BUILDING INTELLIGENCE IN GAMING AND TRAINING SIMULATIONS

Dennis Jacobi[a], Don Anderson[a], Vance von Borries[a],
Adel Elmaghraby [b], Mehmed Kantardzic[b], Rammohan Ragade [b],
Quasim Mehdi[C], Norman Gough[C]
[a]Intellas Modeling and Simulation, LLC 15424 Beckley Hills Drive, Louisville, KY, USA;
[b]J. B. Speed Scientific School, University of Louisville, Louisville, KY, USA
[C]University of Wolverhampton, School of Computing and Information Technology,
35/49 Lichfield S Wolverhampton, WV1 1SB United Kingdom
djacobi@intellas.com

KEYWORDS
Simulation, Agents, Artificial Intelligence, Game, Particle swarm

ABSTRACT

Current war games and simulations are primarily attrition based, and are centered on the concept of "force on force." They constitute what can be defined as "second generation" war games. So-called "first generation" war games were focused on strategy with the primary concept of "mind on mind." We envision "third generation" war games and battle simulations as concentrating on effects with the primary concept being "system on system." Thus, the third generation systems will incorporate each successive generation and take into account strategy, attrition and effects.

This paper will describe the principal advantages and features that need to be implemented to create a true "third generation" battle simulation and the architectural issues faced when designing and building such a system. Areas of primary concern are doctrine, command and control, allied and coalition warfare, and cascading effects. Effectively addressing the interactive effects of these issues is of critical importance. In order to provide an adaptable and modular system that will accept future modifications and additions with relative ease, we are researching the use of a distributed Multi-Agent System (MAS) that incorporates various artificial intelligence methods. (Anderson 2002a, Anderson 2002b)

INTRODUCTION

The act of wargaming has been practiced for centuries as more of an art than a science. The first generation of wargames can best be described as "mind on mind" with such strategic abstractions as the game of chess. The second generation of wargames can best be described "force on force", with mathematically based attrition models as the engine for resolving an engagement. This paper formally introduces what has been called a Third Generation Wargame which uses the concept of "system on system" and that incorporates the interactive and cascading effects within each and between systems. The goal of a Third Generation system is to simulate as realistically as possible the variety of effects that military operations will have on a scenario. We use the term "scenario" to mean the strategic level of operations, including not only battle effects, but also other factors in the theater of operation such as morale, logistics, civilian support/unrest and refugees. The granularity in which to program such a complex system is one of the focuses of our research. The level of granularity that is sufficient to produce an effective and useable training and planning tool are one of the focuses of our research.

The use of a multi-agent (Sycara 1998; Weiss et al. 2000) architecture (MAS), with added intelligence was chosen as our platform for research for a number of reasons. Independent agents mirror the reality of the players in a theater of operations and allow for the interactive and cascading effects that occur. Agents can also be created at any level of granularity desired from an individual person to a large unit. Our research thus far has indicated that, even when the overall granularity level is at regimental level, certain key individuals do play important roles. Certain select key commanders, aces, and heroes can be made a part of the agent structure to influence the unit as a whole.

The purpose of our research is to design and build an architecture that will serve as a strategic training tool. The emphasis is on creating a tool. This is not a predictor of future action, but a tool in which strategic moves can be simulated to avoid major mistakes and provide, over several iterations, a set of boundaries in which the campaign is

likely to be conducted. Planners can conduct various "what if" exercises to measure reaction to their actions.

APPROACH AND METHODOLOGY

The approach that has been pursued is to take a portion of the North African Campaign during World War II from 8 November 1942 to 13 May 1943 and provide a well-researched historical campaign to test the accuracy of a computerized model. The initial version will have its agents "built out" with an increasing number of rules and attributes that will help determine actions it would take given a situation. Artificial intelligence and learning components will be added so that as a player has battle experiences it can learn to proceed differently the next time. The initial computerized version will be created by the development team to prove the methodology of the approach. A toolkit will later be created to provide trainers and planners the ability to place specific characteristics into generic agents to create their own scenarios. Key to the development is defining methods in which relationships between agents are coordinated in a multi-agent system, and who/how the relationships are created.

