A Bee Colony Task Scheduling Algorithm in Computational Grids

Zohreh Mousavinasab[1], Reza Entezari-Maleki[2], and Ali Movaghar[1,2]

[1] Department of Information Technology, Sharif University of Technology, International Campus, Kish Island, Iran
[2] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
mousavinasab@kish.sharif.edu, entezari@ce.sharif.edu, movaghar@sharif.edu

Abstract. The efficient scheduling of the independent and sequential tasks on distributed and heterogeneous computing resources within grid computing environments is an NP-complete problem. Therefore, using heuristic approaches to solve the scheduling problem is a very common and also acceptable method in these environments. In this paper, a new task scheduling algorithm based on bee colony optimization approach is proposed. The algorithm uses artificial bees to appropriately schedule the submitted tasks to the grid resources. Applying the proposed algorithm to the grid computing environments, the maximum delay and finish times of the tasks are reduced. Furthermore, the total makespan of the environment is minimized when the algorithm is applied. The proposed algorithm not only minimizes the makespan of the environment, but also satisfies the deadline and priority requirements of the tasks. Simulation results obtained from applying the algorithm to different grid environments show the prominence of the algorithm to other similar scheduling algorithms.

Keywords: Task scheduling, grid computing, bee colony optimization, makespan, delay time.

1 Introduction

Grid computing [1] is a large scale distributed environment designed for solving the computational- and data-intensive problems in science and industry. Actually, the grid is an infrastructure which supplies a mechanism to run the applications over computational resources which are heterogeneous and geographically distributed. The computational resources within the grid environments may belong to the various individuals and institutions [1], [2]. Bringing the computational resources together from various administrative domains provides a tremendous computational environment to execute the tasks submitted to the environment. In order to execute the tasks, two factors are of great importance in the grid computing environments: resource management and task scheduling [1], [3]. Resource management provides the possibility to have access to all of the resources within the grid environment, regardless of their platforms and hardware architectures. Task scheduling algorithms

V. Snasel, J. Platos, and E. El-Qawasmeh (Eds.): ICDIPC 2011, Part I, CCIS 188, pp. 200–210, 2011.

are needed to effectively use the tremendous capabilities of the grid computing environments. Grid managers apply task scheduling algorithms to dispatch the submitted tasks among the grid resources appropriately. Generally, the grid tasks are submitted to the grid managers by grid users and then the manager schedules the tasks within the available resources [4].

In order to schedule the tasks to the multiple resources distributed within the grid environments, the need for an efficient and proper scheduling algorithm is sensed. Since the scheduling is an NP-complete problem, many heuristic approaches have been proposed to appropriately schedule the tasks within the grid resources. Heuristic algorithms produce a good solution by solving a simpler problem that contains or intersects with the solution of a more complex problem.

Due to the study of the behavior of social insects, a source of inspiration for the design and manipulation of optimization algorithms has been found by computer scientists recently. Nature-inspired algorithms are considered by the ability of biological systems to efficiently regulate the mostly changeable environments. These algorithms include evolutionary computation, neural networks, ant colony optimization, particle swarm optimization, artificial immune systems, and bacteria foraging algorithm [5], [6]. Swarming and showing the different characteristics and behaviors are the main specifications of the various colonies of social insects such as bees, wasps, ants and termites. The behavior of the social insects is first and foremost characterized by autonomy, distributed functioning and self-organizing [5]. Swarm intelligence is the part of the artificial intelligence that is based on the study of actions of individuals in various decentralized systems [7]. Recently, the bee colony optimization (BCO) has been introduced as a new horizon in the discussion of swarm intelligence [4]. Artificial bees stand for agents, which are collaborative solutions to solve the complex combinatorial optimization problems. The impression BCO suggests a multi agent system (colony of artificial bees) to solve the complex optimization problems [5]. The study of bee colonies proved to be interesting enough for developing problem-solving algorithms. For this reason, a new task scheduling algorithm inspired by bee colonies is presented in this paper.

The rest of this paper is organized as follows. Section 2 presents the related works done in the field of task scheduling algorithms and optimization problems using BCO. In Sect. 3, the concepts of the BCO are introduced. Section 4 proposes the new task scheduling algorithm based on BCO. Section 5 presents the simulation results and compares the proposed algorithm against the well-known scheduling algorithms in a hypothesis grid environment. Finally, Sect. 6 concludes the paper and presents future work.

