A Bio-inspired Adaptive Job Scheduling Mechanism on a Computational Grid

Yaohang Li†

†*Department of Computer Science, North Carolina A&T State University, Greensboro, NC 27411, USA*

Summary
A computational grid is a highly dynamic and distributed environment. Unlike tightly-coupled parallel computing environment, high performance computing on the grid is complicated by the heterogeneous computational performances of each node, possible node unavailability, unpredictable node behavior, and unreliable network connectivity. Compared to a static scheduling, an adaptive scheduling mechanism is more favorable and attractive in a grid-computing environment, because it can adjust the scheduling policy according to its dynamically changing computational environment.
In this paper, we present a job scheduling mechanism that enable the adaptation of naturally parallel and compute-intensive jobs to clustered computational farms with heterogeneous performance. The kernel of this scheduling technique is a swarm intelligent algorithm, which is inspired from the ants' behavior in a social insect colony. We applied the bio-inspired adaptive mechanism in a simulated computational grid and compared it with static scheduling algorithms. Our results showed good performance, adaptability, and robustness in a dynamic computational grid with respect to its competitors.

Key words:
Grid Computing, Swarm Intelligence

1. Introduction

Grid computing is characterized by large-scale sharing and cooperation of dynamically distributed resources, such as CPU cycles, communication bandwidth, and data, to constitute a computational environment [1]. A large-scale computational grid [19] can, in principle, offer a tremendous amount of low-cost computational power. This attracts many computationally intensive scientific applications. On the other hand, significant challenges also arise. The computational grid exhibits dynamic and unpredictable behaviors – the computational performances of each node vary greatly from time to time; the network connections may become unreliable; nodes may join or leave the grid system at any time; nodes may become unavailable without any notifications. As a result, a computational job running on different nodes on the grid will lead to a huge range of completion times. In some extreme cases, a job may never be able to complete. Therefore, how to effectively schedule the grid resources to minimize the job execution time is an issue of prime importance.
Social insects, such as bacteria [2], ants [3], and caterpillars [4], exhibit a collective problem solving capability, which shows strong adaptability and robustness to dynamically changing environment. This property is referred as the swarm intelligence [20]. In a swarm intelligence system, agents are specialized in particular unsophisticated functionalities and interact with their environment to exhibit globally collective intelligence. Particularly, the foraging behavior and the collaboration of specialized type of ants in an ant colony inspire us to investigate in the ants' behavior and adopt this mechanism in adaptive job scheduling on the computational grid.
In this paper, we consider the problem of scheduling a set of natural parallel jobs with different arrival times to run on a computational grid. We present a novel job scheduling mechanism inspired by the behavior of the ant colony to effectively utilize the dynamic distributed resources in the grid-computing environment to achieve an optimal job completion time. Similar to the collective behavior of social insects, this scheduling mechanism exhibits strong adaptability and robustness to the dynamic nature of the grid-computing environment.
The remainder of this paper is organized as follows. In Section 2, we analyze the nature of the ant colony's social behavior. We discuss the behavior of grid resources and introduce the bio-inspired job scheduling mechanism using swarm intelligence in Section 3 and Section 4, respectively. We compare our simulation results of the bio-inspired scheduling mechanism in a simulated computational grid with other scheduling mechanisms in Section 5. Finally, Section 6 summarizes our conclusions and future research directions.

2. The Ant Colony

Workers of some social insects specialize in particular tasks and perform them during greater part of their lives than other workers do. For example, the soldiers specialize in killing enemies, the scouts aim at searching for food sources, the carriers focus on collecting water and food,

Manuscript revised March 2006.

the servants are responsible of keeping the hive clean and warm, and the queen's task is producing new ants [7]. The specialization of social insects associated with morphological adaptations is called "caste polyethism" [8]. In a social organization, each social insect worker carries out relatively simple functionality; however, the collective behaviors of these unsophisticated workers with various specialties cause coherent functional intelligent global patterns to emerge. The swarm intelligence exhibits strong adaptability and robustness in dynamically changing environment.

