Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

EFFICIENT AGENT-BASED SIMULATION FRAMEWORK FOR MULTI-NODE SUPERCOMPUTERS

Toshihiro Takahashi
Hideyuki Mizuta

Tokyo Research Laboratory
IBM Research
Shimotsuruma 1623-14, Yamato-shi
Kanagawa-ken 234-8502, JAPAN

ABSTRACT

In recent years the importance of a large-scale Agent-Based Simulation(ABS) that can handle large complex systems is increasing. We developed a large-scale ABS framework on BlueGene, which is a multi-node supercomputer. The ABS processes the agents' communications. When the number of transmissions among the agents is large, the transmission costs seriously affect the performance of the simulation. It is possible to reduce the amount of transmission among the nodes by clustering the agents which communicate heavily with each other. Assuming that an agent is a graph node, and that a data transmission between agents is a graph edge, this problem can be formulated as a Maximum-Flow and Minimum-Cut Problem. In this paper we present an efficient algorithm to find an approximate solution. Our algorithm is reliable, simple, and needs little computation. We demonstrate its beneficial effects with some experiments.

1 INTRODUCTION

There are various methodologies used to understand various phenomena in the real world. Mathematical analysis, continuous simulations like the finite element method (which is very popular in physics), and Discrete-Event Simulation are some of these methodologies. Agent-Based Simulation (ABS) is receiving attention as a very powerful tool to understand various kind of complex systems. In recent years the importance of large-scale ABS systems that can handle phenomena such as a complex social simulation, workforce simulation, and so on is increasing.

There is substantial public research on and development investment in ABS. These systems include Repast by the University of Chicago and Argonne National Laboratory, Swarm by the Santa Fe Research Institute, MASON by George Mason University, and SOARS by the Tokyo Institute of Technology, among others. In particular, MASON and SOARS focus on large-scale ABS. MASON is designed to handle millions of agents. SOARS aims to support PC clusters and grid computers for large-scale ABS.

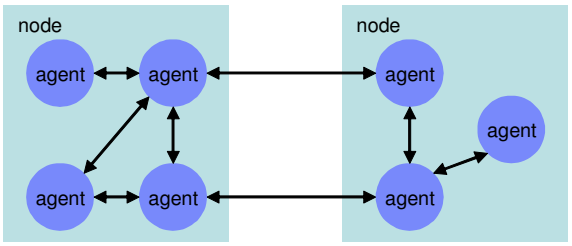

Figure 1: The Design of Our ABS Framework

In this paper, we describe our prototyping of a simulation system that can handle a large number of agents on a large computer system, such as a multi-node supercomputer or a grid computer.

2 ARCHITECTURE OF FRAMEWORK FOR A LARGE-SCALE ABS

We developed a large-scale ABS framework on BlueGene, which is a multi-node supercomputer from IBM. This system consists of many nodes. Each node has CPUs, memory, and other resources as a minimum computation unit. These nodes are connected to each other with fast networks such as gigabit Ethernet.

Figure 1 shows the design of our ABS framework. In our framework, each agent has a procedure and some variables. Each node handles a subset of all of the agents. Each agent communicates with the same node's agents by using shared memory, and each agent communicate with other nodes' agents by using sockets (over a fast network such as gigabit Ethernet). There is a large difference in communications speed between the same node's agents and other nodes' agents. ABS works through these communications between agents.

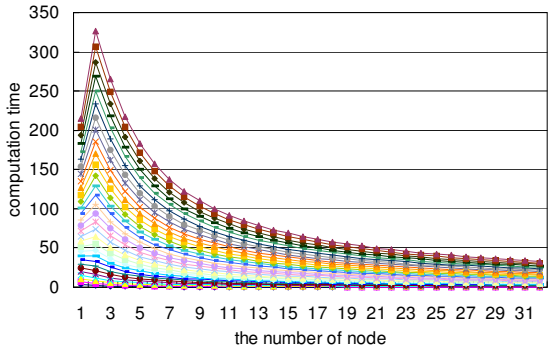

Figure 2: Estimated Computation Times

3 PERFORMANCE BOTTLENECK

Under the general design in which each node handles a subset of all of the agents, transmissions occur among the nodes. When the number of transmissions among the agents is large, the transmission costs seriously affect the performance of the simulations. In a general ABS, there are few situations in which all of the agents communicate with all of the other agents. The assumption that each agent communicates only with certain agents doesn't limit the problems that can be simulated. It is possible to reduce the amount of data transmission among the nodes by clustering the agents which communicate heavily with each other.

