Pergamon

Expert Systems with Applications, Vol. 12, No. 1, pp. 1–10, 1997

0957-4174/97 $17.00+0.00

PII: S0957-4174(96)00075-9

Decentralized Problem Solving Using the Double Auction Market Institution

VIJAY RAJAN AND JAMES R. SLAGLE†

Computer Science Department, University of Minnesota, Minneapolis, MN 55455, USA

JOHN DICKHAUT AND ARIJIT MUKHERJI

Carlson School of Management, University of Minnesota, Minneapolis, MN 55455, USA

Abstract—*Decentralized decision making is an important problem in distributed artificial intelligence (DAI) and multi-agent systems (MAS). Given a multi-agent system, the actions or choices made by one agent can affect the actions or choices that can be made by other agents. Wellman (Artificial Intelligence Research, 1, 1–23, 1993) has recently proposed the approach of market-oriented programming for decentralized decision making. By transforming the decentralized decision making problem into a computational economy, market-oriented programming draws on the theory of general equilibrium to establish the existence of a competitive equilibrium. The competitive equilibrium of the computational economy represents the market solution to the original problem. General equilibrium theory, however, is institution free and provides no information about the dynamic process by which the competitive equilibrium is found. Wellman uses a variant of the market institution known as tâtonnements. Tâtonnement's requirement of strict synchronization of the individual agents restricts market-oriented programming to purely atemporal situations (problems that have a stationary environment and hence a stationary equilibrium). In this paper we suggest the use of the continuous double auction as a general framework for market-oriented programming. By using the continuous double auction, market-oriented programming can also be applied to problems where the environment, and therefore the equilibrium, is continuously evolving with time. Copyright © 1997 Elsevier Science Ltd*

1. INTRODUCTION

A CENTRAL PROBLEM IN MULTI-AGENT SYSTEMS is the coordination of autonomous agents without any central control. Although centralized control may provide a better solution to the problem at hand, it is often not feasible when many autonomous agents or sites are involved. For example, traditional methods of query execution become impractical in large distributed data systems, such as Mariposa (Stonebraker et al., 1994a), that support high data mobility and heterogeneous host capabilities (Stonebraker et al., 1994b). Hence, there is a growing need to design mechanisms or institutions that allow a society of autonomous artificial agents to solve problems without any central control.

Societies of artificial agents share some of the same problems faced by real societies, the most important of which are the allocation of resources in ensuring that the necessary tasks of the society are performed. Over the centuries people have found only three ways (Heilbroner, 1992) of organizing human societies. The first is based on tasks being performed under the command of a central authority. The second is based on tradition; individuals performing the tasks that were done by their parents. The third is based on the doctrine of *laissez-faire* and allows each individual to do exactly as he or she saw fit. This simple rule of allowing people to do what is best for them was the foundation of the *market system*. In such a system the lure of gain, not tradition or authority, guided agents to their tasks. Since its beginnings in the late eighteenth century the market system has quickly grown in popularity and is currently the basis for nearly every modern society. For instance, Heilbroner (1992) refers to the adoption of the market system as the most

† All correspondence should be addressed to: Professor James R. Slagle, Department of Computer Science, 4-192 EE/CSci Building, 200 Union Street S.E., Minneapolis, MN 55455, USA.

important revolution in shaping modern society, far more important than the American, French or Russian revolutions.

It is not our intention to argue the merits or misgivings of the market system, but to study its applicability to a society of software agents. In many ways the growth of software systems resembles the growth of human society. For small software systems, the traditional model of computing based on central control worked well. With the increase in the complexity of computational problems, the information and resources required to make efficient decisions are no longer available at a central location, hence the necessity for decentralized models of computation. Since the market system has worked well in human societies, it seems reasonable to explore using markets as a model for artificial societies as well. Wellman's WALRAS system (Wellman, 1993) provides an approach at a framework for decentralized problem solving based on market principles. However, WALRAS's reliance on a variant of tâtonnement is a severe restriction. By using the continuous double auction, market-oriented programming can serve as a more general model of decentralized problem solving in multi-agent systems. Since variants of the double auction are the most widely used market institutions in financial and commodities markets worldwide, societies of artificial software agents can also serve as a means to understand the dynamics of real markets.

This paper builds on the work presented by Rajan and Slagle (1995). The remainder of the paper is organized as follows. Section 2 discusses the use of markets as a model for multi-agent systems. A description of related work is provided in Section 3. Section 4 discusses market-oriented programming and Section 5 describes the double auction. Sections 6 and 7 discuss examples based on atemporal and temporal situations respectively. The conclusion is discussed in Section 8.

2. MARKETS AND MULTI-AGENT SYSTEMS

Two important issues in the design of multi-agent systems are local versus central decision making and synchronous versus asynchronous operation. Although a synchronized system with central decision making may produce a better solution, the time required to acquire the information needed for centralized decision making increases significantly as the size and complexity of the multi-agent system increases. It is therefore desirable to have a mechanism that supports both local decision making and asynchronous operation. However, in a system with no central control some new problems arise. What is the motivation for individual agents to work and how can their activities be led in the direction of a global goal? Markets provide some answers to these questions. Adam Smith, while formulating the laws of the market, sought what he called the *invisible hand*, whereby the private interests of individuals are led in the direction that is most agreeable to the whole society. In *The wealth of nations* (Smith, 1909), Smith describes the two factors that make the market system work: Self-interest and competition. Agents are driven to action by their own self-interest. The regulatory mechanism that guides the behavior of the system is competition. Adam Smith relied on the notion of atomistic competition (no individual was powerful enough to interfere with or to resist the pressure of competition). In this self-regulatory system each agent is forced to scurry after its self-interest in a vast social free-for-all. In a society of software agents this leads us to three main principles: Decentralized decision making, completely asynchronous operation and a market mechanism that reveals appropriate information to the individual agents so that they can compete with each other.

