A Bio-inspired Job Scheduling Algorithm for Monte Carlo Applications on a Computational Grid

Yaohang Li
Department of Computer Science
North Carolina A&T State University
Greensboro, NC 27411, **USA**
Phone: ++1-336-334-7245, Fax: ++1-336-334-7244, E-mail: yaohang@ncat.edu

Michael Mascagni
Department of Computer Science and School of Computational Science
Florida State University
Tallahassee, FL 32306-4530, **USA**
Phone: ++1-850-644-3290, Fax: ++1-850-644-0098, E-mail: mascagni@fsu.edu

Abstract - In this paper, we present a bio-inspired job scheduling mechanism that enables the adaptation of large-scale, naturally parallel and compute-intensive Monte Carlo tasks to clustered computational farms. Examples of such farms include large-scale computational grids, with heterogeneous and dynamic performance. The kernel of this scheduling mechanism is a swarm intelligent algorithm, which is inspired from ants' behavior in their social insect colony. We apply the bio-inspired adaptive mechanism in a simulated computational grid and compare it with static scheduling algorithms. Our preliminary results show good performance, adaptability, and robustness in a dynamic computational grid with respect to a simple scheduling mechanism and the *N-out-of-M* scheduling mechanism.

Keywords: Monte Carlo Methods, Bio-inspired Methods, Grid Computing

I. INTRODUCTION

Monte Carlo applications are widely perceived as computationally intensive but naturally parallel. They are a natural fit to the grid-computing environment using the dynamic bag-of-task model. On the one hand, a large-scale computational grid [FOS, 01] can, in principle, offer a tremendous amount of low-cost computational power. This attracts many computationally intensive scientific applications to the grid. On the other hand, a computational grid is a highly dynamic and distributed environment. High performance computing on the grid is complicated by the heterogeneous computational capabilities of each node, possible node unavailability, unpredictable node behavior, and unreliable network connectivity. Therefore, an efficient job scheduling algorithm to effectively use grid resources is critical for minimizing the completion time of a large-scale Monte Carlo computation on the computational grid [LI, 03]. Compared to simple static scheduling, an adaptive scheduling mechanism is more favorable and attractive for carrying out Monte Carlo applications in a grid-computing environment, because it can adjust the scheduling policy according to its dynamically changing computational environment.

Social insects, such as bacteria [DEN, 83], ants [SHA, 88], and caterpillars [FIT, 98], exhibit a collective problem solving capability, which shows strong adaptability and robustness to dynamically changing environments. This property is referred to as "swarm intelligence" [BON, 00]. In a swarm intelligence system, agents are specialized in particular but 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 one to investigate the ants' behavior and to attempt to incorporate this mechanism into adaptive job scheduling of naturally parallel Monte Carlo tasks on the computational grid.

In this paper, we consider the problem of partitioning a very large Monte Carlo task into small subtasks and then scheduling the subtasks to run on a dynamic computational grid. We present a novel job scheduling mechanism inspired by the behavior of an ant colony to effectively utilize the dynamic distributed resources in the grid-computing environment to achieve an approximately optimal Monte Carlo task completion time.

The remainder of this paper is organized as follows. In Section II, we analyze the grid-based Monte Carlo paradigm. We discuss the behavior of grid resources and introduce the *N-out-of-M* scheduling mechanism for grid-based Monte Carlo in Section III and Section IV, respectively. In Section V, we illustrate the characteristics of an ant colony and then in Section VI, we provide our bio-inspired scheduling mechanism. We compare the simulation results of the bio-inspired scheduling mechanism in a simulated computational grid with a simple scheduling and the *N-out-of-M* scheduling mechanisms in Section VII. We also discuss the interesting issues raised by the bio-inspired scheduling

method in Section VIII. Finally, Section IX summarizes our conclusions and discusses future research directions.

