Multi-Agent Oriented Distributed Computing Optimized by DS-GA

Koichi NAKAYAMA*†, Hirokazu MATSUI‡, Katsunori SHIMOHARA† and Osamu KATAI*
*Graduate School of Informatics, Kyoto University, Kyoto, Japan
†ATR Network Informatics Laboratories, Kyoto, Japan
‡Faculty of Engineering, Mie University, Mie, Japan
Email: knakayama@atr.jp, hmatsui@robot.mach.mie-u.ac.jp, katsu@atr.jp, katai@i.kyoto-u.ac.jp

***Abstract*— In this paper, we propose the Multi-Agent Oriented Distributed Computing (MAO-DC) approach to efficiently using the surplus resources of personal computers (PCs) connected to a network such as the Internet.**

Agents of MAO-DC transfer from one PC to the next through a network sequentially, and they process distributed tasks by using the surplus resources of the PCs. As an optimization algorithm for MAO-DC, we use Dynamically Separating GA (DS-GA), which is one of the effective learning algorithms for a multi-agent system. By applying DS-GA, an agent that senses not global information but local information can obtain the global optimality. In other words, DS-GA has a feature that allows "Swarm-Sensing" to emerge in DS-GA.

We verified experimentally that MAO-DC is optimized by DS-GA, thus maximizing the collective performance of the agents that use the surplus resources in MAO-DC.

I. Introduction

In recent years, the use of personal computers (PCs) has become widespread, and the ability of these PCs is increasing every year. However, the collective capability of these PCs is not fully used. For example, many computers do not work in the place of the night on the earth. Accordingly, various types of distributed computing have been studied for effectively using these surplus computer resources on networks [1][2]. Some of them are already in use [3][4].

Almost all the conventional research on distributed computing is based on a task transfer, where a host computer passes distributed tasks to each computer and then receives the results of the tasks. In other words, distributed computing typically involves master-slave computations in wide-area distributed environments. In this paper, we aim at optimization of load balancing that is realized by scheduling management of task-processing, in an extended distributed computing that can use the surplus resources among PCs without task control by a master computer. Then, we consider an extended situation, where distributed tasks are transferred from one PC to the next through a network sequentially to process the tasks. We focus on a multi-agent-oriented approach [5] [6] for the above type of distributed computing, and thus we propose Multi-Agent Oriented Distributed Computing (MAO-DC). MAO-DC realizes distributed computing by many software agents. Each agent holds a distributed task that consists of programs to process and data to be processed. It transfers from one PC to the next sequentially, and executes the programs with the data for processing a distributed task by using the surplus resources of the PCs. In MAO-DC, the load balancing is determined by the state-action strategy included by each agent. In this paper, the state-action strategy is represented by a transference table that describes the transference PC according to the current state.

In a large-scale network such as the Internet, it is difficult to grasp all of the characteristics of a network or its attached computers. Furthermore, these characteristics are various and change dynamically. Applying a learning algorithm of agents that only have local information for global optimization is an effective approach under the above environment. Therefore, we apply the Dynamically Separating Genetic Algorithm (DS-GA)[7] as a learning algorithm of agents based on local information to MAO-DC. DS-GA is the optimization algorithm for a multi-agent system. DS-GA has the ability to increase system-level optimality by autonomous learning of agents based on the local information. In short, "Swarm-Sensing," which is defined below, emerges from agent-sensing in DS-GA.

Swarm-Sensing: Swarm Intelligence is defined as any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of the social insect colonies and other animal societies [8]. On the other hand, Swarm-Sensing is the property of swarmed simple units and their smart cooperation. Upper-class sensing emerges from collective simple sensing. Even if certain information cannot be sensed by a simple unit alone, the swarmed units with Swarm-Sensing can sense the information by the local interaction of units. Swarm-Sensing provides a basis for exploring distributed problem solving without centralized control or global information.