In markets the communication between agents and the formation of prices are governed by specific rules that are referred to as the market institution. The market institution directly affects the way in which agents communicate and behave. It specifies the messages that an agent can send, decides whether the price formation is centralized or decentralized, and whether the agents operate synchronously or asynchronously. The market institution also plays a role in deciding what information is revealed to the individual agents. The behavior of an agent i is characterized by the messages it sends. The messages are specified as a function of the agents environment E^i and the institution I; $m_i = \beta(E^i \mid I)$. It is important to note that different market rules can lead to different results. Ledyard et al. (1994) discuss the effects of the market rules in a computerized exchange system used to solve an allocation problem in project management. So far, two institutions have been used in the study of multi-agent systems. The negotiation procedures used most commonly in multi-agent systems [e.g. ENTERPRISE (Malone et al., 1988) and SPAWN (Waldspurger et al., 1992)], are based on the sealed bid auction. The price formation procedure used by the WALRAS (Wellman, 1993) system is based on the institution of tâtonnement. WALRAS's reliance on tâtonnement requires strict synchronization between the agents and thus restricts its use to completely atemporal situations. The auction procedures used by the ENTERPRISE and SPAWN systems restrict their use to applications where there is one agent selling a resource and there are many buyers (or one agent buying a resource and there are many sellers). Since each agent has to conduct its own auction, much time is spent on communication and the benefits of such a system can decrease as the number of agents increases. For example, the benefits of the ENTERPRISE task scheduler (Malone et al., 1988) leveled off with networks of more than 10 machines. WALRAS also does not completely achieve local decision making since the agents submit their bids to a

central agent (known as the Walrasian auctioneer) that makes the final decision about the market clearing price and quantity.

Our approach is to use a single continuous market rather than having each agent conduct its own auction. The idea is to provide a common language and a central meeting place (marketplace) where the agents can negotiate and exchange resources. The market institution we choose is the continuous double auction (CDA). It provides a completely asynchronous model of operation allowing each agent to decide its own level and time of participation in the market. This provides a realistic environment for market-oriented programming since many of the markets around the world, such as the New York Stock Exchange, are based on variants of the double auction. Decision making is decentralized since each agent decides its own level of participation. From an AI perspective, the market can be viewed as a blackboard with additional functionality to maintain the rules of the market.

3. RELATED WORK

The idea of market prices being used in multi-agent systems is not new. However, previously the emphasis was on models of communication and some form of a sealed bid auction was used. The following is a brief description of systems that use procedures that closely resemble the sealed bid auction. The Contract Net Protocol (CNP) developed by Randall Davis and Reid Smith (1983) was the earliest model of communication between agents in a multi-agent system. CNP provides a protocol for agents in a decentralized environment to coordinate their activities by negotiating without any central control. The negotiation consists of five main steps: Task announcement, task evaluation, bidding, bid evaluation and awarding. An agent that needs a task to be performed is referred to as a manager. The manager begins the bidding process by announcing a description of the task it would like to contract out. All the agents that receive the task announcement make evaluations to determine if they are interested in performing the task. Any agent that is interested submits a bid to the manager. The manager after waiting a certain period of time evaluates all the bids it has received and awards the task to the agent with the winning bid. This protocol closely resembles the sealed bid auction that is commonly used in many government contracts. The agents that receive contracts may in turn sub-contract their tasks. The manager and the contract agents can exchange messages about the progress of the task. CNP serves as a general negotiating protocol but has several problems when used specifically for resource allocation. For example, how long should a manager wait for the other agents to submit bids? Should agents be allowed to submit bids to several managers? If it is allowed, a single agent may win many contracts leading to uneven (and perhaps inefficient) distribution of work.

CNP served as the starting point for many later systems. One such system is the ENTERPRISE task scheduler developed by Malone et al. (1988). ENTERPRISE runs on a network of workstations and distributes tasks from workstations that are busy to those that are idle. The ENTERPRISE scheduling protocol consists of three steps: Announcement, bid and award. A workstation makes a task announcement containing a description of the task. Workstations that are idle respond by submitting bids that represent their estimated processing time for the task. The manager evaluates the bids and awards the task, typically to the bidder with the fastest execution time. ENTERPRISE was able to provide significant performance improvement with a network of 5–10 workstations. The performance did not improve much with more than 10 workstations. Waldspurger et al. (1992) built on the ideas from CNP and ENTERPRISE to develop SPAWN. SPAWN draws more directly on market forces for allocating resources in distributed networks. The relative values of different resources are determined using a second price sealed bid auction. Task allocation is directly based on the amount of resources that participating agents are willing to pay for the resources. Since SPAWN uses a sealed bid second price auction, each workstation would have to independently auction its resources. This can cause a significant portion of the negotiation process to be spent on communication. Our approach is to create a centralized market resulting in less communication and faster contracting.

