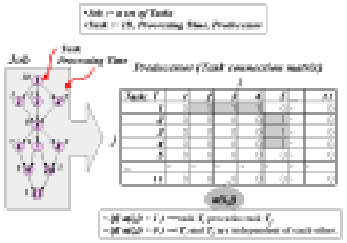
problems through cooperative behaviors based on multi-agent system. In the system, each of the immune agents acts to assign a part of own tasks through interactions and it is to produce an efficient scheduling solution by load sharing. This algorithm is applied to `Standard Task Graph Set' problems to investigate the validity, and the system behaviors are examined.

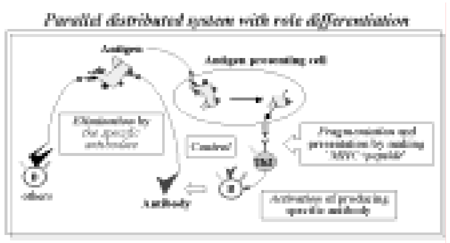
1 INTRODUCTION

Adaptive problem solving techniques, such as neural networks and genetic algorithms, are based on information processing in biological organisms and they are applied on many kinds of optimization problems. A biological immune system is one of the adaptive systems and the studies are making advances [1,3,4,5]. The biological immune system is widely recognized as one of the adaptive biological system whose functions are to identify and to eliminate foreign materials. Especially, it has important notions that carry out highly parallel-distributed functions, for a design of multi-agent system.

In this paper, we propose an immune optimization, which is inspired by immune cell-cooperation and immune tolerance, for meta-scheduling problems. Meta-scheduling can be loosely defined as the act of locating and allocating resources for a job on a parallel-distributed computing [7]. A meta-scheduling system should make a collection of resources transparently available to the user as if it were a single large system. The cell-cooperation is considered as a kind of parallel-distributed system with role differentiations in biological immune system, and the function offers beneficial notions for solving the problems. By analogy, we constructed a system to solve the problems through cooperative behaviors based on multi-agent system. In the system, each of the immune agents acts to assign a part of own tasks through interactions and it is to perform an efficient scheduling solution by a load sharing. The proposed algorithm solves meta-scheduling problems through interactions between agents, and between agents and environment by immune functions. There are two functions in our algorithm: *division-and-integration processing* and *immune tolerance*. The division-and-integration processing resolves precedence constraints, and the immune tolerance performs a load sharing. Through implementation of such functions, we could construct an adaptive algorithm that solves meta-scheduling administers the resources in order to use them. The scheduling issues that surround the creation of such a system is the focus of this section [10].

In this paper, we target at STG as a simplified meta-computer and job. STG is a kind of benchmark for evaluation of multiprocessor scheduling algorithms [11]. To efficiently execute programs in parallel on a multiprocessor environment, a minimum execution time must be solved to determine the assignment of tasks to the processors and the execution order of the tasks so that the execution time is minimized [2]. The multiprocessor scheduling problem treated in their project is to determine a non-preemptive schedule that minimizes the execution time, or the schedule length, when a set of *n* computational tasks having *arbitrary* precedence constraints and *arbitrary* processing time are assigned to *m* processors of the same capability. These tasks are represented by a *directed acyclic graph (DAG)* called a ``task graph", as shown in figure 1.





Parallel distributed system with role differentiation
Antigen
Antigen presenting cell
Elimination by the specific antibodies
Control
Fragmentation and presentation by making "MHC-peptide"
Th2
Antibody
others
Activation of producing specific antibody

Figure1. Illustration of job for multi-processor scheduling problem in STG.

Figure2. Concept of immune cell-cooperation.