Agents dynamically change their communication partners as a simulation runs. We need an iterative clustering algorithm to keep communication costs low in cases like this.

Figure 2 shows the roughly estimated computation times for ABS based on measured results from BlueGene's network performance. We assumed that a single step of computation for an agent is 2 millisecond and one unit of transmission from an agent is 1 byte per time step. The x-axis shows the number of nodes and the y-axis shows the estimated computation times. The lower curve is for 100,000 agents, and the upper one is for 3,000,000 agents. For one node, transmission occurs only inside that node. When there are two nodes, transmission occurs between those nodes. Two nodes takes a longer time than one node in spite of the assumption that the amount of transmission by each agent is only 1 byte per time step. We see that transmissions among nodes seriously affect the performance of simulations.

There has been research on load-balancing of network costs in distributed systems. Bononi et al. investigated a load-balanced parallel and distributed simulation system. Their practical experience demonstrated that a speed-up in the simulation of parallel and distributed systems is achievable. Theodoropoulos and Birmingham proposed a load-balanced Interest Management mechanism on a distributed system.

Assuming that an agent is a graph node, a data transmission between agents is a graph edge, and if the number of nodes in a multi-node supercomputer is two, then this problem can be formulated as a clustering problem that divides these graph nodes into two clusters to minimize the total number of graph edges spanning both clusters. This problem is called the Maximum-Flow and Minimum-Cut Problem. The well known Push-Relabel algorithm for the Maximum-Flow-Minimum-Cut Problem runs in $O(n^2 \times m^{1/2})$ time, where n is the number of graph nodes and m is the number of graph edges. This complexity is unrealistic for millions of agents. In addition, sophisticated algorithms which cluster the graph nodes into three or more clusters have been little investigated.

4 OPTIMIZATION OF TRANSMISSION COSTS

The problem tackled here is how to minimize the amount of transmission among nodes. Our simple idea is that each node observes how many messages each agent that belongs to the node sends to other nodes, and exchanges subsets of agents with the other nodes based on this observed information.

A is defined as a set of agents. C is defined as a set of nodes. $B_k \subset A$ is defined as the subset of agents which belongs to node $k \in C$. $S(i,k)$ is defined as the number of transmissions between agent $i \in A$ and node $k \in C$. The procedure for node k is as follows:

- Record $S(i,k')$ for $\forall i \in B_k$ and $\forall k' \in C$.
- Obtain $\Delta R(i,k,k')$ for $\forall i \in B_k$ and $\forall k' \in C$, where $\Delta R(i,k,k')$ is defined as

$$\Delta R(i,k,k') = S(i,k) - S(i,k'). \qquad (1)$$

  $\Delta R(i,k,k')$ represents the difference in the number of transmissions among the nodes when the agent i moves from node k to k'.
- Obtain $k'_{max}(i,k) \in C$ for $\forall i \in B_k$, where $k'_{max}(i,k)$ is defined as

$$k'_{max}(i,k) = \underset{k'}{\operatorname{argmax}}\, \Delta R(i,k,k'). \qquad (2)$$

- Add i to $Bpool(k, k'_{max}(i,k)) \subset A$ for $\forall i \in B_k$, where $Bpool(k, k'_{max}(i,k))$ is a set of candidates of agents to be sent to node $k'_{max}(i,k)$
- Set $n(k,k')$ for $\forall k'$, where $n(k,k')$ is the number of agents to send from node k to k'. We set the value for $n(k,k')$ in advance here. Usualy this number is relatively small to the number of agents which each node handles. Only if the number of agents which belong to $Bpool(k,k')$ is less than

this predetermined number, $n(k, k')$ is set to the number of agents which belong to $Bpool(k, k')$.
- Receive $n(k', k)$ and obtain $m(k, k')$ for $\forall k'$, where $m(k, k')$ is defined as

$$m(k, k') = min(n(k, k'), n(k', k)). \quad (3)$$

- Send the top $n(k, k')$ of agents from $Bpool(k, k')$ to k' for $\forall k'$.
- Receive agents from k' for $\forall k'$.