Agent Development

A set of generic agents will first be developed from which individualized characteristics can be added. Using object oriented technology concepts, an abstract generic agent will be defined, which has these characteristics (called attributes) and functions (called methods or member functions). Further, each agent has a knowledge base consisting of rules to act in given situations. This will be appropriate to handling the "what-if" situations. In addition, each agent has a learning component to handle learning specific tasks and having a learned response as a result. This component parallels training issues and subsequent performance.

The initial design of the agent-based system allows the integration of agent learning. The learning mechanisms will be fully designed and then adjusted based on our experimentation result. An intelligent agent will be created that will be given a minimum of background knowledge at the beginning (doctrine in the form of rules), and then learn appropriate "behavior" as it becomes more experienced. Initially, an agent will have in it attributes, methods, rules and a "blank" learning component. Attributes include the various assets in terms of personnel, equipment and capability. Methods are doctrinal based ways of operation. Rules provide the boundaries and limitations for the unit. Initial setting for the learning component may come from historical precedent, as in the case of a training system, or intelligence data, as in the case of a planning system. This would essentially bring the agents or "military unit" up to pre-battle readiness. Gradually, as the agent gains experience, more knowledge would be stored for decision-making. This learning approach presents a satisfactory solution to the *trust* and *competence problems* of intelligent agents. While the agent gradually develops its ability, the users of the system obtain more trust in the agent's decisions and actions. The generic design of the agent layers is show below in Figure 1.

ID
Attributes
Methods
Rules
Learning Component

Figure 1. Generic Agent Structure

Given this abstract agent concept which provides a template, specific agent types can be defined as a subclass (extension). This process may have further subclasses to desired levels, including one to one, many to one, one to many or many to many relations among agents as appropriate. Clearly, there may be an exponential growth in the complexity of these agents and their relationships. Hence agents will be aggregated at appropriate levels of granularity and the simulation will examine the relevant scenarios. (Davis 1995; Herz and Macedonia 2002; Smith 1998) Scenarios (theaters or circumstances) and transitions amongst the scenarios will determine which agents will be under consideration in the theater.

Mechanisms for learning by agents are discussed in Section 2.3. By acquiring knowledge from different sources, the agent gradually learns how to better execute the desired objective. Through incremental learning agents become more competent. As they accumulate knowledge about different situations they can handle them more successfully. Also, the agents can be trusted. (Palmer, Stone 2000)

Advantages of a Multi-agent Architecture

Four main characteristics describe the development of a military Multi-Agent System (MAS):

- *Real-time* domains are those in which success depends on acting in response to a dynamically changing environment.
- *Noisy* domains are those in which agents cannot accurately perceive the world, nor can they accurately affect it.
- *Collaborative* domains are those in which a group of agents share a common goal.

- *Adversarial* domains are those in which there are agents with competing goals.

Multi-agent systems in complex, real-time domains require agents to act effectively both autonomously and as part of a team. Focus should be placed on the problem of designing a collective of autonomous agents that individually perform sequences of actions such that the resultant sequence of *joint* actions achieves a predetermined global objective. The crucial design step in multi-agent systems centers on determining the private objectives of each agent so that as the agents strive for those objectives, the system reaches a desired global solution. Because of the inherent complexity of this type of multi-agent system, we will investigate the use of machine learning within multi-agent systems.

MAS takes care of the mechanics of executing actions controlled by an agent, passing messages between actions, coordinating multiple agents, arbitrating resource conflicts between agents, updating sensor values, and interleaving higher-level processes such as planning. When a group of agents in a multi-agent system share a common long-term goal, they can be said to form a team. Team members (or teammates) coordinate their behaviors by adopting compatible cognitive processes and by directly affecting each other's inputs via communicative actions. Other agents in the environment that have goals opposed to the team's long-term goal are the team's adversaries. The team member agent architecture within a flexible structure proposed, allows agents to decompose the task space into flexible roles and allows agents to smoothly switch roles while acting. The agents are assumed to have at their disposal the following resources:

- Inputs from the environment that give partial, noisy information;
- The ability to process the input information and use it to update a world model;
- Learning mechanisms that dynamically affect the model;
- Communication capabilities.

A MAS will also allow or even require distribution of the agents among either remote or co-located computer systems. The distributed system will allow for great flexibility in the processing power of the simulation, since computational tasks can be spread across multiple PC's. In an instructional setting, each command center could be run from a separate PC in geographically diverse locations.