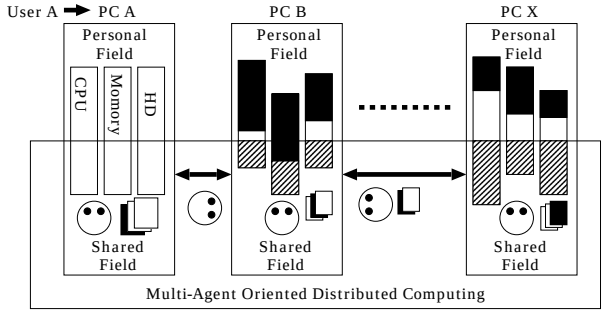
In DS-GA, agents are separated into colonies, and an agent has limited interaction within the colony. By this separation, System-level information emerged from collective agent-level information. By applying DS-GA, we have conducted research on the optimal route acquisition [9] and the optimal task distribution [10] for using surplus resources of PCs on a network. However, we have not taken into consideration the restriction of network resources in this research. Particularly in a large-scale network such as the Internet, not computer resources but network resources become the bottleneck of distributed computing in many cases.

In this paper, we verify experimentally that the total amount of task-processing in the entire MAO-DC is maximized by DS-GA even if network resources are unknown and change dynamically.

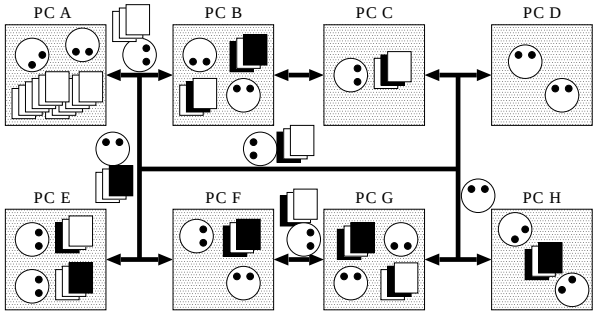
II. Multi-Agent Oriented Distributed Computing (MAO-DC) with DS-GA

A. Outline of MAO-DC

PCs and network links on a network provide a MAO-DC with their arbitrary surplus computer and network resources (Fig. 1), and the type or quantity of these resources need not be uniform.



(A) Conceptual representation of Shared Field in MAO-DC



(B) Conceptual representation of Network in MAO-DC

Fig. 1. Multi-Agent Oriented Distributed Computing (MAO-DC).

MAO-DC has many transference agents to process the distributed tasks. An agent executes the programs with the data in using the surplus computer resources of the current PC. Unlike a computer virus, an agent works only on the provided surplus computer resources. The agent acts according to its state-action strategy. In this paper, since the actions on the state-action strategy are restricted to transference to the next PC according to its current state, only a transference table is used as a state-action strategy.

In this model, the determination of the task-processing scheduling is equivalent to the determination of the transference tables of all agents. Therefore, in order to maximize the total quantity of task-processing in MAO-DC, the transference table of each agent should be optimized.

B. Outline of DS-GA

We apply DS-GA as a learning algorithm to determine the transference table of each agent. The basic idea of the DS-GA is as follows. The DS-GA separates agents into colonies. An agent cannot contact any agent in the other colonies. An agent with a high evaluation value is split into two agents, and an agent with a low evaluation is extinguished. A colony changes dynamically according to the number of agents in the colony. A colony is divided into two halves when the number of agents in the colony increases, a colony is extinguished when the number of agents it contains becomes 0. Each agent can transfer to another colony according to a certain migration probability. In this paper, a channel corresponds to a colony. Figure 2 shows a schematic representation of the channel.

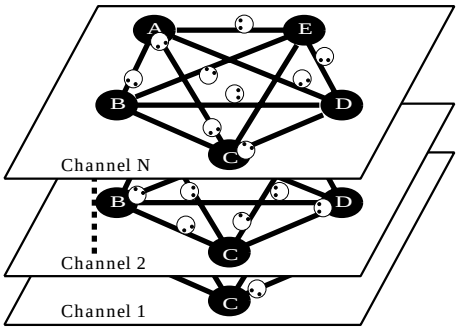


Fig. 2. Conceptual representation of channels.

By acting on the above dynamic separation, DS-GA permits an agent with only local information to learn the global optimality.

C. Implementation to MAO-DC of DS-GA

We apply DS-GA to optimize the transference tables of the agents in the MAO-DC, since swarm-sensing emerges from agent-sensing in DS-GA. To apply DS-GA to MAO-DC, we add a self-evaluation value and a channel parameter to each agent in MAO-DC. Agent a consists of not only a transference table $Tab_{transference}(a)$ for a part of MAO-DC, but also a self-evaluation value $E_A(a,t)$ and a channel parameter for a part of DS-GA. The learning algorithm for MAO-DC with DS-GA is as follows.