Each node executes these steps continuously. The amount of data transmitted among nodes decreases gradually.

5 BEHAVIOR OF OUR ALGORITHM

We modeled some typical agent behaviors that appear frequently in ABS and experimented with these models.

5.1 Basic Model

The first experimental model used the following parameters:

- The number of nodes is two.
- 1,000 agents are randomly allocated on $[0, 1) \times [0, 1)$ in two dimensional space.
- Each agent sends messages only to agents within a distance of 0.1.
- The size of an agent's message is 1 byte.
- The maximum number of agents to exchange is 1.

Figure 3 shows the status of agents allocated in two dimensional space. As the simulation progresses, we can see that agents are divided into two nodes. Figure 4 shows the amount of data transmitted. The amount of data transmitted between the two nodes decreased from 8,997 bytes to 1,132 bytes. The amount of data transmitted within each node increased from 8,964 and 9,034 bytes to 15,912 and 17,816 bytes. It took about 220 time steps to converge.

5.2 Multi-Exchange Model

The second model is almost the same as the basic model, but the maximum number of agents to exchange is 5.

Figure 5 shows the status of the agents. Figure 6 shows the amount of data transmitted. The amount of data transmitted between the two nodes decreased from 8,997 bytes to 1,059 bytes. It takes about 50 time steps to converge. In this case, the speed of convergence is about five times faster than for the first model. We can see that the speed of convergence can be adjusted by changing the maximum number of agents to be exchanged.

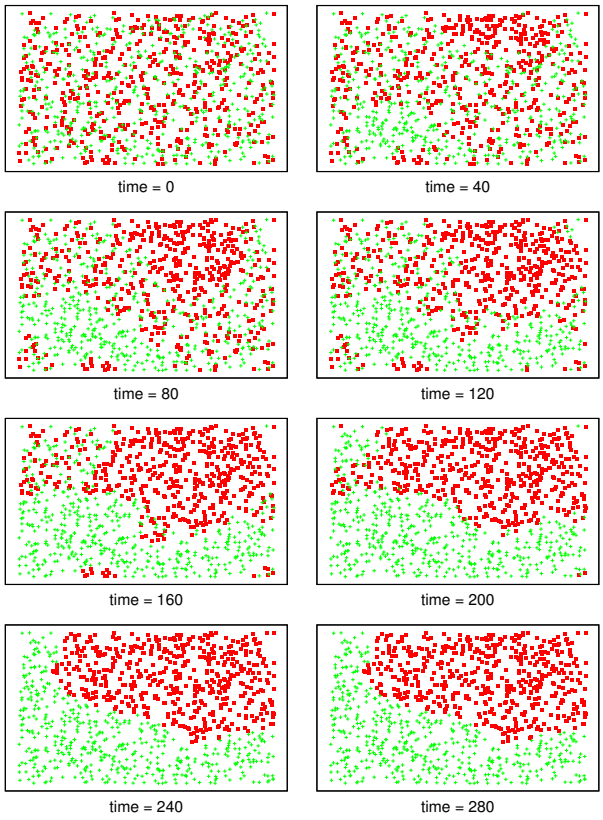

Figure 3: The State of Agents in the Basic Model

5.3 Multi-Node Model

The third model is almost the same as the basic model, but the number of nodes is 4. Many multi-node supercomputers have three or more nodes. We investigated behavior of our algorithm when there are three or more nodes.

Figure 7 shows the status of the agents. Figure 8 shows the amount of data transmitted. The amount of data transmitted between nodes decreased from about 2,100 bytes to about 500 bytes. We see good results when the number of nodes is 4.