The behaviors of social insects have captured the attention of many scientists because of their problem solving capability with relative simplicity of the colony's individuals. An important and interesting behavior of ant colonies is their foraging behavior, and, in particular, how ants can find the shortest paths between food sources and their nest. Ant algorithms were inspired by the observation of real ant colonies. While walking from food sources to the nest and vice versa, ants deposit a type of chemical named pheromone on the ground, forming in this way a pheromone trail. The pheromone trail allows the ants to find their way back to the food source (or to the nest). More importantly, other ants can use the pheromone trail found by their nest mates to find the location of the food sources. Moreover, pheromones evaporate, meaning that an obsolete trail will gradually disappear.

In an ant colony, the ants can be modeled as probabilistic processes. In the absence of pheromone, the ants explore the surrounding area in a totally random manner. If pheromone exists, the ants can smell pheromone and follow the pheromone trail with a high probability. If two pheromone trails cross each other, the ants tend to choose, in a higher probability, paths marked by stronger pheromone concentrations to follow. At the same time, the ants reinforce the trail by depositing their own pheromones. Where the more are the ants following a trail, the more that trail becomes attractive for other ants to follow. The quantity of pheromone in a shorter path grows faster than that on the longer one, and therefore the probability with which any single ant chooses the path to follow is quickly biased towards the shorter one. Finally, most of the ants will choose the shorter path. However, the decision of whether to follow a path or not is never deterministic, thus always allowing new routes to be explored. Eventually, the shortest path to the food source will emerge [11].

The phenomenon of foraging in an ant colony shows that minimal level of individual complexity can explain sophisticated collective behaviors. Satisfactory computational models have been developed to simulate the food searching process of an ant swarm. Algorithms that take inspiration from ants' behavior in finding shortest paths have recently been successfully applied to combinatorial optimization [5], circuit switched communications network problem [9], and adaptive routing problem [6].

3. The Resources in the Grid

Grid computing, which can be characterized as large-scale distributed resource sharing and cooperation, has quickly become a mainstream technology in distributed computing. In a computational grid, large-scale computational resources, global-wide networking connectivity, access to high-end scientific instruments, participation of scientists and experts in different areas, and coordination of organizations make the grid a powerful and cost-effective platform to carry out large-scale scientific computing operations. Nevertheless, despite the attractive characteristics of grid computing, to successfully apply the grid technique in scientific computation, the grid environment presents a number of significant challenges:

Heterogeneity: The grid resources within a computational grid exhibit heterogeneous computational performances. The capabilities of each node vary greatly. A node might be a high-end supercomputer, or a low-end personal computer, even just an intelligent widget. Also, different grid nodes may employ different job-running policies. As a result, a task running on different nodes on the grid will have a huge range of completion times. Moreover, the grid job scheduler may not be able to obtain any indication of the performance of a grid node.

Dynamism: Grid computing is a highly dynamic computational environment – nodes may join or leave the grid system at any time according to their owner's discretion; the network connecting the grid nodes may become unavailable; the performance of grid resources may change frequently over time; the heavy workload may also turn a "fast" node into a "slow" node [17].

Trustworthiness: In a grid-computing environment, the service providers of the grid are often geographically separated with no central management. Faults may hurt the integrity of a computation. These might include faults arising from the network, system software or node hardware. A node providing CPU cycles might not be trustworthy. A user might provide a system to the grid without the intent of faithfully executing the applications obtained. Experience with SETI@home [12] has shown that users sometimes fake computations and return wrong or inaccurate results. The resources in a grid system are so widely distributed that it appears difficult for a grid-computing system to completely prevent all "bad" nodes from participating in a grid computation.

As a result, to efficiently and effectively utilize the grid resources, the grid-computing environment requires a fundamentally new computing paradigm that will differ in both substance and scale from those of the traditional parallel or distributed computing.