(1) Initialization: We define the surplus computer resources provided to MAO-DC as the shared field. The shared field is separated into $N_C(0)$ channels. The N_{Lim} agents are created in each channel. The self-evaluation value of agent a is $E_A(a,0)$, and its transference table $Tab_{transference}(a)$ is initially randomly chosen.

(2) Processing algorithm by MAO-DC:

(2-1) Task acquisition: An agent is passed a distributed task by a PC, when the PC has the task and the agent has no task.

(2-2) Agent transference: An agent transfers to the next PC with the task according to the transference table.

(2-3) Task processing: An agent processes a task in the separated shared field for the channel.

(2-4) Task completion: the PC receives the completed task from the agent when the agent has the completed task.

(3) Learning algorithm by DS-GA:

(3-1) Increase and Decrease of Agent's Evaluation: The evaluation value of an agent increases by some value (10) when the agent completes the task. The evaluation value of an agent decreases by some value (0.5) at every time step.

(3-2) Split and Extinction of Agents: An agent is split into two agents when the evaluation value becomes more than twice the initial value $(E_A(a,0))$. The two agents inherit half of the original agent's value. A transference table is mutated according to the mutation probability P_{mut}. An agent is extinguished when the evaluation value becomes less than zero.

(3-3) Migration of Agent: An agent migrates to a randomly chosen channel according to the migration probability P_{mig}.

(3-4) Dynamic Separation of Channels: When the number of agents in a channel exceeds the limit N_{Lim}, the agents are separated into two channels of almost equal size.

(3-5) Elimination of Channel: All of the agents in a channel chosen at random are eliminated with the channel when the total number of channels in MAO-DC becomes greater than the initial $N_C(0)$.

(4) Time Step Loop: Time step t is added 1 by a complete run from step (2) to (3). The above steps are performed until a stop criterion is reached.

For the experiments in this paper, the number of initial channels was set to $N_C(0) = 50,000$, the limit number of agents in a channel was set to $N_{Lim} = 20$, the mutation probability $P_{mut} = 0.05$, the migration probability $P_{mig} = 0.01$, and the initial evaluation value $E_A(a,0) = 10$ for all agents. Furthermore, 100 runs were performed with different random seeds.

III. Verification of Swarm-sensing

MAO-DC needs information on all of the surplus network resources in order to use the surplus network resources optimally. However, each agent cannot perceive all of the information. We apply DS-GA to MAO-DC, since the swarm-sensing feature it provides allows each agent to perceive all information.

This section shows experimentally that each agent acquires the optimal transference table for the whole by using the swarm-sensing capability of DS-GA.

A. Experimental model

In order to investigate "Swarm-sensing" in a MAO-DC, we conducted experiments on the simple network model shown in Fig. 3.

The network model consists of four PCs, $PC_{\{A,B,C,D\}}$, and five network links. The four PCs have different characteristics of surplus computer resources, where PC_A can requests a task and $PC_{\{B,C,D\}}$ can process a task. $PC_{B,C}$ can execute Program1 but cannot execute Program2. PC_D can execute Program2 but cannot execute Program1. Five network links

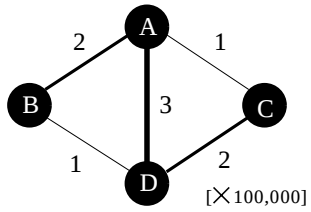


Fig. 3. Conceptual representation of Network Structure and Surplus Network Resources. The number in the figure expresses the quantity of surplus network resources.

have different quantities of surplus network resources. Network links AB, AC, AD, BD and CD have limited quantities of surplus network resources, 2, 1, 3, 1, and 2, respectively. Network link BC has no surplus resource.

We assume two types of tasks: Task X and Task Y. PC_A requests Task X and Task Y at the same ratio. Task X consists of Program1, Program2 and Data X. Task Y consists of Program1, Program2 and Data Y. Task X and Task Y are processed by Program1 and Program2 in order. The states of Task X and Task Y are changed by Program1 from State0 to State1 and by Program2 from State1 to State2.

The sizes of tasks are changed by processing as shown by TABLE I (A). The sizes of Task X on State0, on State1, and on State2 are 2, 1, and 2, respectively. The sizes of Task Y on State0, on State1, and on State2 are 1, 2, and 1, respectively.