II. THE GRID-BASED MONTE CARLO PARADIGM

The dynamic bag-of-tasks (also called bag-of-work) paradigm [CAR, 86, AND, 91, KUA, 02] applies to the situation when the same function is to be executed a large number of times for a range of different parameters. This situation arises naturally in Monte Carlo, as well as in data parallel computations. Applying the function to a set of parameters constitutes a subtask, and the collection of all subtasks to be executed is called the bag of tasks, since they do not need to be solved in any particular order. At each iteration, a worker executes a subtask from the bag and then computes a partial result. Finally, the master assembles the partial results to generate a final result.

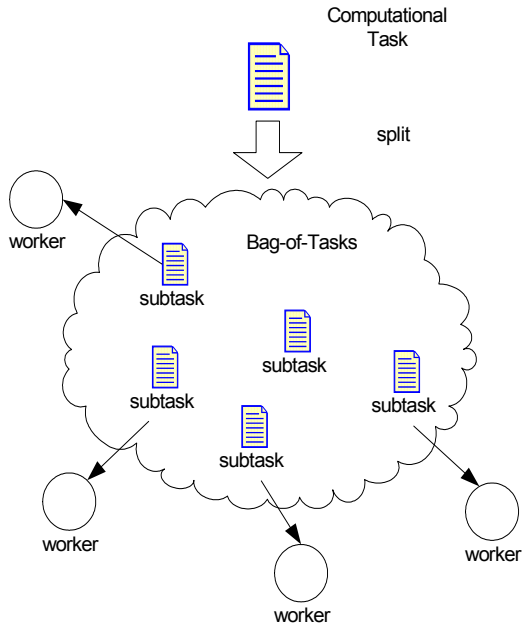

Figure 1: The Dynamic Bag-of-Tasks Paradigm

Bag-of-tasks applications share a generic programming structure. The first step is always to initialize the problem data. Then, the bag of tasks is created, where the global termination condition either represents a fixed number of iterations or is given implicitly in the definition of the problem through something like an error criterion. The actual computation is represented by a loop, which is repeated until the bag of tasks is empty. Multiple workers may execute the loop independently and in parallel. Each worker repeatedly removes a subtask, solves it by applying the main computational function to it, and returns the partial result to the master. Because there is no communication among the workers, the bag-of-tasks paradigm can usually achieve a near-linear speedup. Figure 1 illustrates the generic bag-of-tasks paradigm.

The bag-of-tasks computing paradigm favors applications with naturally parallel characteristics, i.e., situations where the overall computing task can be easily divided into smaller independent subtasks. It is widely assumed that Monte Carlo applications are normally computationally intensive, [FIS, 95] and they tend to work on relatively small data sets while at the same time often consume a large number of CPU cycles. In Monte Carlo, a large task is split into smaller independent subtasks, and each are then executed separately and independently. Each subtask is part of the same problem, i.e., they have the same probability density function definitions and sampling rules, but are executed using different random number streams. During the implementation of a distributed Monte Carlo computation, once a distributed subtask starts, it can usually be executed independently with essentially no communication with other subtasks. Thus, Monte Carlo applications are perceived as naturally parallel, and they can usually be programmed very effectively via the dynamic bag-of-tasks model.

III. RESOURCES IN THE COMPUTATIONAL 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: Grid resources within a computational grid exhibit computational heterogeneity. 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 possible completion times. Moreover, the grid job scheduler may not be able to obtain any indication or information about the performance of a grid node.

Dynamism: Grid computing takes place on 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; and a heavy workload may also turn a "fast" node into a "slow" node [AND, 04].

Trustworthiness: In a grid-computing environment, the service providers of the grid are often geographically separated operating under no central management. Thus, faults may ruin the integrity of a computation. These might include faults arising from the network, system software or node hardware. Also, a node providing CPU cycles might not be trustworthy. A user might provide a node to the grid without the intent of faithfully executing the applications obtained. Experience with SETI@home [KOR, 01] 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 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.