4. Bio-inspired Job Scheduling Mechanism using Swarm Intelligence

The goal of every job-scheduling algorithm on the grid is to minimize the execution time of the computational jobs by effectively taking advantage of the large amount of distributed resources. In our bio-inspired job scheduling mechanism using swarm intelligence, we design various software ant agents with simple functionalities. No direct communications occur among these agents. The only indirect communication is via the pheromone values stored in a global grid resource table. We expect the collective behaviors of these simple agents suit the dynamic nature of the grid.

4.1 Grid Resource Table

The only global data structure used in the bio-inspired job schedule algorithm using swarm intelligence is a grid resource table. The grid resource table keeps track of the available grid nodes providing computational services and the pheromone value associated with them. The pheromone value decreases as time goes on to simulate the "evaporating" process. An ant agent can also deposit pheromone by increasing the pheromone value in the grid resource table.

4.2 Specialized Ant Agents

Similar to the caste polyethism in social insects, we simulate several agents with distinct simple functionalities in the bio-inspired scheduling mechanism on the grid. These specialized agents are categorized as follows:

- Scout: Grid computing exhibits high dynamism – the grid resources may become available or unavailable without any notice. The responsibility of the scout is to discover the new grid nodes providing appropriate computational services. Once such a new grid node is found, the scout adds it to the available resource table with an initial pheromone value.
- Tester: A tester executes a small sample program on a grid node and test for the computational time of the sample program. The tester updates the pheromone value of this particular grid node according to the job completion time of the sample program.
- Worker: A worker chooses an available grid node and carries out a computational job in the system on this node. The grid nodes with higher pheromone value will have a higher probability to be selected.
- Cleaner: A cleaner maintains the available grid resource table in the system. It removes the unavailable resources (with low pheromone value) from the grid resource table.
- Queen: The queen's task is to produce the specialized agents, including the scouts, testers, cleaners, and workers.

All these agents fulfill their own simple functionalities. There is no direct communications among all these agents. Fig. 1 shows the behaviors and simple functions of these specialized agents.

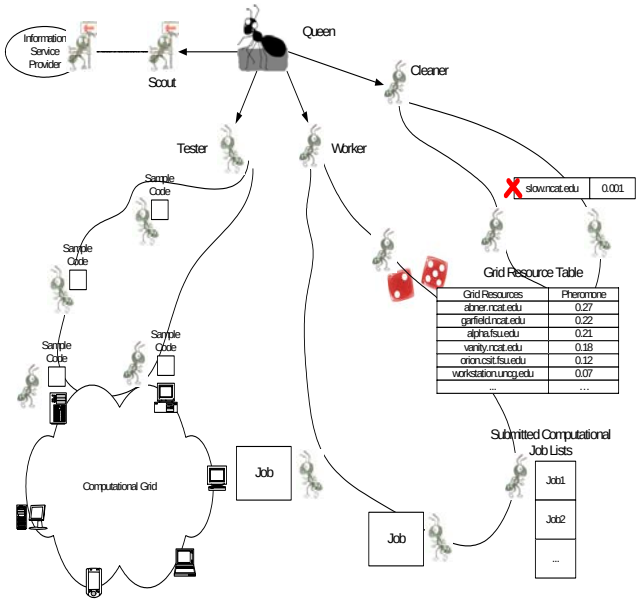

Fig. 1 Behaviors of Ant Agents

4.3 Bio-inspired Job Scheduling Mechanism using Swarm Intelligence Algorithm

Let us put all the pieces of the swarm intelligent algorithm together. The bio-inspired mechanism of job scheduling on the computational grid is depicted as follows:

1. Initially, the queen spawns scouts, cleaners, and workers. The queen also produces testers at a time period of T.
2. A scout visits the information services providers of the grid and explores those nodes providing computational services. The scout finds the available nodes and adds them to the grid resource table with initial pheromone value, θ.
3. Once a job is submitted to the computational grid, a worker will try to schedule this job to an available node. A node having a higher pheromone value will be selected with a higher probability. A node i will be selected with probability, q_i, of

$$q_i = p_i / \sum_{j=1}^{n} p_j ,$$

where p_i is the pheromone value of node i and n is the total number of available nodes in the system.
4. Testers are produced periodically. Each tester carries out a small sample program on every node in the grid resource table. When the sample program is complete,

the pheromone value p_i of node i is updated by $p_i + 1/T_i$, where T_i is the execution time of the sample program on node i. If a sample program cannot be complete by a node, the pheromone value associated with this node will not change. The period θ of scheduling the testers is a tunable parameter.

5. The pheromone values of nodes evaporate. In our implementation, we implement the evaporation process by normalizing the pheromone values in the grid resource table periodically. The pheromone value becomes

$$p_i / \sum_{j=1}^{n} p_j .$$

After normalization, all the pheromone values of the nodes in grid resource table obey

$$\sum_{j=1}^{n} p_j = 1 .$$

6. When the pheromone value of a node is lower than some threshold value, τ, which usually means that this node has been unavailable for a long time or this node is an extremely slow node with an undesired job completion time, the cleaner will remove it from the grid resource table.

In the bio-inspired scheduling algorithm, variables T, θ, and τ, are tunable parameters subject to the grid system.

4.4 Analysis of the Bio-inspired Job Scheduling Mechanism

1. Adaptability and Robustness

A computational grid is a highly dynamic and distributed computing environment. To be adaptive to a dynamically changing computational grid, the key of the bio-inspired job scheduling mechanism is to keep track of the pheromone value table with the pheromone values reflecting the most update performance of each node in the grid-computing environment. The trade-off is the overhead of scheduling and running the tester program on the grid nodes.

Also, due to the wide distribution and uncontrollability of grid resources, a grid node may be temporally unreachable [17]. To handle this situation, the bio-inspired scheduling mechanism will not remove a previously well-performing node from the grid resource table immediately even though it becomes temporally unavailable. Only when the grid node has left the grid system for a long time and its associated pheromone value has evaporated to be lower than the threshold value, τ, the grid node will be removed from the grid resource table by the cleaner and its assigned jobs will be rescheduled to other nodes.

2. Trustworthiness

A surprising byproduct of the bio-inspired job scheduling mechanism is that a way to improve the trustworthiness of the computational grid is provided. In the bio-inspired job scheduling, the sample program carried out by the tester can not only test the performance of a grid node, but also verify whether a grid node can faithfully carry out and accurately execute its assigned tasks or not.

To enable the bio-inspired job scheduling mechanism the capability of verifying the trustworthiness of grid resources, we design the tester sample program to produce a series of intermediate values. These intermediate values vary according to the random initial condition. To the grid node that runs the tester sample program, these values will be unknown until the task is actually executed and reaches a specific point within the program. On the other hand, to the clever scheduler, certain intermediate values are either pre-known and secret or are very easy to generate. Therefore, by comparing the intermediate values and the pre-known values, we can control whether the sample tester program is actually faithfully carried out or not. The crude Monte Carlo integration programs can be the ideal candidates as such tester sample programs [13].

5. Simulation Results

In a computational grid, two common scheduling mechanisms are widely used, including the random scheduling method and the heuristic scheduling method. The random scheduling is employed in many volunteer computing systems such as SETI@home [12] and Condor [14], which has no extra information of the performance of a participated grid node and, thus, schedules computational jobs to its grid nodes in a random manner. The heuristic scheduling method [15, 16] takes advantage of the previous overall performance of a grid node in managing workload on different grid nodes. To investigate the performance of the bio-inspired job-scheduling algorithm, we simulate a computational grid with participant nodes of heterogeneous computational performances. The arrival rate of the naturally parallel and computation-intensive jobs in the grid system conforms to a Poisson distribution. Each job requires a constant number of operations. We compare the bio-inspired mechanism with the random scheduling mechanism and the heuristic scheduling mechanism in this simulated grid environment with grid nodes using a time-sharing policy or a queuing policy.