2 Related Works

Since the BCO is a meta-heuristic and can be displayed as a general algorithmic framework, it has been able to apply various optimization problems in management, engineering, and control [5], [6]. On the other hand, the problem of task scheduling in distributed systems and especially in grid environments has been souled by various heuristic methods such as genetic algorithms (GAs), particle swarm optimization (PSO), simulated annealing (SA), ant colony and so forth [8], [9]. In the below, some

of the algorithms related to the task scheduling problem are introduced. Also, the several research works which use the BCO as an optimization technique in various optimization problems are introduced.

Parsa et al. [3] have presented a new task scheduling algorithm which is called RASA in grid environments with the goal of reducing the makespan of the grid resources. RASA takes advantages of two algorithms, Min-min and Max-min [10], and avoids their drawbacks. Entezari-Maleki et al. [11] have proposed a genetic-based scheduling algorithm to reduce the makespan of the grid applications.

The proposed algorithm in Ref. [11] schedules the submitted tasks to the grid resources considering the completion time of the tasks on each of the resources. The simulation results show that the proposed algorithm reaches a good level of throughput in comparison with the others. But, none of the algorithms presented in Refs. [3] and [11] consider the deadline and arriving rate of the tasks, as well as the cost of the task execution on each of the resources.

Zheng et al. [8] have offered a model that combines two optimal schemes, genetic algorithm (GA) and simulated annealing (SA), based on the probability distribution. Actually, the algorithm uses the benefits of the genetic algorithms and simulated annealing and provides a parallel genetic-simulated annealing algorithm to solve the task scheduling problem in grid environments. Liu et al. [9] have proposed an improved ant colony algorithm which is called adaptive ant colony algorithm for grid task scheduling which can balance the loads of the resources within the grid environment. The algorithm has avoided impressively the phenomena of hunger and starvation on the resources due to the fighting for preponderant resources by more tasks.

Davidovic et al. [12] have presented the BCO heuristic algorithm which is inspired by bees behavior for task scheduling problem on homogeneous processors. This algorithm has the ability to obtain the optimal value of the objective function in all small to medium size test problems. However, there is no theoretical background at the moment that could support the presented approach.

Wong et al. [13] have presented a BCO algorithm for traveling salesman problem (TSP). The BCO algorithm is made based on the collective intelligence shown in bee foraging behavior. In Ref. [13] both approaches are integrated to solve TSP; the BCO model and the 2-opt heuristic. Actually, the BCO model is further improved by using a local search approach, named 2-opt heuristic, that increased the performance of the algorithm significantly.

Chong et al. [14] have offered a novel approach that uses the honey bees for aging model to solve the job shop scheduling problem. In the construction industry, job shop scheduling is an important task for improving machine utilization and decreasing cycle-time. The proposed algorithm is based on self-organization of a honey bee colony. Although the algorithm has better performance of ant colony algorithm, Tabu search heuristics work more efficient than the proposed algorithm. Quijano et al. [15] have proposed the algorithm based on foraging habits of honey bees to solve a class of optimal resource allocation problems. The algorithm uses the strategy of evolutionarily stable strategy where several such algorithms compete in the same problem domain. In addition, the authors have proved that they can achieve an ideal free distribution for a single hive or multiple hives in this problem.

3 Bee Colony Optimization

On study of individuals in multiple decentralized systems, swarm intelligence is thought to constitute part of artificial intelligence. As a new horizon, the BCO meta-heuristic has been introduced as one of the approaches in swarm intelligence. Artificial bees represent agents which form collaborative solutions to issues of complex combinational optimization. The artificial bee colony behaves partly similar to and partly different from bee colonies in nature. They search the state space of the problem to seek the possible solutions. Artificial bees work with each other and exchange information to attain the suitable solutions [13]. In order to use BCO in grid scheduling algorithms, the required preliminaries should be mentioned. To do this, first the behavior of the bees in nature is briefly introduced and then, a short introduction of a general BCO algorithm is given.

3.1 Bees in the Nature

Bee system consists of two essential components: food sources and foragers [5]. Food sources depend on different parameters such as its proximity to the hive, richness of the energy and ease of extracting this energy. Foragers consist of two types: *unemployed* and *employed foragers* [12].