An advantage of a distributed MAS is that each agent or groups of agents may be executed on different computers and in different operating systems (OS). Additional agents can also be added into the system without modifying existing agents. This creates a system that is more robust and less prone to failure. Agent classification is object oriented which allows the sub-classing of agents to match the hierarchical classifications of the agent families. The methodology of our design is to divide a large task into a lot of sub-tasks to allow each sub-task to be solved by different agents. This creates an easier programming task by allowing a greater number of less complex programs to be written.

One of the major issues in defining an action set for an agent, and, arguably, one of the major issues in defining any kind of intelligent behavior is the problem of forming abstractions. No agent designer will want to specify the solution to a given problem in terms of primitive low-level actions and sensations. Instead, the designer will first build more powerful abstract actions, which encode solutions to a range of problems, and use these actions when faced with a new problem. Our MAS should support abstraction by providing the mechanisms to construct a hierarchy of actions. In the hierarchy, abstract actions are defined in terms of simpler ones, ultimately grounding out in the agent's effectors. The very lowest level of the hierarchy consists of very primitive actions, like move or apply-force. Although actions are abstract at higher levels of the hierarchy, they are nonetheless executable. At the same time, the hierarchy implements a multi-level computational architecture, allowing us, for example, to have both cognitive and reactive actions within the same framework. This means that higher levels should provide goals and context for the lower levels, and lower levels provide reports and messages to the higher levels (goals down, knowledge up). A higher level cannot overrule the information provided by a lower level, nor can a lower level interfere with the control of a higher level.

The term "plan" is used to denote an action that satisfies a goal. More specifically, an activity plan usually begins with a system at some initial state, specifies some desired final or goal state, and identifies constraints on the allowable sequence of actions. Usually, military planning is a part of a five-stage process:

- Mission analysis
- Intelligence preparation of the battlefield
- Development of courses of action
- Analysis of courses of action
- Decision and execution

These steps rely on a detailed and extensive knowledge base of the domain, environment, enemy and friendly capabilities. Planning is necessary when the goal is satisfied by several actions and we have to decide between them. A planner's effectiveness is determined by the ability to cope with the complexities of a continuous, dynamic, real-time domain. The planner's distinguishing feature is that it evaluates plans by efficiently simulating ahead in a more abstract space.

Learning Approaches

There are three main learning approaches that are being used in our research: Machine Learning (ML), Reinforcement Learning (RL), and Artificial Neural Networks (ANN). Implementation of Machine Learning mechanisms is a promising area to merge with the inherent complexity of multi-agent systems. Central to the process of learning, is the adaptation of behavior in order to improve performance. ML has the potential to provide robust mechanisms that leverage upon experience to equip agents with a large spectrum of behaviors, ranging from individual performance in a team, to collaborative achievement of independently and jointly set high level goals. Using hierarchical task decomposition, multiple ML modules can be combined to produce more effective behaviors than a monolithic ML module that learns straight from inputs and outputs.
The approach will break the problem down into several behavioral layers and use ML techniques when appropriate. Given hierarchical task decomposition, layered learning allows updates at each level of the hierarchy, with learning at each level directly affecting learning at the next higher level. Starting with low-level behaviors, the process of creating new behavior levels and new ML subtasks continues towards high level strategic behaviors that take into account both teammate and opponent strategies.

A learning component acquires its competence from different sources and in different ways:

- *Observing and imitating the successful actions and decisions of other agents*
- *Receiving positive and negative feedback from the user and higher command*
- *Receiving explicit instructions from the user*
- *Communicating and obtaining advice from other agents in the*

Reinforcement Learning (RL) represents the second alternative for learning in our MAS. This is the branch of machine learning that is concerned with an agent who periodically receives “reward” signals from the environment that partially reflect the value of that agent's private utility function. The goal of an RL algorithm is to determine how, using those reward signals, the agent should update its action policy to maximize its utility.

