Synthesis and Adaptation of Multi-Agent System

O. Badawy[1], *U. Aburawash*[2], *Y. Hassan*[3], *M. Abdeen*[3]

[1]*Department of Computer Science,*
Arab Academy for Science and Technology, Alexandria, Egypt,

[2]*Department of Mathematics,*
Faculty of Science, Alexandria University, Egypt,

[3]*Department of Mathematics & Computer Science,*
Faculty of Science, Alexandria University, Egypt,

Abstract

The need for intelligent systems has grown in the past decade because of the increasing demand on humans and machines for better performance. The researchers of AI have responded to these needs with the development of intelligent hybrid systems. This paper describes the modeling language for interacting hybrid systems in which we will build a new hybrid model of cellular automata and multi-agent technology. Simulations with complex behavior will be model social dynamics. Therefore, in our approach, cellular automata form a useful framework for the multi-agent simulation model and the model will be used for traffic and computer network systems. In this paper, we will analyze, describe, and design environments in which agents can operate effectively and interact with each other productively. The environment or cellular automata grid will provide a computational infrastructure for such interactions to take place. The infrastructure will include protocols for agents to communicate and interact with each other. The rules that control the agents' growth, death, and behavior have been designed locally for each agent.

1. Introduction

Intelligent agents and multi-agent systems are one of the most important emerging technologies in computer science today [9]. The advent of multi-agent systems has brought together many disciplines in an effort to build distributed, intelligent, and robust applications. They have given us a new way to look at distributed systems and provided a path to more robust intelligent applications.

Multi-agent systems deal with coordinating intelligent behavior among a collection of autonomous agents. Emphasis is placed on how the agents coordinate their knowledge, goals, skills, and plans jointly to take action or to solve problems. Constructing the multi-agent systems is difficult [1]. They have all the problems of traditional distributed and concurrent systems plus the additional difficulties that arise from flexibility requirements and sophisticated interactions.

A Cellular Automaton (CA) [6], as the term is used in this paper, is a discrete state system consisting of a countable network of identical cells that interact with their neighbors. This network can take on any number of dimensions, starting from a one-dimensional string of cells. The cellular automata model is perhaps the simplest, general model available [9]. It is simple in that the basic units are small, local, finite state machines (cells). It is general since: cellular automata model is support universal computation, and the rule represents a general form of local interaction. The two-dimensional cellular automata model is a grid of squares, each square having surrounded adjacent neighbors. A cell occupying a square is born, lives, or dies based on the number of living neighbors it has.

The basic model is detailed in section 2. We delineate below the three basic features by which it differs from the cellular automata model:

1. Whereas the cellular automata model consists of uniform cells, each containing the same rule, we consider the non-uniform case where different agents may contains different rules.
2. The rules are slightly more complex than cellular automata rules.
3. Evolution takes place not only in state space as in the cellular automata model, but also in rule space, i.e. rules may change (evolve) over time.

In section 3, the applications of our model to traffic and computer network systems are presented. The paper will be discussed and concluded in section 4.

2. The Cellular-Agents Model

This section describes the overall design of our automaton. A system with a collection of communicating agents is constructed by composition of atomic agents. It will consider two-dimensional cellular automata model which consists of array of cells *x(i, j)*. The use of a cell as intelligent agent provides an even greater amount of flexibility to the ability and configuration of the system itself. Each agent contains the same or different rule according to which agent states are updated in a synchronous and local manner. The neighborhoods of the agent *x(i, j)* are the von-Neumann neighborhood of radius *r*.

The agent in our new model consists of four sub-agents: ***input***, ***output***, ***learner***, and ***controller***. The sub-agent ***input*** communicates with the neighbors to get their states and transfers them to the sub-agent ***controller***. The sub-agent ***output*** takes as input the variable *change* to update the state of the agent. The sub-agent ***learner*** selects and evolves the rule that will be used to change the state of the agent. The sub-agent ***controller*** models the control laws and outputs the variable *change*. We will describe these sub-agents in more details below. The neighbor agents communicate through the sub-agent **input** and the communication between the agents can be hidden from the outside world. The variable *change*, used internally by the agent, cannot be accessed from the outside.