5.4 Random Walk Model

The fourth model is almost the same as the multi-exchange model, but each agent also moves a short distance in a random direction in each step. Agents slowly change their communication partners.

Figure 9 shows the status of the agents. Figure 10 shows the amount of data transmitted. We can see that the amount of data transmitted is kept low in spite of the agents' movements.

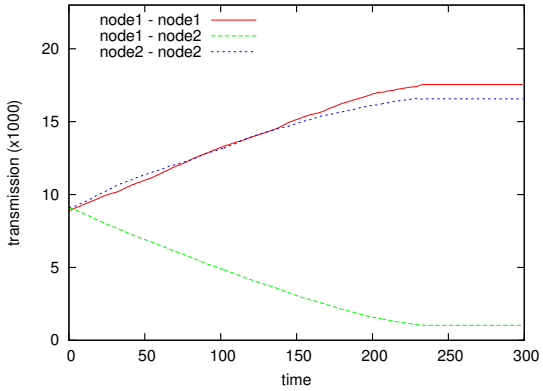

Figure 4: The Amount of Data Transmitted in the Basic Model

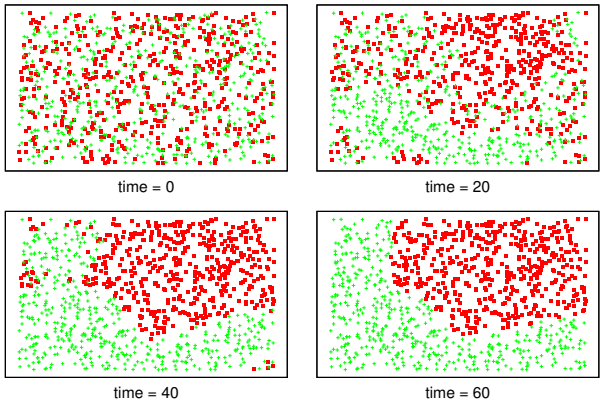

Figure 5: The State of Agents in the Multi-Exchange Model

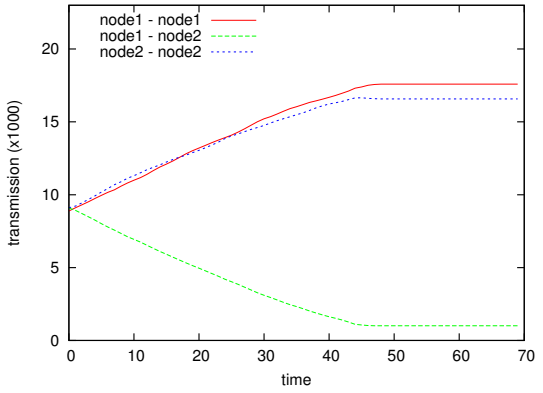

Figure 6: The Amount of Data Transmitted in the Multi-Exchange Model

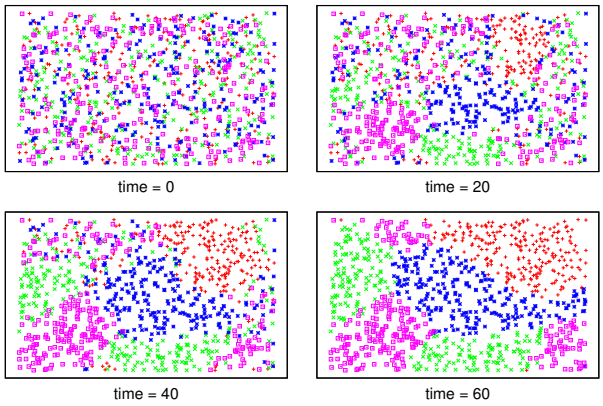

Figure 7: The State of Agents in the Multi-Node Model

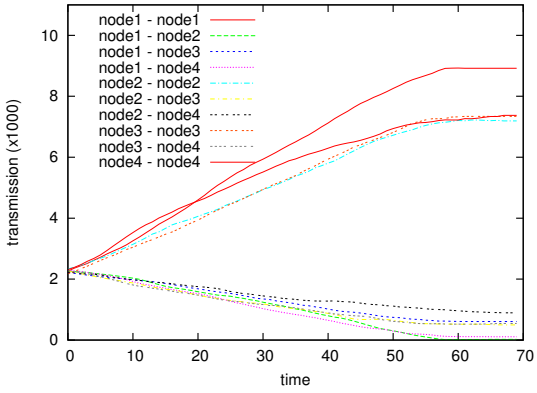

Figure 8: The Amount of Data Transmitted in the Multi-Node Model

6 ONLINE AUCTION MODEL

The fifth model is a more practical model, based on a dynamic online auction model using ABS that was developed by Mizuta and Steiglitz. We ported this online auction model into our framework and investigated the effectiveness of our algorithm.

In our online auction model some auctions start at random and each auction is open for a certain period. Bidders can participate in multiple auctions at the same time, watch the auctions' statuses and bid according to their strategies. In our framework we designed both the auctions and the bidders as agents, so the transmissions occur as agents for the bidders to communicate with the auction agents. The number of nodes was two.

Figure 11 shows the status of the agents. A box shows an auction, and an oval shows a bidder. Each bidder participating in an auction is connected to that auction by an