5.1 Grid Nodes using a Time-Sharing Policy

In this experiment, we simulate the behavior of a computational grid whose participant nodes employ a time-sharing policy, i.e., once a job is scheduled to a grid node, it will be executed concurrently with existing jobs in the grid node. Fig. 2 shows the comparison of the average job completion times of the bio-inspired mechanism, the heuristic mechanism, and the random mechanism with various job arrival rates. The data in Fig. 2 come from

simulation on a grid whose participant nodes have fixed performance values. This can be a cluster constructed from dedicated computers. The bio-inspired mechanism and heuristic mechanism significantly outperform the random mechanism. The jobs using bio-inspired scheduling mechanism take slightly more completion time compared to the heuristic mechanism because of the overhead of the execution of the tester sample program.

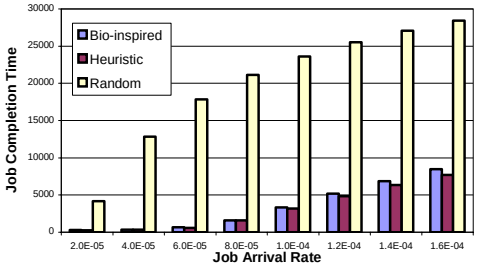

Fig. 2 Comparison of bio-inspired mechanism, heuristic mechanism, and random mechanism in job scheduling on a simulated computational grid. Jobs in a grid node share CPU time. Nodes have fixed performance values.

To introduce dynamism to the simulated grid, we allow the performance of a grid node change with a probability of ρ at each time step. Fig. 3 illustrates the job completion times on a simulated computational grid in which the performances of the participant nodes change from time to time. In practice, this grid can be a volunteer computing system such as Condor [14] or SETI@home [12]. At each time step, the performance of every node within the simulated grid changes with a probability of ρ = 0.0001. The performance value of a grid node may change to 0, which indicates that the node leaves the grid-computing environment. When ρ is high enough, i.e., a highly dynamic grid system, the heuristic values in the heuristic scheduling mechanism can no longer accurately reflect the computational performances of the grid nodes in the dynamically changing computing environment, which leads to a poor performance of the heuristic scheduling mechanism. In contrast, the bio-inspired mechanism shows a better average job completion time than the random mechanism and heuristic mechanism.

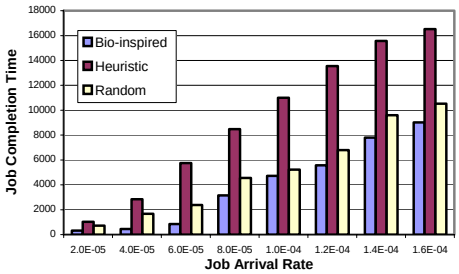

Fig. 3 Comparison of bio-inspired mechanism, heuristic mechanism, and random mechanism in job scheduling on a simulated computational grid. Jobs in a grid node share CPU time. Nodes change performance with probability of 0.0001.

Fig. 4 depicts the adaptability of the bio-inspired scheduling mechanism. As the probability, ρ, of node performance changing increases, the simulated grid evolves from a slightly dynamic system to a heavily dynamic system. The data in Fig. 4 shows that the performances of the heuristic mechanism and the random mechanism change dramatically; in contrast, the bio-inspired mechanism exhibits a rather steady performance.