Wellman's WALRAS system differs from the systems described earlier. The multi-agent problem is first transformed into a computational economy. The competitive equilibrium of the computational economy represents the market solution of the problem. The main difference between WALRAS and the system we propose is how the competitive equilibrium is found. WALRAS relies on a procedure that is a variant of the tâtonnement process to solve for the competitive equilibrium. Individual agents submit their demand and supply functions and WALRAS determines the market clearing price and quantity for each commodity in the market. WALRAS computes the market equilibrium for all auctions taken together in an atemporal fashion. In our approach the equilibrium continuously evolves with time. The differences between WALRAS and our system is detailed in the next section. The Reasoning Economy (RECON) system of Doyle (1994) is built on WALRAS. However it attempts to extend WALRAS to continuous markets by repeatedly using WALRAS to find discrete approximations of an equilibrium that changes over time. This can lead to significant resources (communication and time) being spent on finding the equilibrium. For example, the WALRAS based market for the economic allocation of computation time developed by Nathaniel Bogan (1994) spends on the average 10 times as long computing the market equilibrium as it does executing processes.

4. MARKET-ORIENTED PROGRAMMING

Market-oriented programming is a term coined by Wellman to refer to the use of market based solutions to distributed problem solving (Wellman, 1993). In this approach the original problem is transformed into a computational economy that allocates resources among the agents. To create the computational economy we need to identify the producer agents, the consumer agents and the commodities being traded. In addition, the market institution and the agents behavior can affect the success of this method. For finding the steady-state conditions the computational economy is run until an equilibrium is reached. In general it is difficult to converge at the equilibrium, but for our purpose any solution close to the equilibrium will suffice. This represents the market solution of the original problem. The closer we are to the competitive equilibrium (CE) the better the solution.

General equilibrium theory establishes the existence and optimality of a competitive equilibrium (CE) given certain conditions about the environment (e.g. endowments, preferences) of the agents in the economy (see Debreu, 1959 and Varian, 1992). However, general equilibrium theory is institution free and does not specify the dynamic process by which the economy evolves to the CE. A central agent that has complete knowledge of the structure of the economy can compute the CE. This violates one of the most important conditions of distributed decision making, that the information should be decentralized. As pointed out by Hayek (1945), "the problem is to show how a solution is produced by the interactions of agents each of whom has partial knowledge".

The central issue in market-oriented programming is how to find the competitive equilibrium of the computational economy. WALRAS uses a procedure that is a variant of tâtonnement to determine the price of each commodity. Tâtonnement was first described by the French economist Leon Walras and closely resembled the trading practices of the Paris Bourse during the times of Walras. In this approach a central agent, the auctioneer, begins the auction by setting a price. All the traders respond by sending their desired demands at the particular price. The auctioneer then adjusts the price and the process is repeated until the sum of the demands is zero. This is referred to as the market clearing price. The tâtonnement procedure is not guaranteed to find the competitive equilibrium. Herbert Scarf (1973) has shown that the tâtonnement process can get trapped in limit cycles that never converge to the CE. The approach used by WALRAS is a one-shot clearing house version of tâtonnement.

From the perspective of a multi-agent system the drawback of the tâtonnement procedure is that it requires strict synchronization between the auctioneer and the agents participating in the market. The market clearing price for a commodity can be calculated only after all the agents have submitted their bids. If the market clearing price is calculated before all agents send in their bids, the price might fluctuate significantly without converging towards the equilibrium. There are some additional problems with using tâtonnement to determine the market clearing price. The iterative version of tâtonnement is not incentive compatible (Hurwicz, 1972). Since no trade really occurs until the market clearing price is calculated, the agents can in general misrepresent their supply and demand to shift the computation of the market clearing price to their advantage. Furthermore, the one-shot clearing house version of tâtonnement is often unrealistic since it expects the agents to have a complete mathematical description of their utility functions and depending on the utility function the message space can be extremely large. Because of its simplicity tâtonnement was often used in economic literature to explain the process of price formation, especially in the 60s and 70s; since the early 80s its use has declined. In practice, the only markets that resemble tâtonnement are the London gold market and some Japanese wholesale rice markets (Friedman, 1993).

We propose the use of the continuous double auction (CDA) as the market institution in market-oriented programming. The double auction can support local decision making as well as completely asynchronous operation (allowing continuous price formation) and is therefore not restricted to problems involving atemporal equilibria. Over three decades of experimental research (with human traders) has found that the CDA gets close to the CE both in price and quantity, and the allocations of assets are nearly 100% efficient (Friedman, 1993). Competitive behavior is observed in these markets even with few traders (experimental research typically has used 10 or 12 traders).

5. THE CONTINUOUS DOUBLE AUCTION MARKET

Variants of the double auction (DA) are the most widely used institutions in financial and commodities markets around the world. While many variants of the DA exist (see Friedman, 1993), what we describe below is the version of the continuous double auction (CDA) that we have used over the last few years (Fig. 1). Agents participating in the market can at any time make offers to buy and sell. They do so by sending a message to an agent called the market monitor. The role of the market monitor is similar to that of a specialist in the trading pit of a stock exchange except that it does not participate in any trading. The market monitor verifies the validity of the messages using the market rules, determines when transactions occur and notifies agents that are involved in a trade. The monitor also posts the history of all valid bids, asks and transactions on a blackboard that is accessible to all agents participating in the market. Thus

the market monitor is only an information intermediary. Note that although the messages go through the market monitor, each agent decides its own level of participation in the market. Hence the decision making is decentralized. In addition the messages too can be distributed by having different market monitors for different goods.