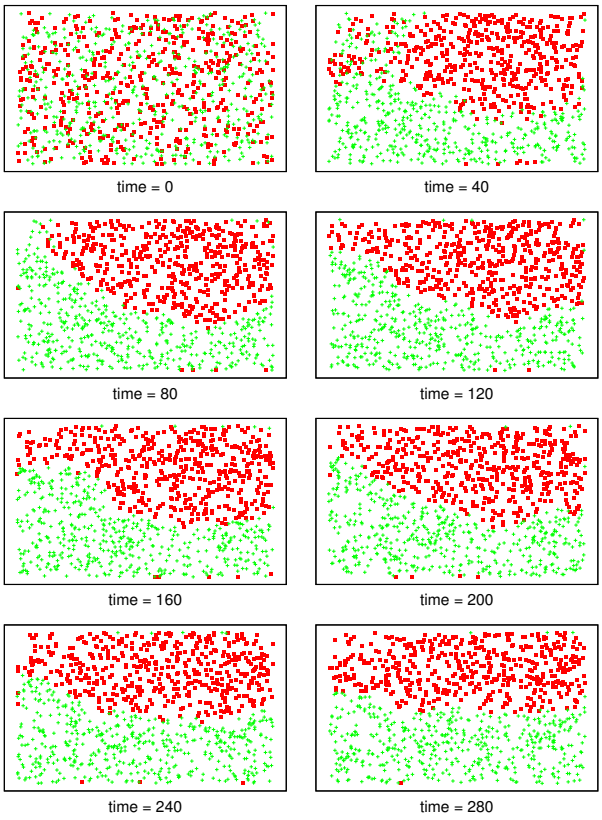

Figure 9: The State of Agents in the Random Walk Model

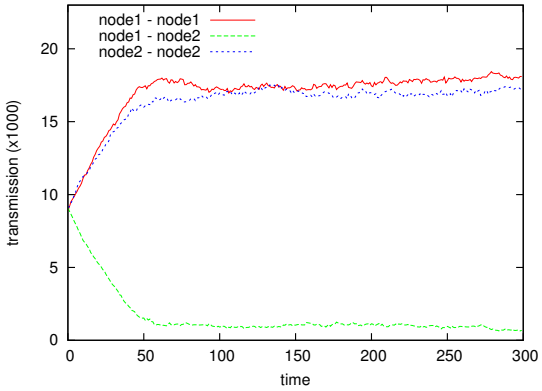

Figure 10: The Amount of Data Transmitted in the Random Walk Model

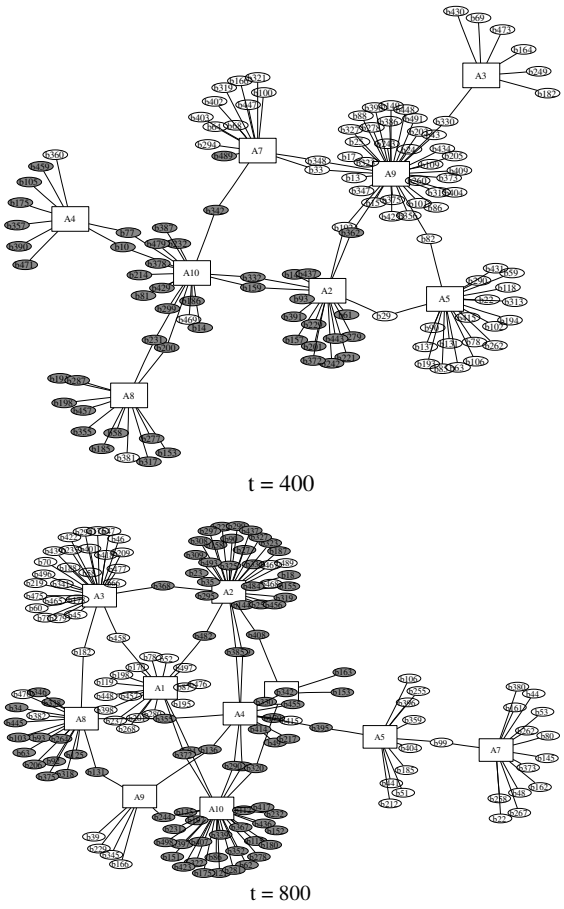

Figure 11: The State of Agents in the Online Auction Model

edge. In this figure, some agents participate in only one auction, and others participate in two or more auctions. The light bidders belong to node 1 and the dark ones belong to node 2. We can see that the agents participating in the same auction tend to belong to the same node. The agents are effectively separated into two nodes, even though this graph changes as the simulation progresses.

Figure 12 shows the amount of data transmitted when our algorithm is used. Figure 13 shows the amount of data transmitted without our algorithm. We can see that the amount of data transmitted is kept low in spite of a change in the agents' behaviors.