IV. THE *N-OUT-OF-M* SCHEDULING STRATEGY

Monte Carlo applications exhibit statistical characteristics, which enable us to employ the *N-out-of-M* strategy, a more aggressive scheduling mechanism, in order to improve the performance of Grid-based Monte Carlo. The main idea of the *N-out-of-M* strategy [LI, 02, LI, 03] for grid-based Monte Carlo computations is to schedule more subtasks than are required in order to tolerate possible delayed or halted subtasks on the grid to achieve improved performance. The statistical nature of Monte Carlo applications allows us to enlarge the actual size of a computation by increasing the number of subtasks from N to M, where $M > N$. Each of these M subtasks uses its unique and independent random number stream, and we submit M instead of N subtasks to the grid system. When N partial results are ready, we regard the whole task for the grid system to be completed.

Suppose a Monte Carlo computation is executed on a grid system using the dynamic bag-of-tasks model. The task splitting service splits it into N subtasks, with each subtask based on its unique independent random number stream. The scheduling service then schedules each subtask onto the nodes in the grid system. In this case, the assembly of the final result requires all of the N partial results generated from the N subtasks. Each subtask is a "key" subtask, since the suspension or delay of any one of these subtasks will have a direct effect on the completion time of the whole task.

When one is running Monte Carlo applications, what one really cares about is how many random samples (random trajectories) must be obtained to achieve a certain, predetermined, accuracy. One does not much care which random sample set is estimated, provided that all the random samples are independent in a statistical sense. Thus, employing the *N-out-of-M strategy* to enlarge the number of subtasks from $M > N$ subtasks will not change to computational results in a statistical sense. Each of these M subtasks uses its unique independent random number set. Therefore, M bags of computation will be carried out and M partial results may be eventually generated. However, it is not necessary to wait for all M subtasks to finish. When N partial results are ready, we consider the whole task for the grid system to be completed. The Monte Carlo application then collects the N partial results and produces the final result. At this point, the grid-computing system may broadcast abort signals to the nodes that are still computing the remaining subtasks. Figure 2 shows the scheduling mechanism using the *N-out-of-M* strategy.

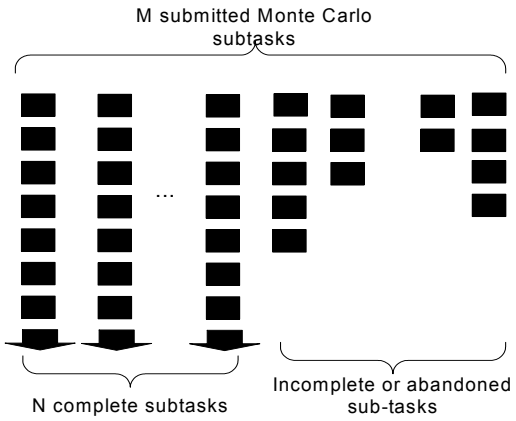

Figure 2: A Scheduling Mechanism using the N-out-of-M Strategy

In this *N-out-of-M* strategy, more subtasks than are needed are actually scheduled, and at most $M - N$ delayed or halted subtasks can be tolerated. However, if there are more than $M - N$ delayed or halted subtasks, the completion of the entire grid computation will still be delayed or halted. Also, the computations on relatively slow grid nodes may be discarded and thus cannot contribute to the whole Monte Carlo computation. If M is significant larger than N, a large amount of computation will be wasted. To more effectively take advantage of the computational power on the grid, a scheduling mechanism that can be adaptive and robust to the dynamic performance of grid is desired. In nature, an ant colony exhibits a collective problem solving capability, which shows strong adaptability and robustness to dynamically changing environments. This interesting property inspires us to develop a bio-inspired scheduling mechanism for grid-based Monte Carlo computing.

V. THE ANT COLONY

Workers of some social insect species specialize in particular tasks, and perform them during the greater part of their lives much more 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, the servants are responsible for keeping the hive clean and warm, and the queen's task is reproduction [DER, 92]. The specialization of social insects associated with morphological adaptations is called "caste polyethism" [WIL, 71]. In a social insect organization, each worker carries out relatively simple functions; however, the collective behaviors of these unsophisticated workers with various specialties cause coherent functional intelligent global patterns to emerge. This "swarm intelligence" exhibits strong adaptability and robustness in dynamically changing environments.