The rules of the double auction are as follows. The lowest ask submitted so far is called the current ask, and the highest bid the current bid. The double auction used in many of the real markets, such as the New York Stock Exchange, maintains all the bids and asks in descending and ascending queues respectively. In the version of the double auction we currently use, only the current bid and ask are maintained. Since no queues are maintained we use what is known as the spread reduction rule. By this rule an incoming ask has to be lower than the current ask, and an incoming bid higher than the current bid. If the incoming ask is equal to or less than the current bid a transaction occurs (at the bid price), otherwise it becomes the current ask. Similarly if the incoming bid is equal to or greater than the current ask a transaction occurs (at the ask price), otherwise it becomes the current bid. After a transaction the current bid and ask are removed. However, if the bid and ask quantities were not equal the difference between the bid and ask quantities is issued as a new bid or ask (depending on which was larger). An agent can at any time find the current bid, current ask and quantities.

The working of the double auction market can be characterized precisely in terms of messages. Our characterization of the DA market using messages is similar to Dickhaut and Gjerstad (1994). Each agent in the market is assigned a number i with the market monitor being 0. A message m is the ordered tuple $m=<From,\ Action,\ Commodity,\ Quantity,\ Value>$, where $From \in \{0,...,n\}$ is the originator of the message, $Action \in \{Buy,\ Sell,\ Price,\ Trade,\ Begin,\ End\}$ describes the type of message, *Commodity* represents the resource being sold or bought, *Quantity* is the number of units of the commodity and *Value* represents the price per unit of the commodity. The DA market institution has specific restrictions on the type of messages that the agents and the market monitor can send. Agents can send a message that represent a *Buy* or *Sell* action. The price specified in the *Buy* and *Sell* messages that an agent can send are restricted by the current bid and the current ask respectively. On receiving a valid *Buy* and *Sell* message the market monitor checks to see if a transaction can occur. If a transaction occurs the monitor notifies the agents involved in the transaction. The current bid and current ask are updated based on whether a transaction occurred or not. Agents can also send a *Price* message to find the current bid, current ask and quantities.

The market monitor can originate messages that represent a *Trade*, and the *Begin* and *End* of a trading period (similar to the beginning and ending bell of the NYSE). A *Trade* message is issued by the monitor according to the rules of the DA market. A *Trade* can only happen on two conditions. First, if the monitor receives a message $< i,\ Buy,c,q,b>$ such that $b \geq current\_ask$ and the *current_ask* does not belong to the agent i (an agent cannot buy or sell an asset to itself), then a trade occurs. Second, if there is a message $<j,Sell,c,q,s>$ such that $s \leq current\_bid$ and the *current_bid* does not belong to the agent j, then a trade occurs. In both cases the monitor sends a *Trade* message to the agents that were involved in the transaction. The history of the ith-trading period is $h_i=\{m_1,m_2,...,m_q\}$, where q is the total number of messages in period i. The complete history $H=\{h_1,h_2,...,h_p\}$ represents all the messages observed in the market.

The double auction market has become the focus of

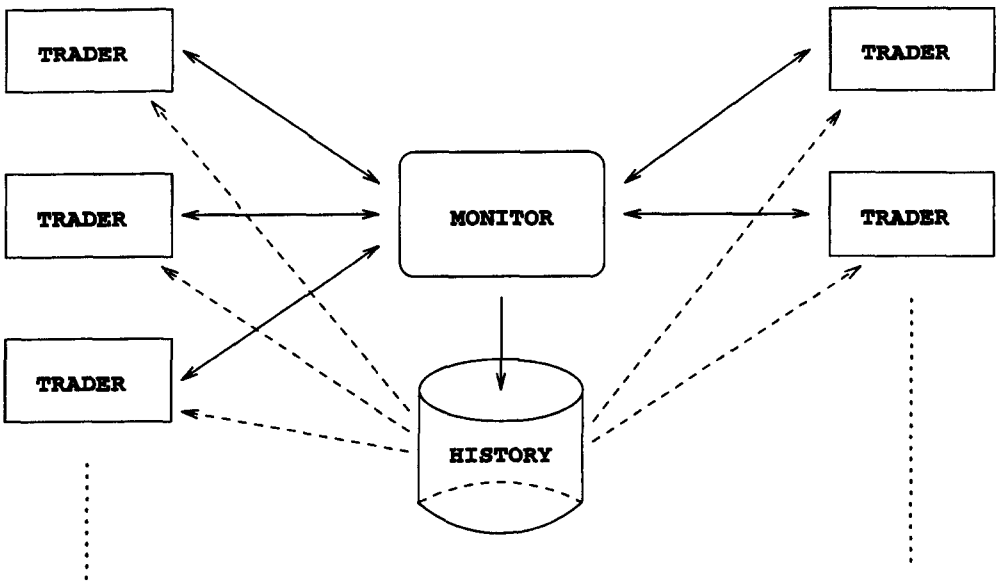

FIGURE 1. The double auction market institution.