In this experimental model, the transference table indicates the next PC that the agent should transfer to, based on the current state of the agent determined by the type of its task and current state of the task as shown in TABLE I (B).

TABLE I

Size of Task and Transference Table of Agent.

(A) Size of Task

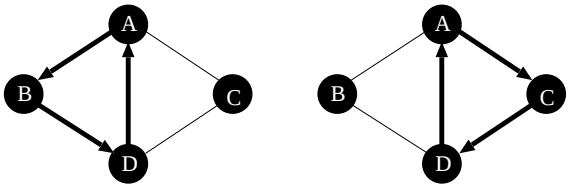
State \ Task	Task X	Task Y
State 0	2	1
State 1	1	2
State 2	2	1

(B) Transference Table of Agent

State \ Task	Task X	Task Y
State 0	TPC(X,0)	TPC(Y,0)
State 1	TPC(X,1)	TPC(Y,1)
State 2	TPC(X,2)	TPC(Y,2)

TPC(*,*)∈ {A,B,C,D}

The self-evaluation value is added when the agent has completed a task that is State2 and the agent returns to the requested PC. In the model, when the quantity of agents' transference on a network link exceeds a limited quantity, the agents chosen at random can transfer less than the limited quantity. An agent with Task X achieves high-efficiency use of the surplus network link resources by transferring along the route of $A \rightarrow B \rightarrow D \rightarrow A$ (Fig. 4(A)), an agent with Task Y has high-efficiency use of network resources by transferring along the route of $A \rightarrow C \rightarrow D \rightarrow A$ (Fig. 4(B)). However, each agent does not sense network resources or computer resources.



(A) Optimal route of Task X (B) Optimal route of Task Y

Fig. 4. Optimal routes of task X and task Y.

B. Experiments

For the 100 runs performed, the results were as follows. In simple GA or DS-GA, TPC (X, 0) and TPC (Y, 0) converged on "B" or "C", TPC (X, 1) and TPC (Y, 1) converged on "D", and TPC (X, 2) and TPC (Y, 2) converged on "A." Consequently, most of the agents pass along the course of $A \rightarrow B \rightarrow D \rightarrow A$ or $A \rightarrow C \rightarrow D \rightarrow A$.

As experimental results, the histories of the population ratio of each transference table by simple GA and by DS-GA are shown in Fig. 5 and Fig. 6, respectively. The history of the average performance of all agents is shown in Fig. 7.

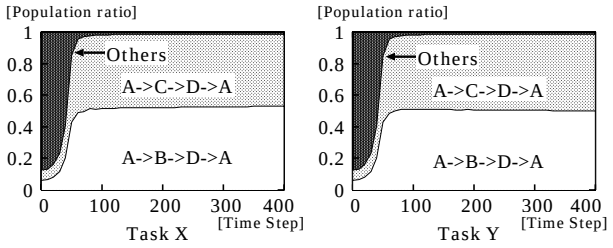


Fig. 5. History of the population ratio of each transference table by simple GA. Each area between the lines corresponds to the ratio of the corresponding parameters of the transference table.

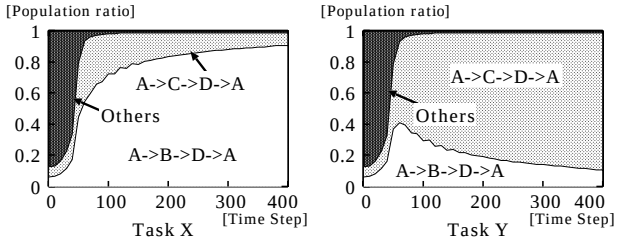


Fig. 6. History of the population ratio of each transference table by DS-GA.

In the experimental model, the agent completes processing in the shortest time with the learned route $A \rightarrow B \rightarrow D \rightarrow A$ or $A \rightarrow C \rightarrow D \rightarrow A$. Furthermore, an agent can use the network resources efficiently when it uses PC_B for Task X and PC_C for Task Y.

In the experimental results of simple GA and DS-GA, agents learned the route $A \rightarrow B \rightarrow D \rightarrow A$ or $A \rightarrow C \rightarrow D \rightarrow A$ before 50 time steps. After that, only in DS-GA did the agents learn the optimal transference tables. As a result, the agents in DS-GA achieve higher collective performance.