If it is assumed that a bee has no knowledge about the food sources in the search field, bee initializes its search as an unemployed forager. There are also two types of unemployed foragers: *scout* and *recruit* bees [13]. If the bee starts searching spontaneously without any knowledge, it will be a scout bee. The percentage of scout bees varies from 5% to 30% according to the information into the hive. The mean number of scouts averaged over conditions is about 10%. The scout bee returns to hive and announce the found locations as well as the amount and the quality of pollen, nectar and propolis. The scout bee exchanges this information by a special sort of dancing called *waggle dance. Waggle dance* is done in a special area of the hive called *dance floor* to expand the sources of food. If the unemployed forager attends to a waggle dance done by some other bees, the bee will start searching by using the knowledge from waggle dance. These types of the unemployed foragers are named recruit bees. When a recruit bee finds and exploits the food source, it will raise to be an employed forager who memorizes the location of the food source. After the employed foraging bee loads a portion of nectar from the food source, it returns to the hive and unloads the nectar to the food area in the hive [12].

Considering the above mentioned bee types and their jobs, three different situations are possible for a foraging bee: 1. the bee can discard the food source and become again an uncommitted follower, 2. it can continue to forage at the food source without recruiting the nestmates, 3. it can recruit other bees with the dance to return to the food location. The bee decides to choose one of these three options due to the quality and quantity of the food resource as well as its distance from the hive.

3.2 BCO Algorithm

Lucic and Teodorovic [5], [16] are the first researchers who applied the basic principles of collective bee intelligence in figuring out combinational optimization

problems. The BCO is an algorithm based on population in which the population of the artificial bees seeks the best solution to which every artificial bee contributes. After the bees partly attain solutions, which is in fact the first phase of their job called *forward pass*, they get back to the hive, beginning the second phase called *backward pass*. During the latter phase, the bees exchange all information and solutions. While the dancing process of bees in nature is aimed at telling other bees how much food exists and how far the source is to the hive, the bees do this in the search algorithm to announce the quality of the solution. In fact, every bee decides with a certain probability whether it will advertise its solution or not. Those with better solutions are more successful in advertising their solutions. The rest of the bees either keep exploring their own solutions in the next so called forward pass or proceed to explore the neighborhood of one of the advertised solutions. Thus, the best solutions have always more chance of attracting attentions to be explored. The two phases are performed iteratively until a condition such as the maximum total number of forward/backward passes is met to stop the whole process.

4 The Proposed Algorithm

The task scheduling algorithm proposed in this paper is based on BCO. The general scheme of the algorithm is presented in Fig. 1.

As shown in Fig. 1, requests or tasks are entered to the system by grid users. When a task is delivered to the system, the priority of the task toward the others and also the deadline of the task is specified. After the arrival of the tasks, a unique ID is assigned to each of the tasks by pre-scheduler. Then, the task is sent to a resource named Priority Based Bee Scheduling (PBBS). All of the tasks are queued inside PBBS to be dispatched within grid resources.

After that, PBBS selects a task considering its priority and deadline. A task with the highest priority and the lowest deadline is selected firstly. Then, the PBBS randomly assigns an algorithm to the task and sends the task information to the Knowledge Base which uses the bee colony algorithm. The Knowledge Base includes one table with the following fields about the task: ID, arrival time, priority, dead time, data size, start time, execution time, finish time, delay time, execution algorithm and the destination resource allocated to the task.

All required information for each task should be saved in the table. The tasks are sorted in the table considering the mixed value called objective function value. The objective function considers the data and computational size of each of the tasks. Actually, the position of a task in the table totally depends on its objective function value. In other words, the algorithm searches the table to find the best position for the new submitted task based on its objective function value.