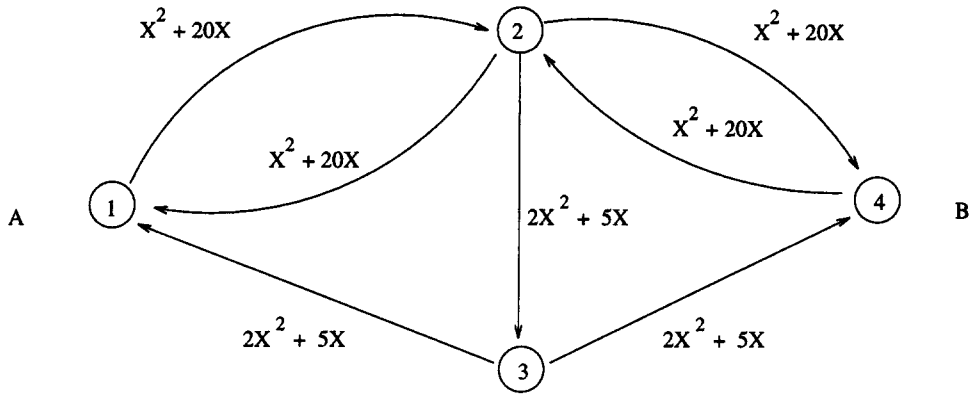

FIGURE 2. A transportation network.

research by economists and computer scientists. Rust et al. (1993) describe the results of a study on the behavior of computerized trading automata in a double auction market. There is also a significant interest in modeling the formation of prices in DA markets (see Dickhaut & Gjerstad, 1994; Easley & Ledyard, 1993; Friedman, 1991; Gode & Sunder, 1993; and Wilson, 1987). In the future, the lessons learned about the formation of prices can be used to build agents with intelligent bidding strategies.

6. SOLVING PROBLEMS THAT INVOLVE A STATIONARY ENVIRONMENT

We have successfully used the approach based on the double auction to solve several multi-agent problems involving decentralized decision making. Here we use the same abstract transportation network (Fig. 2) described first by Harker (1988) and then used by Wellman (1993) to show that the double auction can be used to solve problems that have a stationary equilibrium. An agent A wants to transport cargo from location 1 to location 4. Another agent B wants to transport cargo in the opposite direction from location 4 to location 1. The link from location 2 to location 3 is a common resource to both agents and hence their decisions affect each other. Each agent's objective is to transport 10 units of cargo from its desired origin to its desired destination. The objective of this example is to show local decision making leads to a specific global solution using the double auction.

Let $c_{i,j}(x)$ represent the cost of transporting x units of cargo from location i to location j. The cost functions for the links are

$$c_{12}(x)=c_{21}(x)=c_{24}(x)=c_{42}(x)=x^2+20x$$

$$c_{23}(x)=c_{31}(x)=c_{34}(x)=2x^2+5x$$

We transform this problem into a computational economy. For this we need to specify the goods being traded, the consumer agents and the producer agents. The approach we take is to have the transportation agents A and B, buy tokens for each link in the network that they would like to use. Each token would allow them to transport 1 unit of cargo across the particular link. Assume that they use some currency, say francs, to pay

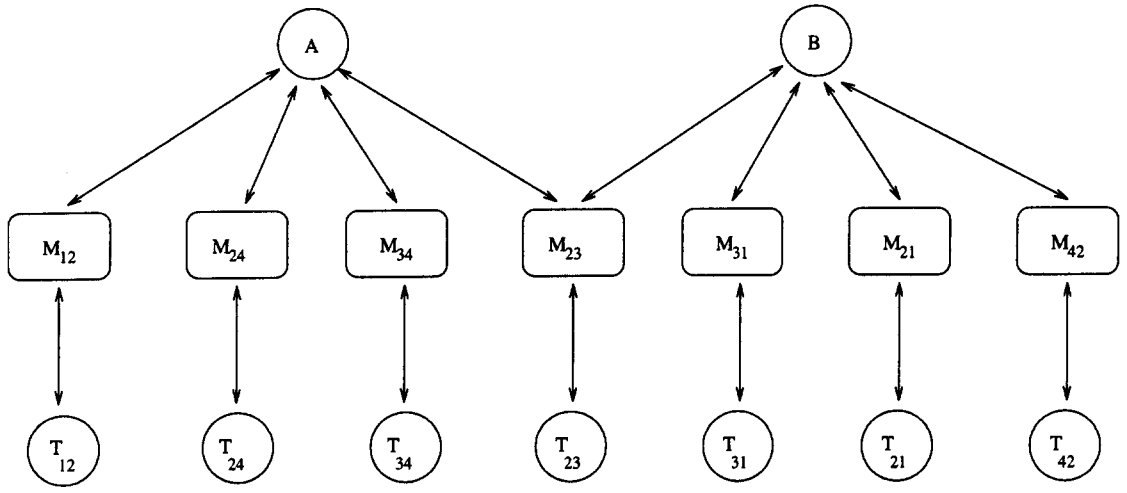

FIGURE 3. Computational economy corresponding to the transportation network.

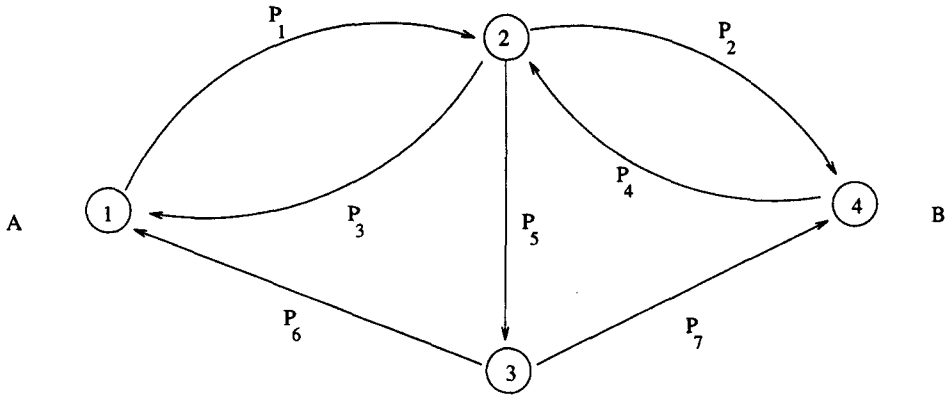

FIGURE 4. The transportation network with a price associated with each link.