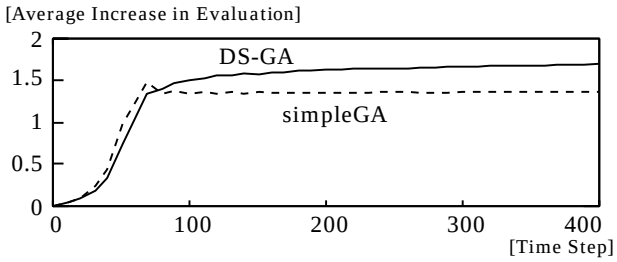


Fig. 7. History of average performance. The solid line corresponds to the average performance of DS-GA. The broken line corresponds to the average performance of simple GA.

C. Discussion

In this experiment, an agent can complete the task only with the route $A \rightarrow B \rightarrow D \rightarrow A$ or $A \rightarrow C \rightarrow D \rightarrow A$. An agent cannot perceive which route is better, since the agents acquire the same rewards for the two routes ABDA and ACDA. Then even if agents have optimal routes for collective performance, there is no increase in the agents that learn based only on agent-sensing.

In DS-GA, if the agents in a channel have higher collective performance than agents in the other channels, the agents in the channel increase. In other words, swarm-sensing emerges in DS-GA. Consequently, agents that have optimal routes for collective performance increase by DS-GA learning.

IV. Application of Swarm-sensing

In the previous section, we investigated "Swarm-sensing" on a simple network model. Here, we show experimentally that the agents learn the optimal routes that we calculated beforehand without perceiving the global information on a more complicated network model.

A. Experimental model

In order to verify "Swarm-sensing" in MAO-DC, we experiment on the more complicated network model shown in Fig. 8.

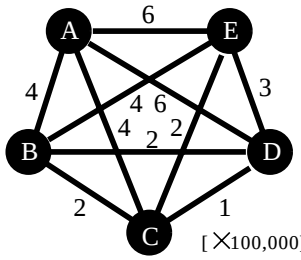


Fig. 8. Conceptual representation of Network Structure and Surplus Network Resources. The parameters in a figure show quantity of network resources.

The network model consists of five PCs, $PC_{\{A,B,C,D,E\}}$, and ten network links. The Five PCs have different characteristics of surplus computer resources, where PC_A can request

a task. PC_B, PC_C, PC_D, and PC_E can execute Program B, C, D, and E, respectively, to process a task as shown in TABLE II(A). Ten network links have different quantities of surplus network resources as shown in Fig. 8.

We assume one type of task: Task Z. Task Z consists of four programs, Program B, C, D, E, and Data Z. Task Z is processed by Program B, C, D and E without an order in this model,

TABLE II
EXECUTABLE PC AND TASK SIZE MAGNIFICATION BY EACH PROGRAM.

(A) Executable PC

Program	Executable PC
Program B	PC B
Program C	PC C
Program D	PC D
Program E	PC E

(B) Magnification of Task Size

Program	Magnification [times]
Program B	×0.5
Program C	×0.5
Program D	× 3
Program E	× 2

The size of Task Z is changed by each processing Data Z as shown by TABLE II (B). The size of the initial Task Z is 4. The size of an agent is disregarded compared with the size of the task. Processing is completed if the task is processed by each of Program B, C, D and E in the four respective PCs once and returned to PC_A.

In this experimental model, the transference table indicates the next PC that the agent should transfer to, based only on the current PC, since the state is corresponded to the PC on which the agent exists currently in the experiment of this section, as shown in TABLE III(A). A self-evaluation value is added when the agent completes the task. When the quantity of agents' transference on a network link exceeds the limited quantity, the agents chosen at random can transfer less than the limited quantity.

The network structure and network resources in the experimental model of this section are shown in Fig. 8 in which the optimal routes are determined as one combination. The optimal combination of the routes is acquired when half of the agents acquire transference table 1 and the other half acquire transference table 2 in TABLE III(B)(C) (Refer to appendix).

TABLE III
TRANSFERENCE TABLES OF AGENTS; DEFINITION AND THE OPTIMAL BY MATHEMATICALLY CALCULATION.