The behaviors of social insects have captured the attention of many scientists due to the problem solving capability achieved with relatively simple individual capabilities. 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 called a pheromone on the ground, forming a pheromone trail. The pheromone trail allows ants to find their way back to the food source or nest. More importantly, other ants can use the pheromone trail deposited by their nest mates to locate 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 follow the pheromone trail with a high probability. If two pheromone trails cross each other, the ants tend to choose, with higher probability, paths marked by stronger pheromone concentrations. Simultaneously, the ants reinforce the trail by depositing their own pheromones. 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 a longer one, and therefore the probability with which any single ant chooses a path to follow is quickly biased towards the shorter one and eventually most of the ants will choose the shorter path. However, the decision of whether to follow a path or not is never deterministic, which permits new routes to be explored. Eventually, the shortest path to the food source will emerge [COL, 92].

The phenomenon of foraging in an ant colony shows that a minimal level of individual complexity can explain sophisticated collective behavior. 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 [DOR, 99], circuit switched communications network problems [DIG, 98], and adaptive routing problems [DIG, 98].

VI. A BIO-INSPIRED JOB SCHEDULING MECHANISM FOR GRID-BASED MONTE CARLO

The goal of every job-scheduling algorithm on the grid is to minimize the execution time of the computational tasks by effectively taking full advantage of the distributed computational resources. In our bio-inspired job scheduling mechanism for grid-based Monte Carlo using swarm intelligence, we design various software ant agents with simple functionalities. No direct communications occurs among these agents. Only indirect communication takes place via the pheromone values stored in a global grid resource table. We expect the collective behavior of these simple agents to suit the dynamic nature of the grid.

A. Grid Resource Table

The only global data structure used in our bio-inspired job scheduling 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 evaporation process. An ant agent can also deposit pheromone by increasing the pheromone value in the grid resource table.

B. Monte Carlo Subtasks

Unlike the large Monte Carlo subtasks (hours to days) in the simple scheduling or the *N-out-of-M* scheduling, in the bio-inspired computation one divides the Monte Carlo task into relatively smaller subtasks. These subtasks usually take minutes to accomplish. These small Monte Carlo subtasks enable even a very slow node to contribute to the overall computation.

C. Specialized Ant Agents

Similar to caste polyethism in social insects, we simulate several agents with distinct and simple functionalities in our 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 notice. The responsibility of the scout is to discover new grid nodes capable of providing appropriate computational services. Once such new nodes are found, the scout adds them to the available resource table with an initial pheromone value.
- Worker: A worker chooses an available grid node and carries out a Monte Carlo subtask in the system on this node. The grid nodes with higher pheromone value

will be assigned a "bigger" subtask with more trajectories.

- Collector: When a Monte Carlo subtask is complete on a grid node, a collector will retrieve the partial results. Also, according to the subtask completion time and the size of the subtask, the collector updates the pheromone value of this particular grid node.
- Cleaner: A cleaner maintains the available grid resource table in the system. Specifically, it removes 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, cleaners, collectors, and workers.

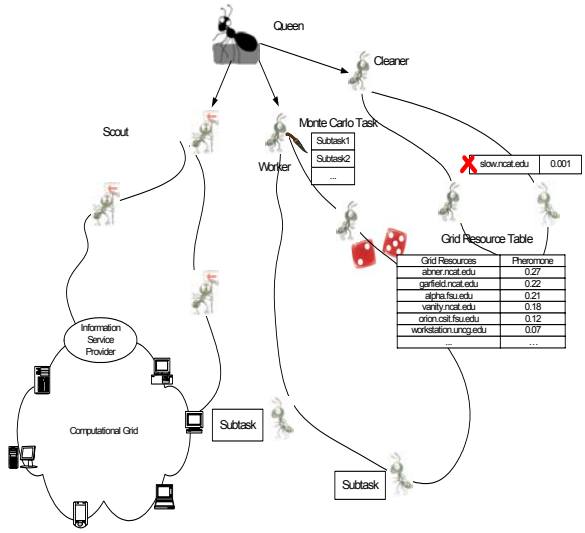

Figure 3: Behaviors of Ant Agents

All these agents fulfill their own simple functions. There is no direct communications among these agents. Figure 3 shows the behaviors and the simple functions of these specialized agents.