for these tokens. Hence the goods in the economy are tokens and francs. The consumers in the economy are the agents A and B (they consume tokens and use francs to pay for the tokens). We introduce seven toll agents (one for each link) who sell tokens. These toll agents are the producers in this economy. The resulting economy is organized as shown in Fig. 3. T_{ij} is the toll agent for the link from i to j. The market institution we use is the double auction. We can configure the system to have one market that allows trading in all commodities, or have a separate market for each commodity. In this example we have chosen the latter configuration. M_{ij} is the double auction market for the link from i to j. One trading period in this computational economy corresponds to the agents A and B each buying enough tokens to transport 10 units of cargo from the source to the destination. After each trading period the agents update their strategies and the process is repeated thereby moving the system towards a global equilibrium.

Finally we need to define the agent behaviors. For now we assume that the toll agents are not trying to profit but are just selling tokens according to the cost function associated with the link. What we describe here is the decision rules used by the agents. The bidding behavior in terms of the exact message sent by the agents can easily be derived from the decision rules.

6.1. Toll Agents (Producers)

During the first trading period the toll agents sell tokens one at a time. Toll agent T_{ij} would sell its yth token for $c_{ij}(y) - c_{ij}(y-1)$ francs. From the second trading period onwards the agents behavior is as follows. Let y_{ij} be the number of tokens sold by agent T_{ij} during the previous trading period. Agent T_{ij} sells up to y_{ij} tokens for $c_{ij}(y_{ij})/y_{ij}$ francs each (the average price). Once y_{ij} tokens are sold the agent reverts to the strategy used in the first trading period.

6.2. Transportation Agents (Consumers)

The objective of each transportation agent is to find the cheapest price to transport its cargo from the source to the destination. During the first trading period these agents buy tokens one at a time, each time comparing prices to find the path with the cheapest cost. From the second trading period onwards these agents re-allocate cargo between the alternative paths based on what the average cost for each path was during the previous trading period. Each agent has two alternative paths from location 2 to its destination. Let us denote them as $path_1$ and $path_2$. If in the ith trading period, the agent bought z_i tokens along $path_1$ at a cost of f_1 francs and $(10-z_i)$ tokens along $path_2$ at a cost of f_2 francs, then its average cost along the two paths are f_1/z_i and $f_2/(10-z_i)$. For trading period $i+1$, if $10>z_i>0$ the agent tries to increase the tokens on the path with the cheaper average cost. If $f_1/z_i \leq f_2/(10-z_i)$ the agent will buy $z_{i+1}=(z_i \times f_2 \times z_i)/(f_1 \times (10-z_i))$ tokens along $path_1$ and $(10\text{-}z_{i+1})$ tokens along $path_2$. If $f_1/z_i > f_2/(10-z_i)$ the agent will buy $z_{i+1}=10\text{-}((10-z_i)\times f_1 \times (10-z_i))/(f_2 \times z_i)$ tokens along $path_1$ and $(10-z_{i+1})$ tokens along $path_2$. If z_i was either 0 or 10 the agent will revert to the strategy used in the first trading period.

The idea is straightforward: "Move some amount of cargo from the more expensive path to the less expensive path." The amount of cargo re-allocated depends on the difference between the two prices. The decision task of each transportation agent is to re-allocate cargo between

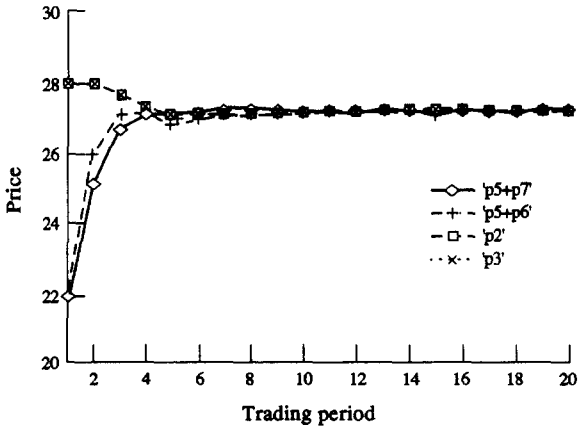

FIGURE 5. The evolution of prices.

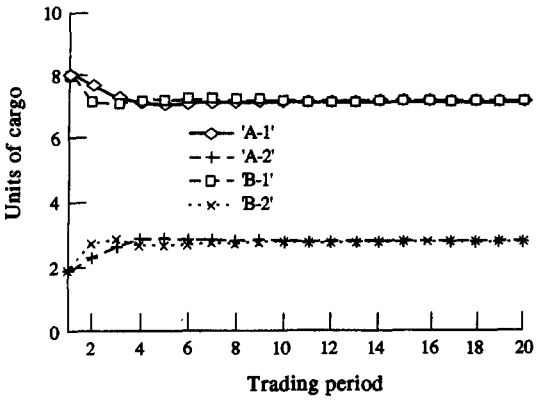

FIGURE 6. Allocation of cargo among alternative paths.

the alternative paths in Fig. 4 until the prices associated with the paths are equal (i.e. $P_2=P_5+P_7$ and $P_3=P_5+P_6$). The evolution of prices is shown in Fig. 5. The allocation of cargo among the alternative paths is shown in Fig. 6 (A_1, A_2, B_1 and B_2 represent the amount of cargo allocated by agents A and B along the two paths $path_1$ and $path_2$ respectively). The market configuration and agent behavior we have defined will find the user equilibrium (Harker, 1988) for this problem (i.e. 2.86 units of cargo will be shipped along the path with the common link). We find that this procedure gets us to within 10% of the equilibrium in about three trading periods and to within 1% of the equilibrium in about nine trading periods.