After finding a best possible position for a submitted task, the Knowledge Base selects a suitable algorithm considering the minimum finish time of the tasks inside the table in that acceptable range. This range can vary based on the implementation of the algorithm, e.g. in our algorithm, the best position for a task with computational size equal to MI is the range of $\left[MI - \frac{1}{10} MI, MI + \frac{1}{10} MI \right]$. Then, the Knowledge Base

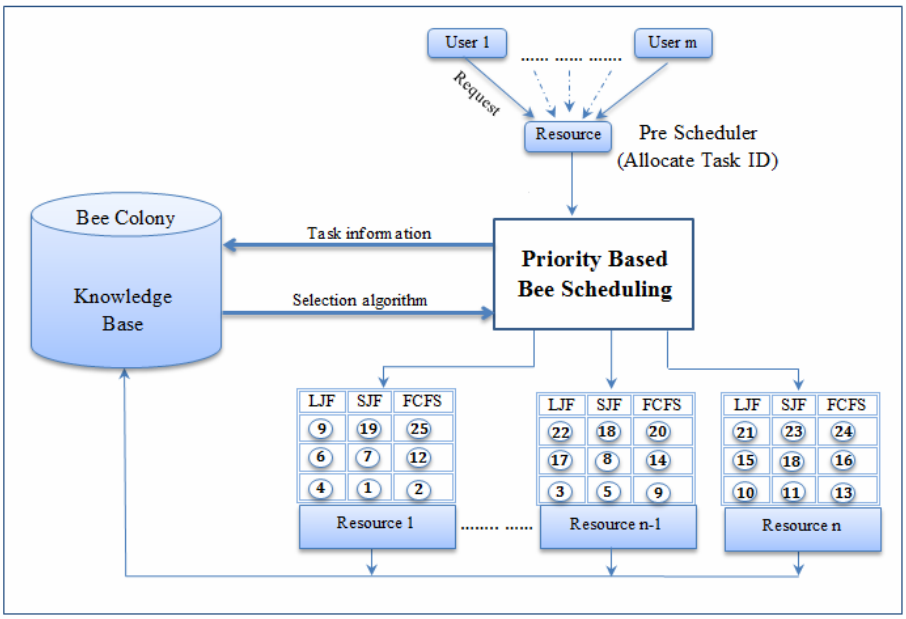

Fig. 1. The general scheme of the proposed algorithm

sends the best algorithm to the PBBS to calculate the p value of the algorithm using Eq. 1.

$$p_{tID}^{u+1} = e^{-\frac{f_{\max} - f_{\min}}{u}} \tag{1}$$

Where u shows the number of the forward passes in the BCO which is equal to one in our algorithm, tID denotes the ID of the task and $f_{\max}$ and $f_{\min}$ denote the maximum and minimum value for the finish time of the task, respectively.

Finally, the PBBS attaches the selected algorithm to the task considering its p value. The p value is more efficient when the numerator in the formula (i.e. the difference between the maximum finish time and the minimum finish time) is a large number.

There is a need to save some information about the resources inside PBBS. Actually the ID number of the resources, the bandwidth of the communication links corresponding to each of the resources and the computational power of the resources should be saved in PBBS. The PBBS calculates the computation and data transmission time of the tasks for all of the existing resources. After that, PBBS selects the best resource by the minimum computation time to execute the task. In the proposed BCO algorithm, three algorithms, First Come First Served (FCFS), Shortest Job First (SJF) and Longest Job First (LJF), in each of the resources are implemented. When the PBBS selects a resource to execute a specified task, one of these three algorithms is also specified to be assigned the task. This assignment is made by considering the logs existing in the Knowledge Base. Therefore, the PBBS sends the

task to the specific queue (related to the selected algorithm) on the chosen resource. Each resource executes the tasks in its own queues considering the deadline of the tasks, so the task with the nearest deadline has higher priority than others inside a resource. After executing the task inside the resource, the resource sends all of the tasks information such as finish time, delay time and execution time to the Knowledge Base. Then, the Knowledge Base updates the task table by the new data.

In the beginning of the algorithm, some random tasks are entered into the PBBS. Later, an algorithm is randomly assigned to each task for training the system. The more random tasks we have in the training process, the more noticeably optimized will be the Knowledge Base. The main steps of the proposed algorithm are illustrated below.