(A)Transferance Table

current PC	Next PC
PC A	TPC(Z,A)
PC B	TPC(Z,B)
PC C	TPC(Z,C)
PC D	TPC(Z,D)
PC E	TPC(Z,E)

TPC(*,*) ∈ {A,B,C,D,E}

(B)Optimal Table 1

current PC	TPC
PC A	B
PC B	C
PC C	D
PC D	E
PC E	A

(C)Optimal Table 2

current PC	TPC
PC A	C
PC B	D
PC C	E
PC D	A
PC E	B

B. Experiments and Discussion

For the 100 runs performed, the results were as follows.

As experimental results of DS-GA, the history of the population ratio of agents with transference table 1 and tables 2 is shown in Fig. 9.

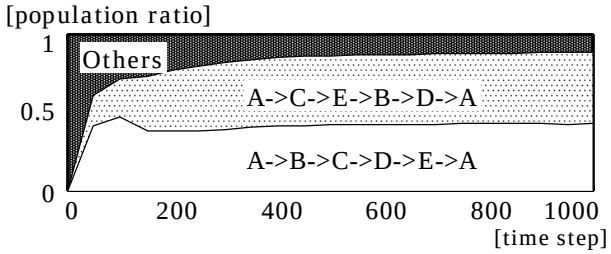


Fig. 9. History of the population ratio of agents with transference table 1 and table 2.

The route of an agent at this time is shown in Fig. 10. Two types of routes, called a pentagon and a star, have appeared.

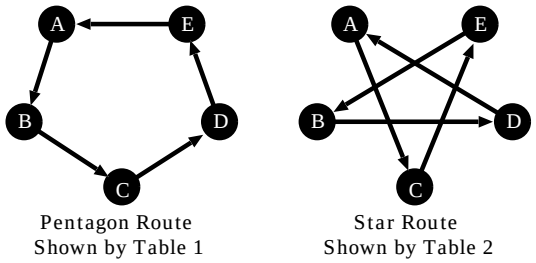


Fig. 10. Optimal routes of the agents.

The agent learned the optimal route for collective performance. This result shows that the characteristics of DS-GA shown in Section III also appears in a more complicated environment.

C. Adaptation in dynamic environment

In a real network environment, network resources may change dynamically. In the experiment of this subsection, we change network resources in 1000 time steps. The network after change is shown in Fig. 11.

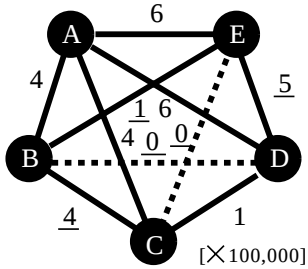


Fig. 11. Conceptual representation of Changed Network Structure and Surplus Network Resources. The underlined parts are the parameters that changed.

The optimal combination of the routes is the case where half of the agents acquire transference table 1 and the other half acquire transference table 3 in TABLE IV (Refer to appendix).

TABLE IV
OPTIMAL TRANSFERENCE TABLES OF AGENTS BY MATHEMATICALLY CALCULATION AFTER CHANGE OF NETWORK.

Optimal Table 1

current PC	TPC
PC A	B
PC B	C
PC C	D
PC D	E
PC E	A

Optimal Table 3

current PC	TPC
PC A	C
PC B	E
PC C	B
PC D	A
PC E	D

The history of the population ratio of agents with transference table 1 and table 3 is shown in Fig. 12.

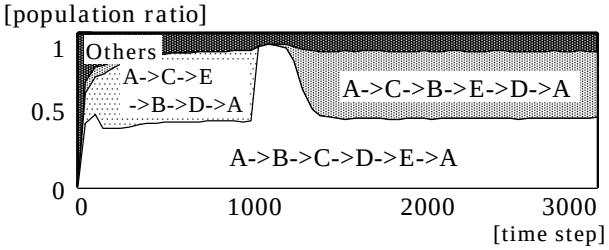


Fig. 12. History of the population ratio of agents with transference table 1 and table 3.

Even in such a dynamic environment, the agent learned the optimal route for collective performance by DS-GA.

V. CONCLUSION

In this paper, we proposed Multi-Agent Oriented Distributed Computing (MAO-DC) and applied DS-GA to MAO-DC to efficiently use the surplus resources of personal computers (PCs). The task-processing agent in MAO-DC learned two types of optimal routes, a pentagon route and a star route, by DS-GA according to the state of the computer resources and network resources without perceiving the information of network resources. Then load balancing of MAO-DC was optimized.