The maturing field of Reinforcement Learning provides much-needed mechanisms for model free and “online" learning features. It is ideally suited for the distributed environment where a “teacher" is not available and the agents need to learn successful strategies based on “rewards" and “penalties" they receive from the overall system at various intervals. As the number of agents increases, the effects of any agent's actions will be swamped by the effects of other agents (noise), making the agent unable to learn well, if at all. In addition, agents cannot be used in situations lacking centralized calculation and broadcast of the single global reward signal. Complexity of synchronization and coordination increases exponentially. The problem is that the space of possible action policies for such systems is too big to be searched. (Chen et al. 2000, Mehdi, et al. 2002)

Artificial neural networks are a third promising approach in MAS learning. They provide a robust statistical learning process in noisy, uncertain, and dynamically changing environments, and therefore a possible solution for learning in war games. Many applications have shown that these networks have sufficient computational power to approximate a very large class of nonlinear functions; non-linearity is one of the main characteristics of complex military systems. Therefore, artificial neural networks offer great potential and power for developing intelligent agents with the inductive learning component based on previous experience. At the same time, these models are also very difficult to integrate into existing military applications. One of the important reasons and disadvantages of this approach is the requirement for a large number of cases (massive experience) to support significant improvements in the learning process. One method that can used to compensate for these disadvantages is the use of Case Based Reasoning. (Pal, et al. 2001; Kolonder 1993)

Case Based Reasoning

Military operations have a very strong theory and historical record. In domains with strong experience another advantage of case-oriented techniques is their ability to learn from historical cases. Gathering these cases may improve the systems ability to find suitable similar cases for current problems. Therefore, the knowledge of experts does not only consist of formalized rules and procedures, but of a mixture of doctrinal knowledge and experience. The arguments for case-oriented methods of learning in are as follows:

- Reasoning with cases corresponds with the training process of military commanders.
- Incorporating new cases means automatically updating parts of the changeable knowledge.
- Textbook knowledge and experience can be clearly separated in a knowledge base, but used together in solving new cases.

The essential benefit from the CBR approach for our system is that the methodology can be applied with a small, or limited amount of experience and incrementally develop the

performance adding more cases to the case base as they become available. The main argument is that users of our system, even the experts, may not have enough or correct historical knowledge/experience in every situation.

Most of the previous inductive learning methods require a significant amount of cases and situations to build the agent's knowledge. Therefore, it was decided to design learning agents in our system using Case Based Reasoning (CBR) methodology. Some of the characteristics of a domain that indicate that a CBR approach might be suitable include:

- Records of previously solved problems exist or they will be acquired.
- Historical cases are viewed as an asset that ought to be preserved.
- Remembering previous experiences is useful especially in a new non-established domain.
- Experience is at least as valuable as textbook knowledge.

Case based learning models are simple yet surprisingly successful in providing extremely good prediction for human behavior in a variety of military applications. Furthermore, the case-based approach provides a more complete account of learning phenomena than rule-based, neural network or reinforcement learning. Essentially, learning occurs in case-based models through storage of a multitude of experience with past problems. New problems are solved through the retrieval of similar past problems. However, it is very important to select the right assumptions about the retrieval of past examples if we want learning with appropriate quality. Each historical case is represented as a point in n-dimensional space, and multidimensional scaling is necessary to treat all dimensions with the same weights in the comparison process. On the other side, based on experience or formalized previous knowledge, some features should receive more weight or attention in the learning process. The similarity (S_i) between new case C and old one x_i (stored in the knowledge base of cases) is assumed to be inversely related to the distance d:

$$S_i = \exp(-d(x_i,C)^n)$$

Evidence for one hypothesis, such as "the presence of a military threat", is computed by summing of all of the activated samples that share the hypothesis:

$$E_1 = \text{sum}(S_i)$$

Evidence for alternative hypothesis, for example "the absence of a threat", is computed for alternative samples:

$$E_2 = \text{sum}(S_j)$$

The final decision is based on a comparison of the evidence for each hypothesis, and for example the possible parameter is a probability of choosing the first hypothesis E_1 over the second E_2:

$$P = E_1 / (E_1 + E_2)$$

While a flat case base is a common structure in most of the CBR applications, a hierarchical structure that stores the cases by grouping them can reduce the search process and increase its performance. There are no universal CBR methods suitable for every domain of application. The challenge in CBR for military wargames is to create methods that are suited for problem solving and learning in particular subject domains and for particular application environments.