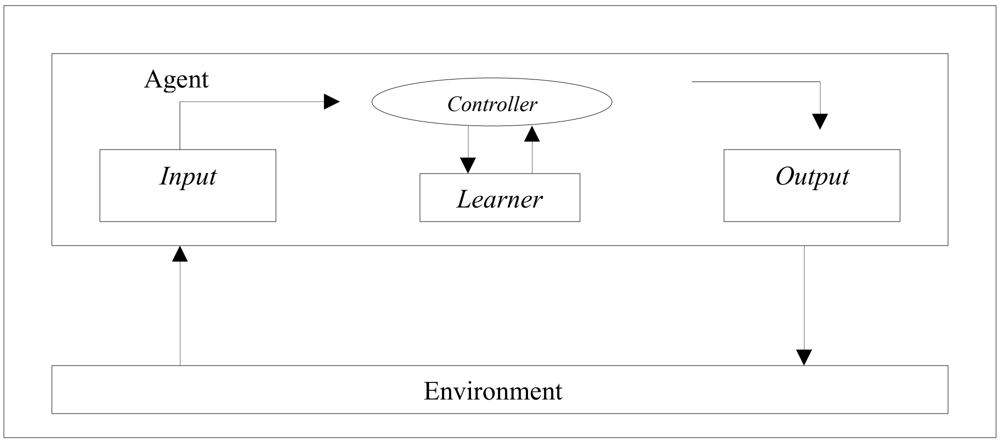

Figure 1: The framework of the agent.

Shown in **Figure 1** is the agent framework and the environment of the agent. The environment reports a value (denoted *y*) sending to the agent. An agent observes the environment through its input and presents an indication of that (denoted *x*) to the controller. The controller in turn decides on the action to be taken (denoted *change*) and reports that action to its output. This action changes the environment. Upon this change, Based on this cycle, the sub-agent leaner tries to learn optimal action selection (or optimal policy) in the agent's environment using genetic programming technique.

The set of agents in the model needs to interact so that the agents use communication protocol to process their interactions. The communication is held between each agent and a

set of neighbor agents. Communication protocols are typically specified at several levels [10]. The lowest level of the protocol specifies the method of interconnection; the middle level specifies the format, or syntax, of the information being transferred; the top level specifies the meaning of the information.
Once the automaton was embedded in the grid, the agent began to follow the rule that stored in it. A single agent cannot do much without interacting with other agents, and it has no concept of the whole. Yet, in combination it can play its part in producing complex results as emergence from local interactions. Briefly, each agent from the grid able to:

1. exercise a degree of freedom in its operations. It takes initiative and exercises a non-trivial degree of control over its own actions (Autonomy).
2. collaborate and exchange information with other agents in the grid (environment) to assist other agents in improving their quality of decision making as well as its own (Collaboration).
3. learn its optimal action by evolving its local rule.
4. change its state and the states of its immediate neighbors.
5. copy its rule into a neighboring died agent (travail self-reproducing). Contention occurs if more than one neighbor attempts to copy itself into the same agent. Such a situation is resolved randomly, i.e. one of the neighbors ‘ wins’ and copies its rule into the cell.
6. neither read nor write directly to other agents in the grid except its immediate neighbors.

2.1. The Evolutionally Engine (Leaner)

In this subsection, we study evolution as it occurs in our model. Learning in an observable and non-stationary environment is still one of the challenging problems in the area of multi-agent systems. Our algorithm of learning for our model requires learning from interactions in an environment in order to achieve certain goals. At each time step, the agent observes its environment and selects the next actions based on that observation. In the next time step, the agent obtains the new observation that may reflect the effects of its previous action and a payoff value indicating the quality of the selected action.
The cellular space considered in this subsection is 2-states, 5-neighbor where states are denoted {0, 1}. The learning or evolution of the rule that the agent may use is achieved by using technique similar to genetic programming [4]. An individual in the population of genetic programming is a tree structure form; therefore, the agent needs to convert the rule into tree structure. Followed that applied the genetic operators. The mutation operation will modify the rule of the agent, while the crossover operator depends on the interaction between the agent’s rule and the neighbor agents’ rules.