7 CONCLUSION

We had quite positive results with a very fast executable algorithm that obtains an approximate solution for graph clustering. Our iterative algorithm is simple and needs little computation time. Each node observes the transmissions of only its own agents and needs no complex negotiations with other nodes. The implementation of this algorithm is quite easy. These features are great advantages.

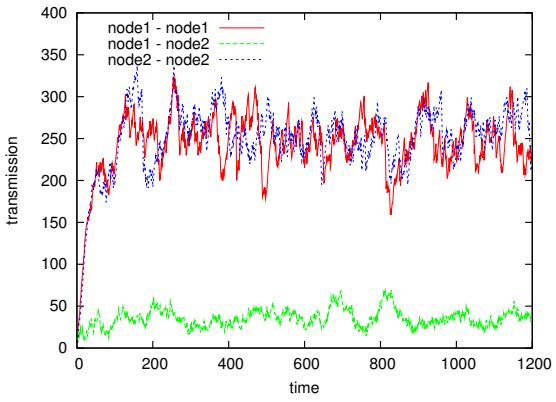

Figure 12: The Amount of Data Transmitted in the Online Auction Model (with Our Algorithm)

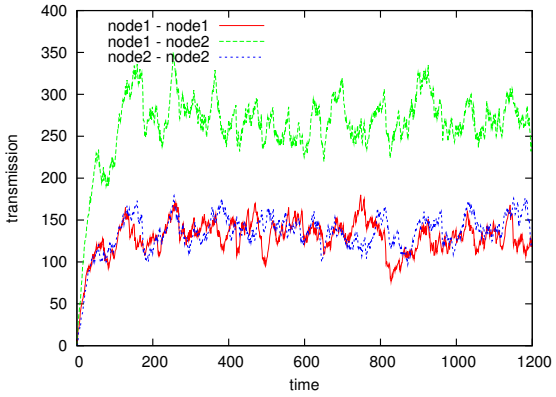

Figure 13: The Amount of Data Transmitted in the Online Auction Model (without Our Algorithm)

This algorithm is expandable to multiple clustering problems without any difficulties.