From these results, we found that MAO-DC with DS-GA is efficient for distributed computing under an unknown and dynamic network environment.

APPENDIX

Here, we analyze mathematically the optimal solution of the experimental model in Section IV. In the model, each agent selects the optimal transference table out of all $5^5 (= 3125)$ types that indicates the next PC to which the agent should transfer, based on the current PC. However, the agents should transfer according to one of the ${}_4P_4 (= 24)$ routes, since from the fixed request PC, PC_A, an agent should travel to the four other PCs, $PC_{B,C,D,E}$, at once and return to PC_A. We take into consideration the 24 routes and define $ROUTES$ as the set of the 24 routes (ABCDEA, ABCEDA, $\cdots$, AEDCBA). We define $Agents_R$ as the agents transferring through route R and define N_R as the number of $Agents_R$.

The optimal solution is the set of values $N_R (R \in ROUTES)$, where the value of $\sum_{R \in ROUTES} N_R$ is maximum.

As described below, we consider the restrict conditions of solving the maximum. At first, we obtain the 24 restriction inequations as follows.

$$N_R \geq 0 (\forall R \in ROUTES), \quad (1)$$

since $N_R (R \in ROUTES)$ is positive or zero.

The agents can transfer through ten network links, since the combination is ten when two arbitrary computers are chosen out of the five computers. We define $LINKS$ as the set of 10 network links (AB, AC, $\cdots$, DE) and define $Limit_L$ as the limited quantity of the network link L.

Here, as an example, we consider the influence of $Limit_{AB}$. Twelve types of agents transfer through the network link AB: $Agents_{ABxyzA}$ and $Agents_{AxyzBA}$, where xyz are the permutations C, D, and E. The size of $Agents_{ABxyzA}$ on the network link AB is 4. The size of $Agents_{AxyzBA}$ on AB is 6 $(4(\times 0.5 \times 3 \times 2) \times 0.5)$, according to TABLE II(B). We obtain the restriction inequality as follows.

$$4 \times N_{ABCDEA} + \cdots + 6 \times N_{ACDEBA} + \cdots \leq Limit_{AB} \quad (2)$$

As in the above case, we get the other 9 restrictions by each network link. As a result, we can calculate $\max \sum_{R \in ROUTES} N_R$ under the above 34 restrictions. We get the following optimal solutions, $N_{ABCDEA} = 500000$, $N_{ACEBDA} = 500000$, and $N_R = 0 (R \in$the others$)$ for Subsection IV-A, and $N_{ABCDEA} = 500000$, $N_{ACBEDA} = 500000$, and $N_R = 0 (R \in$the others$)$ for Subsection IV-A.

ACKNOWLEDGMENT

This research was conducted as part of 'Research on Human Communication' with funding from the National Institute of Information and Communications Technology.

REFERENCES

[1] http://www.globus.org
[2] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, Condor-G: a computation management agent for multi-institutional grids, Proceedings 10th IEEE International Symposium on High Performance Distributed Computing (HPDC10), pp. 55-63, 2001.
[3] http://setiathome.ssl.berkeley.edu/
[4] https://hotpage.npaci.edu/
[5] F. Manola and C. Thompson, Characterizing the agent grid, http://www.objs.com/agility/tech-reports/990623-characterizing-the-agent-grid.html, 1999.
[6] M. Fukuda, Y. Tanaka, N. Suzuki, L.F. Bic, and S. Kobayashi, A Mobile-Agent-Based PC Grid, Autonomic Computing Workshop, pp. 142-150, 2003.
[7] K. Nakayama, K. Shimohara, and O. Katai, Dynamically Separating GA: A New Method of Achieving the System-level Optimality in MAS, The journal of the three dimensional images, Vol. 16, No. 4, pp. 177-183, 2002.
[8] E. Bonabeau, M.Dorigo, and G.Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, 1999.
[9] K. Nakayama, K. Shimohara, and O. Katai, Evolutionally computer network generation by DS-LA(in Japanese), Proceedings of the 48th Annual Conference of the Institute of Systems, Control and Information Engineers , pp. 617-618, 2004.
[10] K. Nakayama, H. Matsui, K. Shimohara, and O. Katai, Proposal of Agent Oriented Distributed Computing System and Its Optimization using DS-GA (in Japanese), IPSJ Symposium Series, Vol. 2004, in Printing, 2004.