Although case-based models have proven to be highly successful, there are some mainly theoretical problems we have to be aware of. These problems will be analyzed and solutions will be proposed in the implementation phase. Also, despite the obvious potential to the gaming world, it must be used carefully to avoid certain pitfalls such as:

- *Mimicking Stupidity* - copying a human strategy that is taught badly
- *Overfitting* - taught a certain section of a problem with a lot of details, and then expected to display intelligent behavior based on its local experience for the entire problem
- *Local Optimality* - a non-optimal solution is reached, in which any small change cannot improve performance of the system
- *Past Behavior* - the behavior that has been successful for the learning process in the past, is not useful any more

The practical approach for the learning process is to combine CBR mechanisms with methods of rule-based agent design. CBR retrieval is used to search for similar cases to support evidences for rule-based decisions obtained by other agents in a distributed system. A CBR part and a rule-base are applied in parallel, the results and the co-ordination of further steps is handled by meta-rules. Further investigation should be conducted with the implementation of a prototype of a Distributed Case-Based Learning System for multi-agent systems. In such a system, each of the agents has a partial and imperfect view of the problem-solving situation. This gives rise to a need for the agents to cooperatively access their case-bases to retrieve the "best sub-cases," and to support or to revise their decisions and actions. A specialized learning agent will have a task to

coordinate all these activities with a case base in a consistent manner, and to provide cases for other agents that are most useful for the present problem-solving situation.

Particle Swarm Technology

Strategic level games have traditionally used a variety of methods from a simple roll of the dice to complex game engines to resolve conflicts of opposing game pieces. This often results in players trying to "game" the system instead of trying to anticipate enemy actions as in a real conflict. Our architecture will use agents as major commands, but will explore the use of particle swarm technology for conflict resolution. Particle swarm allows a large number of individual pieces to be controlled by their commanding intelligent agent and use relatively simple rules for moving around terrain and enemy engagement. Resolution of competing goals within an adversarial domain occurs through a series of individual and group actions. An agent structure can model the command levels and carry out the planning functions of those headquarters, but the management of agents is not optimal for resolution of the conflicting goals. Resolution will be done at the smallest level possible because that is a reflection of the real word that we are attempting to model.

Particle swam algorithms are goal oriented in which the swarm is given a goal and each particle finds its path toward that goal. These goals are often thought of as static, but they can also be dynamic. The authors have experimented with particles detecting the changing edge of a cloud as it is moved by the wind. This experimentation provides a basis for the use of particles for conflict resolution in a dynamic environment. Under the guidance of its controlling agent, particles can have their immediate objective change based on what they encounter in their environment. Each particle will initially have simple rules to follow upon encountering a particle or group of particles from an opposing swarm. Each particle will have a value that represents its combat power and will be compared against the values of opposing particles. Within a defined engagement area, particles with the higher local combat power will proceed toward their main objective while those with the lower local value will be removed from the game board. The complexity of the rules and values of the particles can increase to take into account such factors as the level of supply, moral, experience, armament as well as a variety of other human and logistical factors.

Particle Swarm Optimization (PSO) defines each particle as a potential solution to a problem in D- dimensional space. Thus:

- Particle "i" is represented by: $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$
- Memory of "i's" previous best position: $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$
- Velocity of "i" along each dimension: $V_i = (v_{i1}, v_{i2,} \ldots, v_{iD})$

A new position for the particle is computed for each iteration by combining the P vector of the particle with the best fit for the local area designated "g" and the P vector of the current particle to adjust the velocity. The cognition component is the portion of the velocity influenced by the individual particle's previous best position.

Our experience with the variety of swarm models indicates that the Full Model with variations may be of greatest use. Modification to vary each unit along with the different classes of particle groups will provide a unique conflict resolution tool. Minar et. al. has conducted research regarding swarm and agent technologies similar to what we are proposing. We have also experimented successfully with the applications of particle swarm algorithms in imaging and more recently in robot mapping of hazardous environments. (Hardin, et. al, 2004)

CURRENT STATE OF AUTHORS' RESEARCH

Research and development activities that pertain to the development of this research has been completed to two related key areas:

- A MAS used for medical decision making
- A historically based strategic board game

A demonstration system for medical decision-making has been completed and tested. This provides the foundation to the agent architecture detailed above.