- **While** there is any unscheduled task **do**
    1. Select task *T* based on maximum priority and minimum deadline and then update tasks table,
    2. For each of the resources calculate the computation and data transmission times and then compute the task completion time,
    3. Select the minimum completion time,
    4. **Until** the number of the tasks existing in the training set is less than the pre-specified number **do**
        a. Randomly select the scheduling algorithm between FCFS, SJF and LJF algorithms,
        b. Send the information of task *T* to the Knowledge Base,
        c. Go to step 7.
    5. For each task (or bee) do the forward pass (in the initialization phase, randomly select one of the scheduling algorithms, FCFS, SJF and LJF, and assign it to the task),
    6. For each task do the backward pass:
        a. Send the information of task *T* to the Knowledge Base (This step is equal to the returning of all of bees to the hive),
        b. Sort the tasks' table by the value of tasks' objective functions,
        c. Considering the objective function of the tasks, choose a suitable algorithm for each of the tasks which results in the lowest finish time,
        d. Each task decides with probability p_{tID}^{u+1} to continue its own algorithm. The value of p_{tID}^{u+1} can be calculated using Eq. 1.
    7. Attach the selected algorithm to task *T*,
    8. Send task *T* to the selected queue on the resource R,
    9. Execute task *T* on the resource R based on the nearest deadline,
    10. Update Knowledge Base after finishing the execution of the task.
- End **While**.

5 Simulation Results

To measure the performance of the proposed algorithm and compare the algorithm to the other algorithms, FCFS, SJF and LJF, various case studies have been considered. To achieve more realistic results, several sample grid environments with different number of the resources and tasks have been simulated and the important metrics such as tasks delay times, tasks finish times and total makespan of the environment have been evaluated on these environments. For the sake of brevity, only one scenario is discussed and the related plots showing the comparison results are demonstrated in this section. The other results in various situations which are not presented here emphasize the following statements more.

Assume there is a grid environment with two different resources. The processing speed of the grid resources and their related bandwidth are shown in Table 1. All of these values are generated randomly.

Table 1. Specification of the grid resources

Resources	Processing Speed (MIPS)	Related Bandwidth (Mbps)
R_1	150	175
R_2	225	200

To simulate different loads on the grid resources, various numbers of the tasks are entered to the environment and the required metrics are evaluated when the tasks are dispatched and executed on the resources. Figures 2 and 3 show the delay times of the tasks when the four mentioned algorithms are applied to the submitted tasks. As shown in Fig. 2 and Fig. 3, the BCO algorithm shows lower maximum and average delay times in comparison to the other algorithms. Figures 4 and 5 show the finish times of the tasks for each of the algorithms. As shown in Fig. 4 and Fig. 5, the maximum finish time of all the submitted tasks which can be considered as makespan of the environment, is lower than other algorithms' makespans.

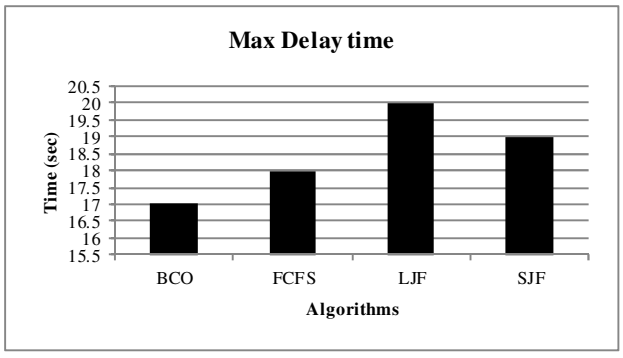

Fig. 2. Maximum delay times of the algorithms

It is necessary to mention that all of the tasks' properties such as arrival times, priorities, execution times and other specifications mentioned earlier have been generated randomly in all of the case studies.