2.2. Agents Communication

When agents function together they form a multi-agent system. Multi-agent systems provide higher level abstractions than traditional distributed programming. These abstractions are closer to user expectations and allow the designer more flexibility in determining behavior. The meaning of a communication you must examine many elements, including perspective, type of meaning, basis (semantics or pragmatics), context, and coverage (the number of communicative acts included).
We can define communication as those interactions that preserve the autonomy and heterogeneity of the parties involved

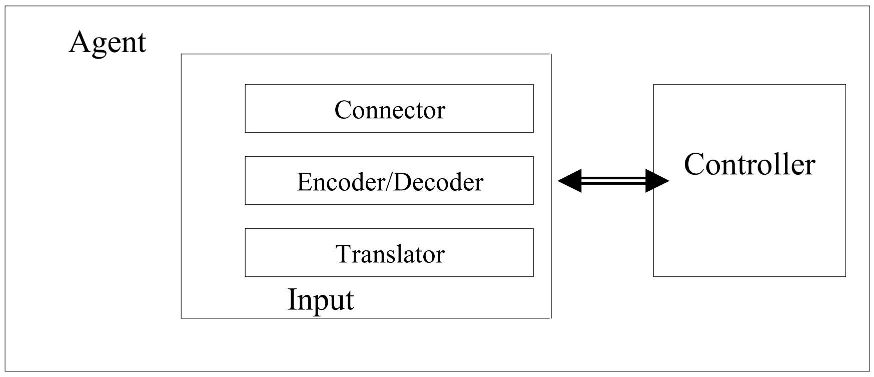

Figure 2: the internal structure of sub-agent sensor.

The agent can communicate with another agent through the sub-agent input. The input provides the interface to the external environment (other agents). The internal structure of the input can be found in **Figure 2**. It consists of three main components: connector, encoder/decoder, and translator. The first component ‘connector’ is responsible for the lowest-level part of the communication, i.e. it uses interconnection protocols. The encoder/decoder is transforming incoming and outgoing message it also passes out coming statements and handles them to translator module. The translator module translates the unified message into the designated agent.
As stated in [8], the communication dialogue between two agents can be expressed in the form of cellular automata. In the suggested CA model, there exist three types of cells: cell “A” which is the sender, cell “B” which is the receiver and cell “M” which is the messenger. A state exchange dialogue between sender and receiver cells occurs in order to achieve a certain goal in the application.
The “M” cell is reproduced by either the sender or receiver according to communication requirement in order to exchange data (“A” will create an “M” cell to start communication and to reply on any message it receives, “B” will create an “M” cell to only reply on the messages it receives). Either cell “A” or “B” will change the “M” cell state according to the type of message. It should be mentioned that each state the messenger cell holds represents one of the seven communicative acts that represents the exchange of information in any heterogeneous and autonomous agent [8].

2.3. Agent Fitness

In this subsection, we enhance our model by adding a measure of agent fitness, specifying how well it performs in a certain environment depending upon the environment under consideration and the interaction with each other. This value is useful to decide that the rule of the agent needs to apply evaluation process on it or not.
Our approach is applied in which each agent contains a single strategy and the agent’s score is then compared to its neighbors and when the agent has a neighbor with higher fitness than its own, apply the evolutionary process to enhance or change its rule.

2.4. Growths and Production System

At this point, we present a system involving the growth and replication of complex structures, which are created from grid agents and are behave as mulicellular organisms once formed [2]. We described a self-reproducing loop, which exhibited a two-fold utilization of information, i.e. translation and transcription. Now we examine a system of reproduction consisting of active agents called share-builder agent. Share-building operation involves two agents operating in unison where each agent builds a part of the structure of new organism.

Again, it is crucial that the whole process be local in nature since no global indicators can be used.
A share-builder agent interacts with another share-builder agent to produce a new offspring, which contains a different rule as mixed of non-equal parts of that the parents have. The offspring begin with state as the maximum state of the parent have. This operation is similar to the crossover operation that is used in Genetic Algorithms [4].
It is important to note the difference between our approach and genetic algorithms [4]. Though we apply the operation similar to genetic operator (crossover operator) in a similar fashion there is no selection mechanism operating on a global level using the total fitness of the entire population. Note also in the standard genetic algorithm model each entity is an independent coding of a problem solution interacting only with the fitness function, never seeing the other entities in the population nor the general environment that exists. In contrast, in our case fitness depends on interactions of evolving organisms operating in an environment.

3. Applications

The Cellular-Agents Model is applied on two applications. The first one dealing with traffic and transportation trying to find an efficient ways to model and predict traffic flow. The second is a simplified model for computer network.