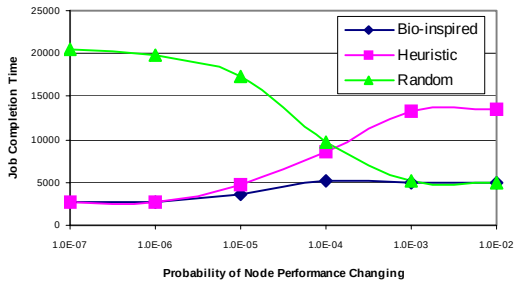

Fig. 4 Performance comparison of bio-inspired mechanism, heuristic mechanism, and random mechanism in job scheduling on a simulated computational grid with different probabilities of node performance changing.

5.2 Grid Nodes using a Queuing Policy

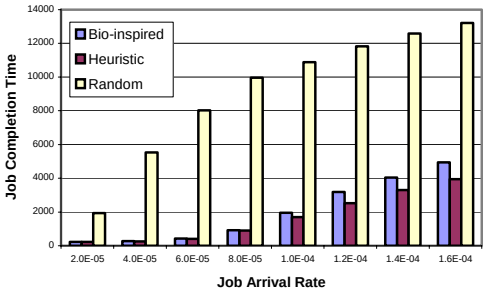

Fig. 5 Comparison of bio-inspired mechanism, heuristic mechanism, and random mechanism in job scheduling on a simulated computational grid. The grid nodes employ a queuing policy. Nodes have fixed performance values.

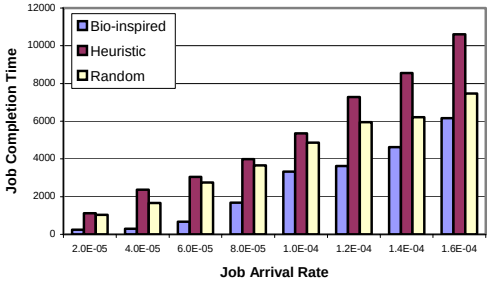

Fig. 6 Comparison of bio-inspired mechanism, heuristic mechanism, and random mechanism in job scheduling on a simulated computational grid. The grid nodes use a queuing policy. Nodes change performance with probability of 0.0001.

We simulate the behavior of a computational grid whose participant nodes employ a queuing policy. Suppose the grid node is an M/M/1 system, i.e., a node in this grid can execute at most one job at a time and process jobs in a FIFO discipline. Fig. 5 and 6 show the average job completion time comparison of the bio-inspired mechanism, the heuristic mechanism, and the random mechanism on a simulated grid with fixed performance nodes and dynamic performance nodes, respectively. The experimental results exhibit similarity to those of the grid with nodes using a time-sharing policy presented in section 5.1. On a grid with fixed performance nodes, compared to the heuristic mechanism, the jobs using bio-inspired scheduling mechanism take slightly more completion time due to the overhead of running the tester sample program. On a dynamic grid, the bio-inspired mechanism outperforms the random mechanism and the heuristic mechanism.

6. Conclusions

The swarm intelligence algorithms generically exhibit natural adaptability and robustness characteristics by collaboration of unsophisticated agents interacting with the environment, which can be employed in a highly dynamic computing environment, such as a computational grid. Inspired from the behavior of an ant colony, in this paper, we presented a bio-inspired job scheduling mechanism that enable the adaptation of naturally parallel and compute-intensive jobs to a computational grid with heterogeneous and dynamic performance. We applied the bio-inspired mechanism in a simulated computational grid and compared it with the random mechanism and heuristic mechanism. Our results indicated both good adaptability and robustness in a dynamic computational grid. Moreover, the bio-inspired mechanism provides an interesting approach to verify the trustworthiness of the distributed computation on the computational grid by designing a tester program that can produce easy-to-verify intermediate values and partial results.

The next phase of our research will be to apply the bio-inspired scheduling mechanism into a real-life computational grid. We are in the development phase of a grid scheduling software package using the Globus Toolkit [18] based on the bio-inspired scheduling mechanism presented in this paper. Our immediate goal is to demonstrate the adaptability and robustness of the bio-inspired scheduling mechanism on various grid testbeds.