3 DESIGN OF IMMUNE META-SCHEDULING SYSTEM

3.1 Analogy from biological immune system

We solve the meta-scheduling problems by means of agent-based computing in the field of multi-agent system. The agents that introduced immune functions decide a scheduling plan (processing order of tasks) through communications between the agents. A purpose of this paper is obtaining a set

In order to apply the system to the meta-scheduling problems, we choose three functions, fragmentation, activation and elimination, in this issue, because the functions are minimum components to perform as problem-solvers. We construct an optimization algorithm based on a concept that (1) fragments a problem, (2) solves the fragmented sub-problems by the specific sub-solutions, and then (3) solves whole the problem through combination of these sub-solutions. In other words, we use these functions (called *division-and-integration processing*) to divide a job into tasks and assign the tasks to resolve precedence constraints. In addition, we introduce an immune tolerance which is a phenomenon that the immune system dosen't respond with one and/or some specific antigens. By regarding the selection function of opponents as a decision function, the tolerance function as a control mechanism is using for agent's behavior arbitration in our algorithm.

3.3 Algorithm

The algorithm solves the problems through two searching ways, *division-and-integration processing* and *immune tolerance*. The procedures of the algorithm against a STG problem are described as below. In the procedures of proposed method, Step1 is processed for initialization once for all. Each of agents processes repeatedly from Step2 to Step4 for solving the problem.

[Step1. Definition of problems.] A *Job* and *multiprocessor system* as the problem must be defined. A Job that consists of a set of tasks, is represented as *directed acyclic graph (DAG)*, and needs the number of tasks, processing time and a task connection matrix (describes precedence constraints), see figure 1. As definition of the multiprocessor system, it is described only the number of processors according to STG. And then, all the agents are initialized for begining following procedures. An agent exists in individual processor, and determines tasks which should be processed by the processor through communications between agents. In the initialization process, all the tasks are assigned into each agent's queue at random.

[Step2. Calculation of objective function.] Each of agents calculates $Time_{work}$ and $Time_{free}$ as objective functions through performing as a current schedule.

- $Time_{work}$ is defined as a processing time of assigned tasks.
- $Time_{free}$ is defined as a free time, that is no processing time.

In other words, a scheduling time for all the tasks equals to $Time_{work} + Time_{free}$.

preferentially. In case of a execution of the task is impossible, go to next procedure, too.

3. **Process a task in which the execution in other agent's queue is possible.**

   A free agent tries to search the task by checking feasible tasks from other agent's tasks. The agent finds out all the feasible tasks by communication against other agent in the beginning. Then, the agent goes on to decide a processing task as well as the previous heuristics, and finally, the processing is started after the agent gets the target task from opponent agent. In case of a execution of the task doesn't exist, the agent doesn't work in this time step.

Each of agents continues processing of above procedures until agents finish processing of all the tasks, that is, all the agent have a feasible scheduling plan. In the manners, it is expected that each of agents produces feasible scheduling plans and efficient sharing plans through two ways of resolving constraints and load sharing.

4 EXPERIMENT

4.1 Definition of problem

To confirm the basic performance, we apply to a benchmark problem which is described in the table 1 which is defined as `proto151.stg' on STG.

4.2 Results and prospects

Figure 3 shows obtained scheduling plans in the first time step 1 and 2. The scheduling time for processing of all the tasks in the 1st time step is *119* that is optimal schedule length according to STG. In the 1st time step, the precedence constraints are resolved to obtain feasible plans by using *division-and-integration processing* (Step4) because the initial plans of agents are made at random. After the 1st time step, the transfers of a task from an overworked agent to a workless agent are executed to achieve a load sharing by using *immune tolerance* (Step3), and then Step4 is executed also. In the 2nd time step, the maximum load is improving from *0.899160* to *0.789916*, and differs of the plans are showed in fig.4 with gray number. The plans in the 1st time step exist a bias that the agents with small ID process many tasks, and the bias is improving so as to be sharing evenly.

the real applications. For example in this results (figure 4), it is possible to use the resources properly according to the situation of LAN, processors or tasks.

A transition of $\underline{Time_{task}}$ of agents is shown in figure 5. In the 1st time step, the time of most workless agent(no.*10*) is only *30* for a set of tasks `*15, 40, 74*', on the one hand, the time of most overwork agent(no.*2*) is *107* for a set of tasks `*6,19,18,34,42,41,57,53,67,73,78,81*'. That is, the difference of the $Time_{task}$ is *77*.

In the opposite direction, the time step has the smallest difference is 15th step, the most workless's time is *67*, the most overwork's time is *90* and the differ is *23*. Such a solution will be a useful planning in case that the user requires to minimize a recovery cost from any fault of processors.