3.1. Traffic model

The present work constructs a traffic model based on the hybrid system of cellular automata and multi-agent system. The new model of traffic system deal with the basic question of decision-making under such an amount of inconsistent information [3]. We will focus on the decision made by an individual driver as well as the consideration of the interaction caused by such a decision on the system as a whole.
We start by proposing a road traffic model suitable for an urban environment. North, east, south and west car displacements and road crossings are possible. *k*-states multi-speed (k finite called maximum velocity v_{max}) car motion is found to be a crucial ingredient to describe highway traffic and phenomena. The basic idea is to consider the grid of cells as two types: first type forms a group to represent buildings (type agent-A) and second agent type constructs groups of agents representing (type agent-B). Each site of agent-B type can be in one of the v_{max} states: it may be empty (state = zero or die), or it may be having an integer velocity between one and v_{max}. This integer number of the velocity is the number of sites each vehicle advances during one iteration.
The agent consists of four subagents, namely ***Velocity***, ***Sensor***, ***Controller***, and ***Learner***. The sub-agent controller models the control laws and outputs the variable *change* which increase or decrease the velocity of the car based on the distance and velocity of the car in front. The sub-agent velocity takes as input the variable *change* and updates the velocity *v* of the agent. The sub-agent sensor detects the velocity of the car in the front and the distance between the car and that in front of it. All agents travel in the same direction of the road, and their positions are updated synchronously in successive iterations (time steps).
Each of the *N* cars in the system starts at a randomly selected site, and the car is assigned a destination site on the lattice. Once a car reached its destination, it will be assigned a new randomly chosen destination. The goal of the driver is the strategy: *Drive as fast as you can and slow down if you have to until arrive the destination site!*. The driver is free to change its mind if an intersection is shortly locked. As a result, the traffic is not distributed uniformly and some road segments are much more loaded than the others are. The load distribution changes with time, as a result of the microscopic fluctuations.

The behavior of cars to reach its destination in this model is actually easy to handcraft, while it is hard to learn for many algorithms. Traffic simulation here is made more realistic by given individual drivers intentions, i.e. an idea of where they want to go and choose an optimal trip. Let us consider how we might develop the implementation using genetic programming technique. Each car has an associated path that defines the combinational actions in the behavior of this car at each road crossing in its path to arrive the destination. After the car determines the destination, leaner builds randomly a set of trees corresponding to paths to this destination as an initial population. Using genetic operators, the set of paths are evolved and shortest path is chosen to the car follows it. If the car faces a jam or lock at road crossing, it uses genetic programming to evolve or change its path (see leaner section).

3.1.1. Results Of Simulation

We now turn our attention to the model itself. This takes the form of a simulation program. As we will see, the phenomena it generates while doing so include simulations of information building, learning processes and interactions between cars in the system. The two-dimensional automata model consists of $L \times L$ sites on rectangular lattice with periodic boundary conditions in both directions. We simulated system of size L = 20, 30, 50, 100 and no striking difference in the behavior of the system was observed. Hence, we exclude finite size effects. The fact that already comparatively small lattices exhibit this independence from their size seems to indicate the existence of only weak spatial correlations between sites separated by a large distance.