D. A Bio-inspired Job Scheduling Mechanism using Swarm Intelligence Algorithm

Let us put all the pieces of the swarm intelligence algorithm together. Suppose a Monte Carlo application requires the evaluation of a total of N random trajectories, and the formulation of a subtask requires at least n random trajectories (samples). The bio-inspired scheduling algorithm for Monte Carlo applications on the computational grid works as follows:

1. Initially, the queen spawns scouts, cleaners, collectors, and workers.
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 of θ.
3. Once a Monte Carlo task is submitted to the computational grid and there is an available grid node, a worker will try to produce a subtask and schedule it on the grid node. Initially, each grid node has the same initial pheromone value, θ, and each Monte Carlo subtask contains T trajectories.
4. If there is more than one grid nodes available, a node having a higher pheromone value will be selected with a higher probability. Node i will be selected with probability q_i given by
$$q_i = p_i / \sum_{j=1}^{n} p_j ,$$
where p_i is the pheromone value of grid node i, and n is the total number of available grid nodes in the system. The subtask submitted to this grid node has $T*p_i/\theta$ random trajectories.
5. When a Monte Carlo subtask is complete on a grid node, the collector will retrieve the partial result and update the pheromone value of the
$$p_i \leftarrow \gamma p_i + \Delta p_i$$
where $\gamma < 1$ is the evaporation constant, and Δp_i is the newly added pheromone amount, which is a function of the completion time of this Monte Carlo subtask.
6. When the pheromone value of a node is lower than a specific threshold value, τ, the cleaner will remove it from the grid resource table. This is because a low pheromone value usually means that either the node has been unavailable for a long time or the node is extremely slow with an undesired job completion time.
7. When the total number of random trajectories evaluated in all complete subtasks exceeds N, the grid Monte Carlo task is complete and the current running subtasks can be aborted.

In this bio-inspired scheduling algorithm, variables T, γ, θ, and τ, are tunable parameters.

VII. PRELIMINARY SIMULATION RESULTS

We use computer simulation to validate the bio-inspired job scheduling mechanism for grid-based Monte Carlo computations. In our simulation program, we simulated a 120-node computational grid. Nodes have a variety of computational capabilities s, which indicates the number of random trajectories (samples) they can process at each time step. We assume that the computational capabilities of the grid nodes are normally distributed with mean m and standard deviation σ.

In our first experiment, we simulate the process of calculating a Monte Carlo task with 10^7 trajectories on the computational grid with $m = 5$ and σ ranging from 1 to 5. We compare the job completion time of simple scheduling, *N-out-of-M* scheduling, and bio-inspired scheduling mechanisms. The simple scheduling divides the Monte Carlo task into 100 subtasks, each assigned 10^5 trajectories, and then assigns these subtasks to randomly chosen grid nodes. The *N-out-of-M* scheduling runs 120 subtasks instead and the Monte Carlo task finishes when 100 subtasks are complete. The bio-inspired scheduling

initially prepares 120 subtasks with 10^3 trajectories. Afterwards, the scheduler adjusts the following subtask sizes using our bio-inspired algorithm according to the pheromone value of the available grid nodes. Figure 4 shows the simulation results. We clearly see that the bio-inspired scheduling mechanism yields better performance, particularly when the computational capabilities of the grid nodes deviate significantly (large standard deviation).

