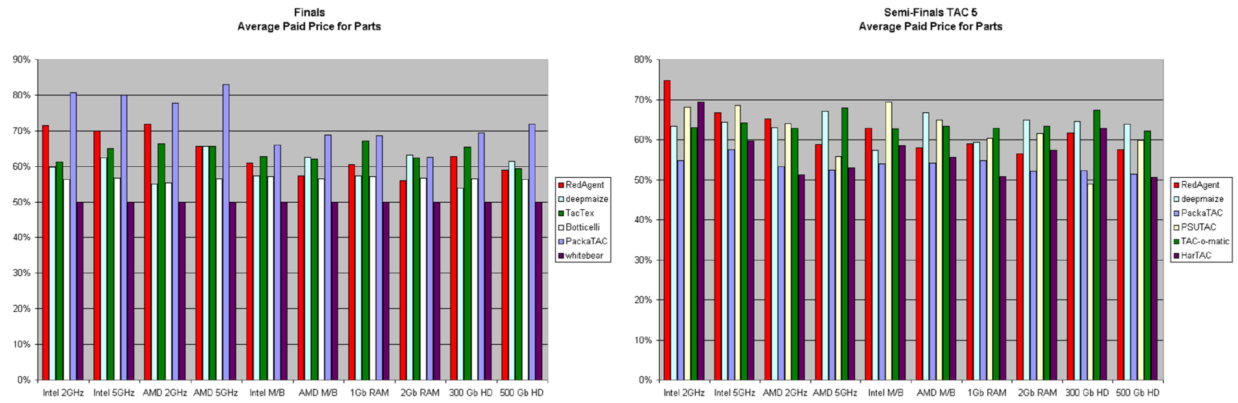

Fig. 8. Average price paid for the different components during the finals (left and semi-finals (right). The price is expressed as a percentage of the base price of each part

ESTELLE, J., VOROBEYCHIK, Y., WELLMAN, M. P., SINGH, S., KIEKINTVELD, C., AND SONI, V. Submitted. Strategic interactions in a supply-chain game.

KELLER, P. W., DUGUAY, F.-O., AND PRECUP, D. 2004. Redagent-2003: An autonomous, market-based supply-chain management agent. In *Submitted.*

KIEKINTVELD, C., WELLMAN, M. P., SINGH, S., ESTELLE, J., VOROBEYCHIK, Y., SONI, V., AND RUDARY, M. 2004. Distributed feedback control for decision making on supply chains. In *ICAPS*.

KUWABARA, K. AND ISHIDA, T. 1994. Equilibratory approach to distributed resource allocation: Toward coordinated balancing. In *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI Volume 830)*, C. Castelfranchi and E. Werner, Eds. Springer-Verlag: Heidelberg, Germany, 133–146.

WALSH, W. E., WELLMAN, M. P., WURMAN, P. R., AND MACKIE-MASON, J. K. 1998. Some economics of market-based distributed scheduling. In *International Conference on Distributed Computing Systems*. 612–621.

WOLSKI, R., PLANK, J. S., BREVIK, J., AND BRYAN, T. 2001. Analyzing market-based resource allocation strategies for the computational Grid. *The International Journal of High Performance Computing Applications 15,* 3 (Fall), 258–281.