The situation when agents slowly change their communication partners is rarely seen in generic graph clustering problems, but appears frequently in ABS. This algorithm is very efficient in this situation. If the speed of agents' movements is variable, we maight be able to handle this problem by adjusting the maximum number of agents to exchange or by increasing the step size of this algorithm.

REFERENCES

Banicescu, I., and V. Velusamy. 2002. Load Balancing Highly Irregular Computations with the Adaptive Factoring. *Proceedings of the 16th International Parallel and Distributed Processing Symposium*.

Bononi, L., G. D'Angelo, and L. Donatiello. 2003. HLA-based adaptive distributed simulation of wireless mobile systems. *Proceedings of the 17th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation*.

Bononi, L., M. Bracuto, G. D'Angelo, and L. Donatiello. 2004. A New Adaptive Middleware for Parallel and Distributed Simulation of Dynamically Interacting Systems. *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real Time Applications*.

Chavez, A., A. Moukas, and P. Maes. 1997. Challenger: A multiagent system for distributed resource allocation. *Proceedings of the First International Conference on Autonomous Agents*.

Deguchi, H., H. Tanuma, and T. Shimizu. 2004 SOARS: Spot Oriented Agent Role Simulator - Design and Agent Based Dynamical System -. *Proceedings of the Third International Workshop on Agent-based Approaches in Economic and Social Complex Systems*, 49-56.

Goldberg, A. V., and S. Rao. 1997. Beyond the flow decomposition barrier. *Proceedings of the 38th IEEE Annual Symposium on Foundations of Computer Science*, 2-11.

Goldberg, A. V., and R. E. Tarjan. 1988. A new approach to the maximum flow problem. *J.Assoc.Comp.Mach.*, 35:921-940.

Logan, B., and G. Theodoropoulos. 2000. Dynamic interest management in the distributed simulation of agentbased systems *Proceedings of the Tenth Conference on AI, Simulation and Planning*, Society for Computer Simulation International and ACM SIGSIM, pp. 45–50.

Logan, B., and G. Theodoropoulos. 2001. The distributed simulation of multiagent systems *Proc. of the IEEE*, Vol.89, Issue 2.

Mizuta, H., and K. Steiglitz. 2000. Agent-Based Simulation of Dynamic On-Line Auctions. *Proceedings of the 2000 Winter Simulation Conference*.

Nowe, A., and K. Verbeeck. 1999. Distributed Reinforcement learning, Loadbased Routing a case study. *Proceedings of the Neural, Symbolic and Reinforcement Methods for sequence Learning Workshop*.

Parent, J., K. Verbeeck, and J. Lemeire. 2002. Adaptive Load Balancing of Parallel Applications with Reinforcement Learning on Heterogeneous Networks. *DCABES*.

Schaerf, A., Y. Shoham, and M. Tennenholtz. 1995. Adaptive Load Balancing: A Study in Multi-Agent Learning. *Journal of Artificial Intelligence Research*.

MASON. [Online]. Available: ⟨`http://cs.gmu.edu/~eclab/projects/mason/`⟩.

Repast. [Online]. Available: ⟨`http://repast.sourceforge.net/`⟩.

Swarm. [Online]. Available: ⟨`http://www.swarm.org/`⟩.

AUTHOR BIOGRAPHIES

TOSHIHIRO TAKAHASHI is a researcher at the Tokyo Research Laboratory of IBM Japan. He received a B.S. degree in mathematics from Waseda University, and a M.S. degree in computer science from Waseda University. His e-mail address is ⟨e30137@jp.ibm.com⟩.

HIDEYUKI MIZUTA is a researcher at the Tokyo Research Laboratory of IBM Japan. He received B.S., M.S. and Ph.D. degrees in physics from the University of Tokyo. He is a member of IPSJ and ACM SIGSIM. His research interests include dynamic socioeconomic systems with heterogeneous agents. His e-mail address is ⟨e28193@jp.ibm.com⟩.