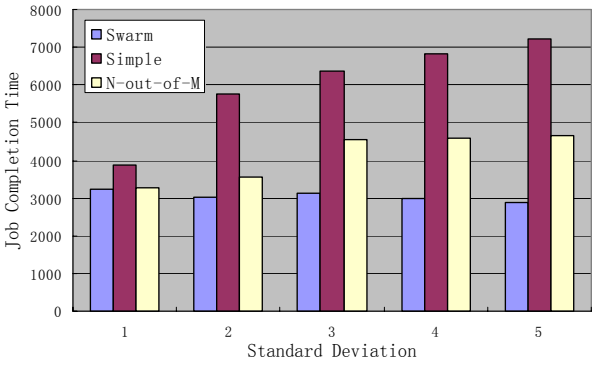

Figure 4: Comparison of Simple, *N-out-of-M*, and Bio-Inspired Scheduling Mechanisms on a Computational Grid in which Grid Nodes have Static Capabilities

In our second simulation experiment, we compare the three scheduling mechanisms on a simulated computational grid in which the computational capabilities of the grid nodes change dynamically. At each time step, a grid node will change to another computational capability with a specified probability, ρ. Figure 5 shows the simulation results with different ρ values, which indicates that the bio-inspired scheduling mechanism exhibits adaptability in a computational grid where nodes have dynamic performance.

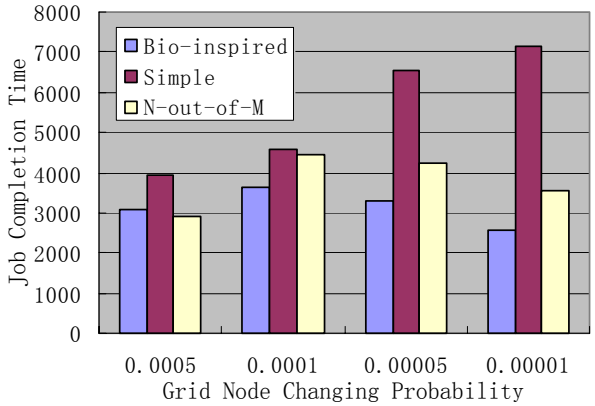

Figure 5: Comparison of Simple, *N-out-of-M*, and Bio-Inspired Scheduling Mechanisms on a Computational Grid in which the Capabilities of the Grid Nodes are dynamically changing with probability ρ

VIII. ANALYSIS OF THE BIO-INSPIRED JOB SCHEDULING MECHANISM

A. Requirement of Parallel Random Numbers

Effectively using the bio-inspired job scheduling mechanism for grid-based Monte Carlo applications requires that the underlying random number streams in each subtask should be independent in a statistical sense, and should not overlap. Compared to the *N-out-of-M* scheduling mechanism, the bio-inspired scheduling mechanism requires more independent random number streams. The SPRNG (Scalable Parallel Random Number Generators) library [MAS, 00] is designed to use parameterized pseudorandom number generators to provide independent random number streams to parallel processes. Some generators in SPRNG can generate up to 2^{78000} - 1 independent random number streams with sufficiently long period and good quality [SRI, 97]. These generators meet the random number requirements of most Monte Carlo grid applications, and specifically meet the requirements here.

B. Temporarily Unavailable Nodes

Also, due to the wide distribution and uncontrollability of grid resources, a grid node may become temporarily unreachable. 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 temporarily unavailable [AND, 04]. 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, will the grid node will be removed from the grid resource table by the cleaner, and its assigned jobs will be rescheduled to other nodes.

C. Reproducibility

Notice that a Monte Carlo computation on the grid using the bio-inspired scheduling mechanism is reproducible. The reason is that we can keep track of which subtasks are involved and which random numbers were used. Then each of these subtasks can be reproduced later. However, the act of reproduction cannot be scheduled in the same way.

IX. CONCLUSIONS

Swarm intelligence algorithms generically exhibit natural adaptability and robustness characteristics through the collaboration of unsophisticated agents interacting with their environment. This can be employed in a highly dynamic computing environment, such as a computational grid. Inspired by the behavior of an ant colony, we presented a bio-inspired job scheduling mechanism that enables the adaptation of naturally parallel and compute-intensive Monte Carlo tasks to a computational grid with heterogeneous and dynamic performance. We applied the bio-inspired mechanism on

a simulated computational grid, and compared it with the simple scheduling mechanism and the *N-out-of-M* scheduling mechanism. Similar to the collective behavior of social insects, our results indicated that the new scheduling scheme is both adaptable and robust within a dynamic grid-computing environment.