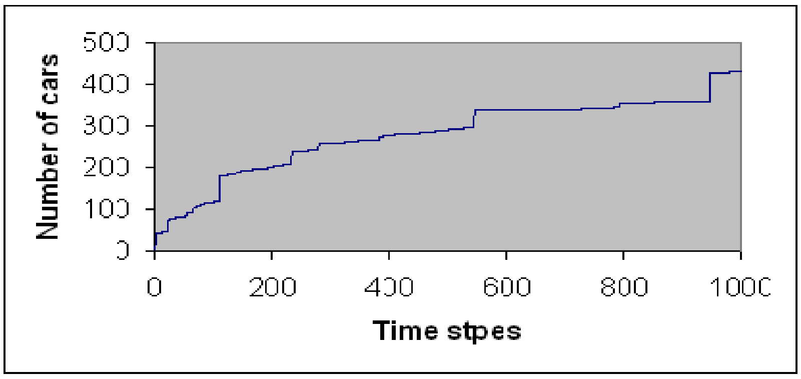

Figure 3: The fitness of the cars

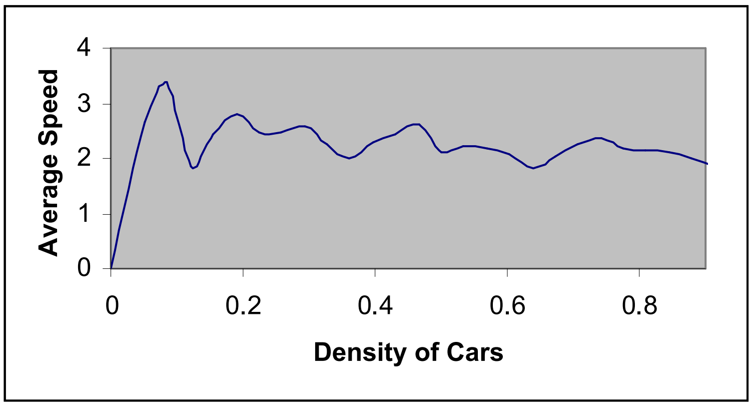

Figure 4: The relation of total average speed and density of cars.

Figure 3 shows the fitness of cars until time steps t=1000. **Figure 4** shows the changing of average speed of agents in the system with the density of cars. We observe that the average speed of the agents reduced with the increase of the density of cars in the system. **Figure 5**

shows the changing of the flow in the system with the density of cars. We observe also that the flow of cars is relatively good in the high density of cars.

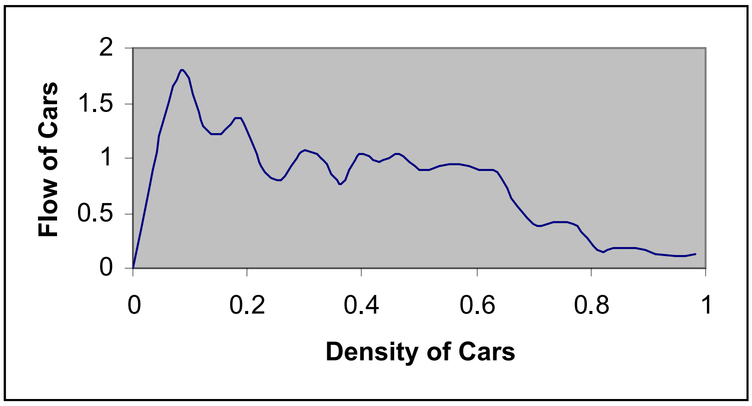

Figure 5: The varying of flow with the modification of density of cars.

3.2. A Computer Network Model

The network model is a one-dimension CA model it composed of two kinds of cells, namely node agent and link agent. The node agent represents ones such as computers, routers, and switchers in computer network. The link agent is viewed as the abstract of the lines of communication among these nodes. The node agent is allowed to have more than one data packets at any time, while the link agent can be occupied by at most one packet at a given instant.

A typical configuration of the network model is shown in **Figure 6**. The number of link agent connected two adjacent node agents stands for the bandwidth of network [7]. The larger the number is, the more time it needs to take for packets to be transferred from a node agent to the other. It directly means a lower bandwidth. The bandwidth reaches the maximum if no link agent exists between two adjacent node agents. In this case, packets can be directly transferred from one node to the other. This implies that the link agent is not necessary and the network model can consist of node agents only.

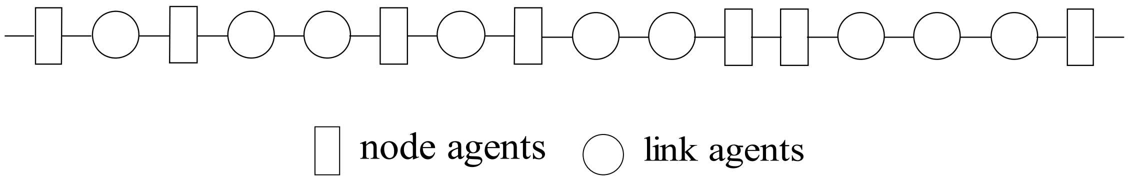

Figure 6: A typical configuration of network model.

The state of agent can be characterized by the number of packet it has [4] . Obviously, for node agent, it can be taken an arbitrary value from 0 to N, where N is the maximum of the packets a node agent can hold at a given instant. However, the state of link agent can have only the value of 0 or 1. Each packet in the network model has a velocity property with it. The property holds information of how the packet “move ahead” in the next time step. For the sake of simplicity, the maximum velocity of packet is limited to 1, that is, it must be taken 0 or 1.