Research has also been completed on a historically based strategic board game. This research provides the historical validation foundation for a computer simulation. It also provides a game template that can provide a basis for the first round of computerization. Historical decisions, conditions and constraints can be used to "replay" history to test the accuracy of the system. The research team, with direction from the United States Air Force, has selected the Tunisian campaign of 8 November 1942 through 13 May 1943 as the template This topic was chosen because:

With these two initial steps completed, research can be conducted in adding a learning component into the MAS, and computerizing the historical scenario with the intelligent MAS architecture.

CONCLUSIONS

A concept of Third Generation Wargames is advocated using intelligent multi agent systems (MAS). The dynamic characteristics of military operations lend themselves well to a multi-agent system. The value of Case Based Reasoning along with learning mechanisms for (artificial) agents (which can be embedded in modern military hardware) is tremendous. The exploration of the use of swarm technology for the resolution of competing goals is an innovative and efficient method of modeling the large numbers of individuals and weapon systems on the battlefield. The benefits of a completed system is the ability to train leaders more effectively to meet the demands of a increasingly smaller military decision cycle. Providing realistic wargaming tools for military planners further aids shortening the decision cycle and creating more accurate plans and contingencies for military operations. (Miller 1997)

ACKNOWLEDGEMENTS

Support for the research that resulted in this paper has been provide from the National Aeronautics and Space Administration (NASA) with a Phase I STTR grant to Intellas and the University of Louisville, and from the US Air Force with a Phase I SBIR grant.

REFERENCES

Anderson, D., Belknap, M., Cui, X., Elmaghraby, A., Jacobi, D., Kantardzic, M., Ragade, R. (2002), "A Distributed Agent Architecture for Human Operations on Space," for the International Society for Computers and their Application 11th International Conference on Intelligent Systems on Emerging Technologies, Boston, 2002.

Anderson, D., Belknap, M., Cui, X., Elmaghraby, A., Jacobi, D., Kantardzic, M., Ragade, R. (2002), "Computer Assisted Space Medic," World Space Congress, Space Ops 2002, Houston, Texas, 2002.

Pew, Richard and Mavor, Anne, editors, *Modeling Human and Organizational Behavior: Application to Military Simulations*, National Academy Press, 1998.

Sycara, K., "Multiagent Systems," *AAAI AI Magazine,* Vol. 19, No. 2, 1998.

Chen, J.R., Wolfe, S.R., and Wragg, S.D. "A Distributed Multi-Agent, System for Collaborative Information Management and Sharing," Proceedings of the 9th ACM International Conference on Information and Knowledge Management, 2000, 382-388.

Davis, P. K. "Distributed Interactive Simulation in the Evolution of DoD Warfare," Modeling and Simulation. Proceedings *of the IEEE*. Vol 83, No 8, 1995.

Herz J.C. and Michael R. Macedonia**, "**Computer Games and the Military: Two Views," Defense Horizons, April, 2002.

Kolodner J., *Case-based Reasoning*., Morgan Kaufmann, San Mateo, 1993.

Lenz M. et al., *Foundations to Applications*, Case-Based Reasoning Technology, Springer, Berlin, 1998, pp. 273-297.

Mehdi, Q., Gough, N., Sulliman, H., "Virtual Agent Using a Combined Cognitive Map and Knowledge Base System," for the International Society for Computers and their Application 11th International Conference on Intelligent Systems on Emerging Technologies, Boston, 2002.

Mehdi, Q., Gough, N., Wen, Z., "A New Approach for Animating Intelligent Agents in Complex 3D Virtual Environment Based on Spatial Perception and Memory," for the International Society for Computers and Their Application 11th International Conference on Intelligent Systems on Emerging Technologies, Boston, 2002.

Miller, C.B., "USAF TACS Battle Management: Preparing for High Tempo Future Operations," http://www.fas.org/irp/eprint/milis.htm, 1997

Pal S. K., Dillon T. S., Yueng S. Y., *Soft Computing in Case Based Reasoning*, Springer-Verlag, London, 2001.

Palmer N., *Machine Learning in Games Development,* http://ai-depot.com/GameAI/Learning.html

Smith, R. D., *Military Simulation Techniques & Technologies*. Distributed Simulation Technology, 1998

Stone P., *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer* Chapter Layered Learning in Multi-Agent Systems , MIT Press , 2000.

Weiss, Gerhard, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence,* The MIT Press, 2000.