7. SOLVING PROBLEMS THAT INVOLVE A NON-STATIONARY ENVIRONMENT

The main benefit of using the continuous double auction is that it allows market-oriented programming to be applied to multi-agent systems where the system, and therefore the equilibrium, evolves over time. The following example is based on the same network (Fig. 2) used in the previous section. A similar argument would work for other problems as well, for example, a market

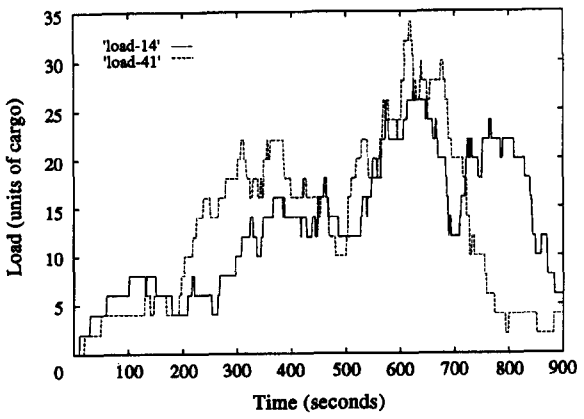

FIGURE 7. Load across the network.

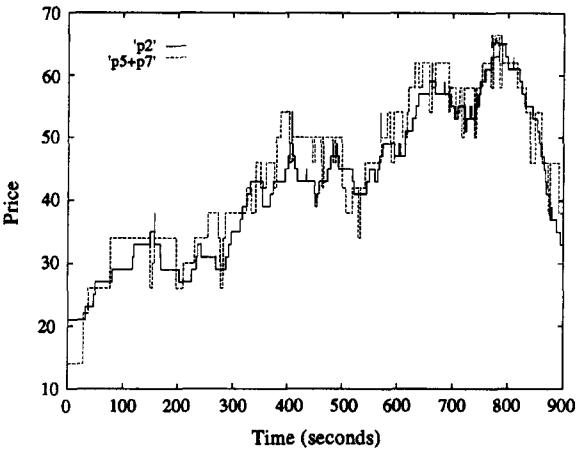

FIGURE 8. Price per unit of cargo on paths 2→4 and 2→3→4.

for computational resources such as processor time and memory. Instead of just the two transportation agents discussed in the previous example, we now allow agents to arrive at random times at locations 1 and 4 and attempt to move two units of cargo each to the other end of the network. The goal here is to dynamically balance the load among the various paths. For this example we consider a duration of 15 min with 50 transportation agents arriving at both location 1 and location 4. The arrival times of the agents are picked randomly using a Gaussian distribution with the mean being 7 min and 30 s and the standard deviation being 4 min. Any negative numbers and values greater than 15 min were discarded. For simplicity we will assume that it takes 2 min for each agent to transport its cargo across the network (in reality the time required to traverse the network should correspond to the load of the network). Figure 7 shows the load across the network during the 15 min duration. In the figure "load-14" represents the units of cargo moving in the direction from location 1 to location 4, and "load-41" represents the units of cargo moving from

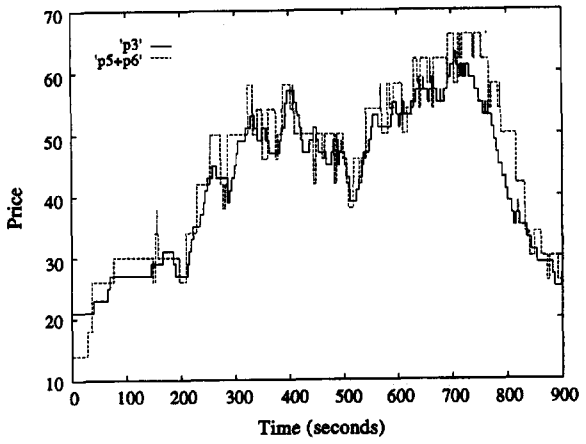

FIGURE 9. Price per unit of cargo on paths 2→1 and 2→3→1.

location 4 to location 1.

The desired global behavior is to balance the load on the network such that $P_2=P_5+P_7$ and $P_3=P_5+P_6$. The agent behaviors required to achieve this global behavior are described below. As in the previous example, what is described are the decision rules and not the exact messages sent by the agents.

7.1. Toll Agents

The toll agents sell tokens one at a time at a price $c_{ij}(l+1)-c_{ij}(l)$, where l is the current load in units of cargo that are currently traversing the path $i \rightarrow j$. They send a message to sell a token whenever the load changes.

7.2. Transportation Agents

The transportation agents on arriving at the starting point compare the cost of transporting cargo across the alternative paths and choose the path(s) that minimizes the cost. They buy enough tokens to transport two units of cargo from the source to the destination.