Assume that the evolution rule of the network model is as follows:

1. Acceleration: For packets that are in the link agents or at the front of the queue of node agents, their velocities are set to be 1. The velocities of other packets remain unchanged.

2. Slowing down: The velocity of a packet will reduce to 0, if the packet satisfies the alternative below:
    - For any agent which holds one packet, the next adjacent one is a link agent and one packet is occupied in it.
    - For any agent which holds one packet, the next adjacent one is a node agent and the number of packet in it has reached the maximum.
3. Randomization: For each node agent, slow down the packet lined at the front of queue and set its velocity to 0 with probability p, if this agent is not null (or it holds agents).
4. Packet motion: Each packet goes ahead according to its velocity. If its destination agent is a node one, it will be added to the end of the queue.

The step 3, the randomization, should be attracted special attention. The randomization is regarded as the abstract of the forwarding packet process of switchers or routers in a real world network. Through the above steps, we can have dynamic behaviors of network.

Like the model in road traffic, the network model can be divided into the closed form and the open one in term of its boundary conditions. The closed model has periodic boundary conditions, that is, the agent at the right boundary is thought to be adjacent to the agent at the left boundary. The total number of packet in the network model with closed boundary conditions is not changeable and all the packets are moving circularly in the agents during a dynamical evolution. In contrast, all the packets could "enter" or "exit" at its boundaries in an open model system.

4. Conclusion

We presented in this paper a system of simple organisms interacting in a two-dimensional environment, which have the capacity to evolve. A process model is used here as a way to represent the action course of cooperative agents committed to a non-deterministic process.

The new model differs than cellular automata model in some points:

1. Our new model presents a general model for cellular automata system. Such that the set of transition rules that are evolved in our model can be applied to give any transition rule for any type of cellular automata.
2. Each cell of the grid is considered as a whole system or agent where our model has the ability that at any time step the agent may contain a different rule than other agent rules in the grid.
3. Evolution takes place not only in state space as in the cellular automata model, but also in the rule space.

It was presented as the applications of our model, the simulated traffic flow and computer network depending on our model. In traffic system it was observed that the overall dynamics are quite sensitive to the driver's behavior at road crossing for choosing its destination. Because the model uses processes, which are fully incremental, it is able to show how a car/environment interaction can modulate and guide an underlying evolutionary process. In The computer network model the simulation results show that it indeed captures some properties of flux of computer networks. We have reason to believe that this CA model will bring a new way to the study of complex behaviors of computer networks.

References

[1] Abul, O., Polat, F., and Alhajj, R., 2000, Multiagent Reinforcement Learning Using Function Approximation, IEEE Transaction on systems, man, and cybernetics-part c: application and reviews, Vol. 30, No. 4, pp. 485-497.

[2] Goldman, C., and Rosenschein, J., 2002, Evolutionary Patterns of Agent Organizations, IEEE Transactions on Systems, Man and Cybernetics-Part A: System and Humans, vol. 32, No. 1, pp. 135-148.
[3] Hassan Y. and Tazaki E., ”Emergence Computation using New Model of Cellular Automata”, Applied Artificial Intelligence, 17(1), 39-69, (2003).
[4] Koza, J.R., 1999, Genetic Programming III Darwinian Invention and Problem Solving, San Francisco, CA.
[5] Nagel K., “Traffic networks, in: S. Bornholdt, H. Schuster (Eds.)”, Handbook on Networks, 2002.
[6] Oliveira G. and Siqueira S., “Parameter characterization of two-dimensional cellular automata rule space”, Physica D, 217, 1-6, (2006).
[7] Ren Z., Deng Z., and Shuai D., “Behaviours of networks with different topologies and protocols”, Computer Phys. Comm., 141(2), 247, (2001).
[8] Salah, H., Hassan, Y., Badawi, O., and Selim, G., Multi-Agent System Communication Protocol Design using Cellular Automata, AAST international conference in computer and systems, 2005.
[9] Sipper, M., 1995, Studying Artificial Life Using a Simple, General Cellular Model, Artificial Life Journal, vol. 2, No. 1: pp 1-35.
[10] Weiss, G., Multiagent Systems, 1999, A Modern Approach to Distributed Artificial Intelligence, the MIT Press.