References

[1] I. Foster, C. Kesselman, S. Tueske, “The Anatomy of the Grid,” Intl. Journal of Supercomputer App., 15(3), 2001.

[2] J. L. Denebourg, J. M. Pasteels, J. C. Verhaeghe, “Probabilistic Behavior in Ants: a Strategy of Errors?” Journal of Theoretical Biology, 105: 259-271, 1983.

[3] J. A. Shapiro, “Bacteria as Multicellular Organisms,” Scientific American, pp. 82-89, 1988.

[4] T. D. Fitzgerald, S. C. Peterson, “Cooperative Foraging and Communication in Caterpillars,” Bioscience, 38: 20-25, 1998.

[5] M. Dorigo, G. Di Garo, “Ant Algorithms for Discrete Optimization,” Artificial Life, 5:137-172, 1999.

[6] G. Di Garo and M. Dorigo, “An Adaptive Multi-Agent Routing Algorithm Inspired by Ants Behavior,” Proc. of 5th Annual Australasian Conf. Para. & Real-Time Sys., 1998.

[7] C. Detrain, J. M. Pasteels, “Caste Polyethism and Collective Defense in the Ant, Pheidole Pallidula: the Outcome of Quantitative differences in recruitment,” Behav. Ecol. Sociobiol. 29:405-412, 1992.

[8] E. O. Wilson, “The Insect Societies,” Harvard University Press, Cambridge, 1971.

[9] G. Di Caro and M. Dorigo, “Ant Colonies for Adaptive Routing in Packet-Switched Communications Networks,” Proceedings of Fifth International Conference on Parallel Problem Solving from Nature, 1998.

[10] H. M. Botee, E. Bonabeau, “Evolving Ant Colony Optimization,” Advances in Complex Sys., 1(2):149-159, 1999.

[11] A. Colorni, M. Dorigo, V. Maniezzo, “Distributed Optimization by Ant Colonies,” Proceedings of First European Conference on Artificial Life, 1992.

[12] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Lebofsky, “SETI@home-Massively distributed computing for SETI,” Comp. in Sci. and Eng., v3n1, 81, 2001.

[13] Y. Li and M. Mascagni, “Analysis of Large-scale Grid-based Monte Carlo Applications,” Intl. Journal of High Performance Comp. App. (IJHPCA), 17(4): 369-382, 2003.

[14] M. Litzkow, M. Livny, M. Mutka, “Condor - A Hunter of Idle Workstations,” Proc. of 8th Intl. Conf. of Distributed Computing Systems, 1988.

[15] M. Wu, X. Sun, “A General Self-adaptive Task Scheduling System for Non-dedicated Heterogeneous Computing,” Proc. of IEEE Intl. Conf. on Cluster Computing, 2003.

[16] C. R. Reeves, “Modern Heuristic Techniques for Combinatorial Problems,” John Wiley & Sons, McGraw-Hill International Ltd.

[17] A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj, D. Snelling, “Open Issues in Grid Scheduling,” UK e-Science Technical Report, 2004.

[18] Globus toolkit website, http://www.globus.org, 2004.

[19] G. Fox, W. Furmanski, “Java for Parallel Computing and as a General Language for Scientific and Engineering Simulation and Modeling,” Concurrency: Practice and Experience, 9(6):415-425, 1997.

[20] E. Bonabeau, G. Théraulaz, “Swarm Smarts,” Scientific American, pp. 72-79, 2000.

Yaohang Li received his B. S. in Computer Science from South China University of Technology in 1997 and M.S. and Ph.D. degree from Department of Computer Science, Florida State University in 2000 and 2003, respectively. After graduation, he worked as a research associate in the Computer Science and Mathematics Division at Oak Ridge National Laboratory, TN. His research interest is in Grid Computing, Computational Biology, and Monte Carlo Methods. Now he is an assistant professor in Computer Science at North Carolina A&T State University.