Figures 8 and 9 show that the prices do not converge to a stationary equilibrium as in the previous example but constantly evolve as the load on the network changes. The example demonstrates how the simple agent behaviors described above lead to the global behavior of dynamic load balancing (i.e. $P_2 \approx P_5+P_7$ and $P_3 \approx P_5+P_6$). Since tokens are sold in units of one, the prices continuously adjust to keep the equilibrium within one unit of cargo. Allowing a token to be split into fractions will narrow the difference between the prices but would require slightly longer negotiation times.

8. CONCLUSION

Market-oriented programming offers a promising method of solving multi-agent problems in terms of resource allocations in a computational economy. However, it inherits the limitations of current economic theory; there is no general theory of trading under different market institutions. Most of the trading institutions used around the world have evolved in an *ad hoc* manner and economists are still attempting to understand the pros and cons of the different institutions. Since no general theory of trading exists, the process by which markets converge is still unsolved. We have relied on the double auction market institution to provide an asynchronous environment that can be used to study and verify the dynamics of multi-agent interactions. The use of the double auction extends Wellman's work on market-oriented programming and provides a more realistic market environment that is capable of solving multi-agent problems with continuously evolving equilibria. The two examples discussed in this paper show that it is possible to design multi-agent systems based on this environment. The relationship between the agent and global behaviors is established by means of self-interest and competition. A key factor in the future success of market-oriented programming will be the ability to design agent bidding strategies that will lead to the competitive equilibrium of the computational economy. We are just beginning to explore the use of rule based systems to represent the behavior of the agents participating in a computational economy. Agents will also need the ability to learn new rules or modify existing ones as they discover new information.

REFERENCES

Bogan, N. R. (1994). Economic allocation of computation time with computation markets. Masters thesis, MIT.

Davis, R., & Smith, R. G. (1983). Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence,* **20**, 63–109.

Debreu, G. (1959). *Theory of value.* New York: Wiley.

Dickhaut, J., & Gjerstad, S. (1994). Price formation in double auction markets. Working Paper, University of Minnesota.

Doyle, J. (1994). A reasoning economy for planning and replanning. Technical Papers of the ARPA Planning Initiative Workshop.

Easley, D., & Ledyard, J. O. (1993). Theories of price formation and exchange in double oral auctions. In Friedman, D., & Rust, J. (Eds), *The double auction market* (pp. 63–97). Reading, MA: Addison–Wesley.

Friedman, D. (1991). A simple testable model of double auction markets. *Journal of Economics Behavior & Organizations*, 47–70.

Friedman, D. (1993). The double auction market institution: A survey. In Friedman, D., & Rust, J. (Eds), *The double auction market* (pp. 3–26). Reading, MA: Addison–Wesley.

Gode, D., & Sunder, S. (1993). Allocative efficiency of markets with zero intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy,* **101**, 119–137.

Harker, P. (1988). Multiple equilibrium behavior on networks. *Transportation Science,* **23**, 39–46.

Hayek, F. A. (1945). The use of knowledge in society. *American Economic Reviews,* **34**, 519–530.

Heilbroner, R. L. (1992). *The wordly philosophers* (6th ed., Chap 2, pp. 19–21). Simon & Schuster.

Hurwicz, L. (1972). On informationally decentralized systems. In Radner, R., & McGuire, B. (Eds), *Decision and organization* (pp. 297–336). Amsterdam: North Holland.

Ledyard, J. O., Porter, D., & Rangel, A. (1994). Using computerized exchange systems to solve an allocation problem in project management. *Journal of Organizational Computing,* **4**, 271–296.

Malone, T. W., Fikes, R. E., Grant, K. R., & Howard, M. T. (1988). Enterprise: A market like task scheduler for distributed computing environments. In Huberman, B. A. (Ed.), *The ecology of computation* (pp. 177–205). Amsterdam: North Holland.

Rajan, V., & Slagle, J. R. (1995). The use of markets in decentralized decision making. In *Proceedings of the Eight International Symposium on Artificial Intelligence*, Monterrey, Mexico.

Rust, J., Miller, J. H., & Palmer, R. (1993). Behavior of trading automata in a computerized double auction market. In Friedman, D., & Rust, J. (Eds), *The double auction market* (pp. 155–198). Reading, MA: Addison–Wesley.

Scarf, H. E. (1973). *The computation of economic equilibria.* Yale University Press.

Smith, A. (1909). *An inquiry into the nature and causes of the the wealth of nations*. The Harvard Classics, P. F. Collier and Son Company. Originally published in 1776.

Stonebraker, M., Aoki, P., Devine, R., Litwin, W., & Olson, M. (1994a). Mariposa: A new architecture for distributed data. In *Proc. 10th Int. Conf. on Data Engineering*.

Stonebraker, M., Devine, R., Kornacker, M., Litwin, W., Pfeffer, A., Sah, A., & Staelin, C. (1994b). An economic paradigm for query processing and data migration in mariposa. In *PDIS*.

Varian, H. (1992). *MicroEconomic analysis* (3rd ed.) Norton.

Waldspurger, C. A., Hogg, T., Huberman, B. A., Kephart, J. O., & Stornetta, S. (1992). Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, **18**, 103–117.

Wellman, M. P. (1993). A market-oriented programming environment and its applications to distributed multicommodity flow problems. *Artificial Intelligence Research*, **1**, 1–23.

Wilson, R. B. (1987). On equilibria of bid-ask markets. In Fiewel, G. W. (Ed.), *Arrow and the ascent of modern economic theory*. New York University Press.