Table 1. Characteristics of proto151.stg.

the number of processors	10
the number of tasks	80
Predecessors (max,ave,min,sum)	10, 3.74, 0, 303
Task processing time (max,ave,min,sum)	12, 9.57, 0, 775

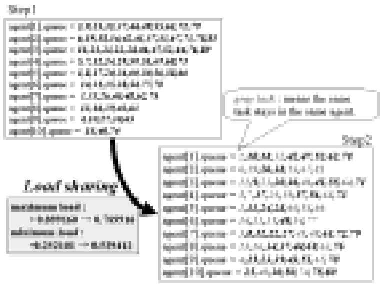


Figure3. An example of load sharing from the step1 to step2.

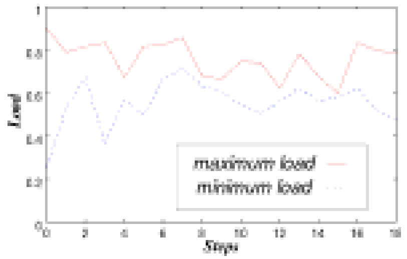


Figure4. Transition of max- and min-load.

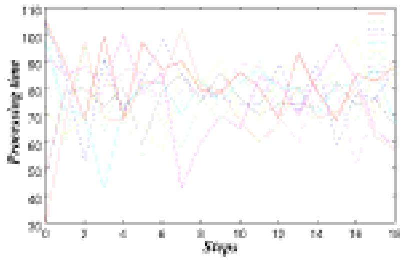


Figure5. Transition of $Time_{task}$.

optimizing of the whole problem by using the local interactions. Since our algorithm can optimize division-of-labor problems, it can expect what is functioned effectively as an optimization algorithm in multi-agent system.

In addition, the system can obtain multiple feasible optimal solutions, namely, we think that it is possible to find some specified characteristics on the problem by means of an identifying common denominators in the solutions. For designing the identifying function, a concept of an immunological memory will be informative and instructive mechanism. Such a scheduling system that learns proper meta-knowledges as a key point of characteristics and produces well-suited plans against the problems, will be a very useful scheduler. As future works, we model and construct such a system to learn proper meta-knowledges.

The first author acknowledges the Grant-in-Aid for Scientific Research (Grant-in-Aid for JSPS Fellows).

REFERENCES

H. Bersini and F. J. Varela, 1991, “The Immune Recruitment Mechanism : A Selective Evolutionary Strategy,” Proc. of ICGA 91.

E. G. Coffman, 1976, “Computer and Job-shop Scheduling Theory,”. John Willey & Sons.

J.D. Farmer, N.H. Packard, A.A. Perelson, 1986, “The Immune system adaptation, and machine learning,”Physica 22D, pp.187-204.

Forrest and A. S. Perelson, 1990, “Genetic Algorithm and the Immune System,” Proc. of PPSN 90, pp.320-325.

Y. Ishida, and N. Adachi, 1996, “Active Noise Control by an Immune Algorithm,” Proc. ICEC 96, pp.150-153.

Charles A. Janeway, Jr., Paul Travers ; with assistance of Simon Hunt, Mark Walport, 1997, “Immunobiology : The Immune System in Health And Disease,”Garland Pub.

Larry Smarr and Charles E. Catlett, 1992, “Metacomputing,”Communications of the ACM, 35:45-52.

N. Toma, S. Endo, K. Yamada, H. Miyagi, 2000, “The Immune Distributed Competitive Problem Solver Using MHC and Immune Network,” Proc. of The 2nd Joint International Workshop -ORSJ Hokkaido Chapter and ASOR Queensland Branch -, pp82-89.

Quinn Snell, Mark Clement, David Jackson, and Chad Gregory , 2000, “The Performance Impact of Advance Reservation Meta-Scheduling,”6th WORKSHOP ON JOB SCHEDULING STRATEGIES FOR PARALLEL PROCESSING.

Kasahara Lab., Waseda Univ , http://www.kasahara.elec.waseda.ac.jp/schedule/