The next phase of our research will be to apply the bio-inspired scheduling mechanism for grid-based Monte Carlo applications in a real-life computational grid. We are in the development phase of a grid scheduling software package using the Globus Toolkit [GLO, 04] 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.

ACKNOWNLEGEMENT

The work was supported in part by NSF under the contract HRD-0450203 and University of North Carolina Office of the President under the contract P342A000189.

REFERENCES

[FOS, 01] I. Foster, C. Kesselman, and S. Tueske, "The Anatomy of the Grid," International Journal of Supercomputer Applications, 15(3), 2001.

[LI, 03] Y. Li and M. Mascagni, "Analysis of Large-scale Grid-based Monte Carlo Applications," International Journal of High Performance Computing Applications (IJHPCA), 17(4): 369-382, 2003.

[DEN, 83] 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.

[SHA, 88] J. A. Shapiro, "Bacteria as Multicellular Organisms," Scientific American, pp. 82-89, 1988.

[FIT, 98] T. D. Fitzgerald and S. C. Peterson, "Cooperative Foraging and Communication in Caterpillars," Bioscience, 38: 20-25, 1998.

[BON, 00] E. Bonabeau and G. Théraulaz, "Swarm Smarts," Scientific American, pp. 72-79, 2000.

[CAR, 86] N. Carriero, D. Gelernter, and J. Leichter, "Distributed Data Structures in Linda,' Proceedings of 13th ACM Symp. on Principles of Programming Languages, 1986.

[AND, 91] G. R. Andrews, "Concurrent Programming: Principles and Practice," Benjamin/Cummings, 1991.

[KUA, 02] H. Kuang, L. F. Bic, and M. B. Dillencourt, "Iterative Grid-Based Computing Using Mobile Agents," Proceedings of 31st IEEE International Conference on Parallel Processing, ICPP2002, 2002.

[FIS, 95] G. S. Fishman, "Monte Carlo: Concepts, Algorithms, and Applications," Springer-Verlag New York, 1995.

[AND, 04] 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.

[KOR, 01] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Lebofsky, "SETI@home-Massively distributed computing for SETI," Computing in Science and Engineering, v3n1, 81, 2001.

[DOR, 99] M. Dorigo, G. Di Garo, "Ant Algorithms for Discrete Optimization," Artificial Life, 5:137-172, 1999.

[DIG, 98] G. Di Garo and M. Dorigo, "An Adaptive Multi-Agent Routing Algorithm Inspired by Ants Behavior," Proceedings of PART98 – Fifth Annual Australasian Conference on Parallel and Real-Time Systems, 1998.

[DER, 92] 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.

[WIL, 71] E. O. Wilson, "The Insect Societies," Harvard University Press, Cambridge, 1971.

[DIG, 98] G. Di Garo 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.

[COL, 92] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," Proceedings of First European Conference on Artificial Life, 1992.

[LI, 02] Y. Li, M. Mascagni, "Grid-based Monte Carlo Applications," Lecture Notes in Computer Science, 2536:13-24, GRID2002, Grid Computing Third International Workshop/Conference, Baltimore, 2002.

[LI, 03] Y. Li, M. Mascagni, "Analysis of Large-scale Grid-based Monte Carlo Applications," International Journal of High Performance Computing Applications (IJHPCA), 17(4): 369-382, 2003.

[MAS, 00] M. Mascagni, and A. Srinivasan, "Algorithm 806: SPRNG: A Scalable Library for Pseudorandom Number Generation," ACM Transactions on Mathematical Software, 26: 436-461, 2000.

[SRI, 97] A. Srinivasan, D. Ceperley, and M. Mascagni, "Random Number Generators for Parallel Applications," Monte Carlo Methods in Chemical Physics, D. Ferguson, J. I. Siepmann and D. G. Truhlar, editors, Advances in Chemical Physics series, Wiley, New York, 1997.

[AND, 04] 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.

[GLO, 04] Globus toolkit website, http://www.globus.org, 2004.