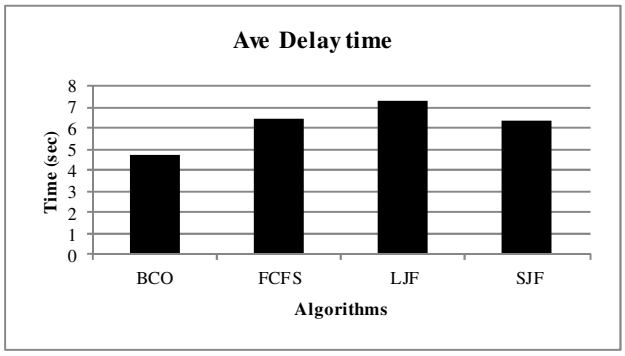

Fig. 3. Average delay times of the algorithms

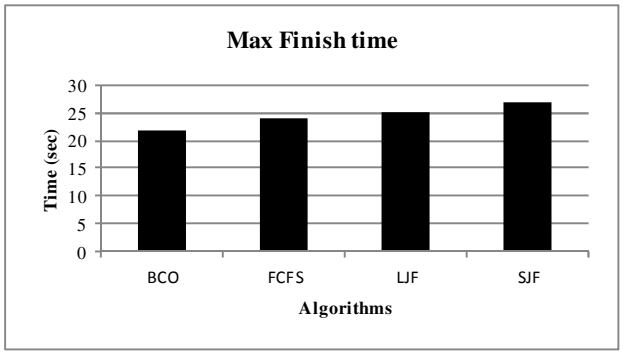

Fig. 4. Maximum finish times of the algorithms

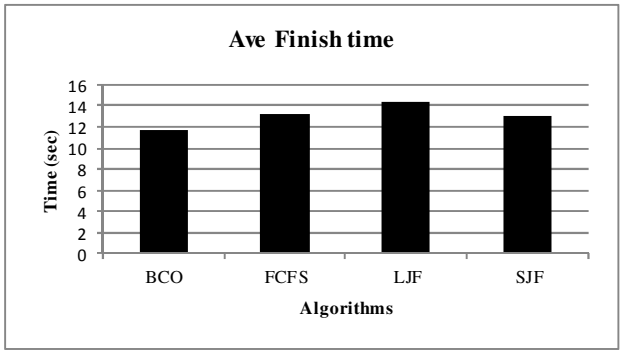

Fig. 5. Average finish times of the algorithms

6 Conclusions and Future Work

To solve the scheduling problem in grid environments, a new algorithm inspired from BCO is presented in this paper. The proposed algorithm can be applied to schedule grid tasks considering their priorities and deadlines to reduce the maximum and average delay and finish times of the tasks. Reducing the maximum finish times of the tasks, the total makespan of the grid environment can be minimized and thereby the throughput of the environment increases. The algorithm proposed in this paper not only uses very simple method to represent the resource allocation using BCO, but also exposes lower makespan values in comparison with the standard scheduling algorithms. Furthermore, the proposed algorithm converges to the suitable solution for the large number of tasks submitted to the grid environment.

There are numbers of research issues remaining open for future work. One can use other QoS measures (instead of finish and delay times of the tasks) to evaluate the bees behavior and find the best possible match between the tasks and resources. Taking into account other criteria (e.g. cost of the scheduling, performance and reliability of the grid environment and so forth) may result in new scheduling algorithms in grid environment. In addition, we expect to improve the algorithm selection mechanism within Knowledge Base by using new methods and probability formulas. Also, improving the behavior of bees to recognize and eliminate the wrong actions is one of the open problems in this algorithm which can be considered in future work.

References

1. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure, 2nd edn. Elsevier and Morgan Kaufmann Press (2004)
2. Khanli, L.M., Analoui, M.: Resource Scheduling in Desktop Grid by Grid-jqa. In: The 3rd IEEE International Conference on Grid and Pervasive Computing, pp. 63–68 (2008)
3. Parsa, S., Entezari-Maleki, R.: Rasa: A New Grid Task Scheduling Algorithm. International Journal of Digital Content Technology and its Applications 3(4), 91–99 (2009)
4. Ferreira, L., Berstis, V., Armostrong, J., Kendzierski, M., Neukoetter, A., Takagi, M., Bing-Wo, R., Amir, A., Murakawa, R., Hernandez, O., Magowan, J., Ieberstein, N.: Introduction to Grid Computing with Globus, 2nd edn, IBM International Technical Support Organization (2003)
5. Teodorovic, D., Davidovic, T., Selmic, M.: Bee colony optimization: The Applications Survey. ACM Transactions on Computational Logic (Published online)
6. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence, 1st edn. Oxford University Press, Oxford (1997)
7. Yanli, H., Lining, X., Weiming, Z., Weidong, X., Daquan, T.: A knowledge-based ant colony optimization for a grid workflow scheduling problem. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 241–248. Springer, Heidelberg (2010)
8. Zheng, S., Shu, W., Gao, L.: Task Scheduling Using Parallel Genetic Simulated Annealing Algorithm. In: IEEE International Conference on Service Operations and Logistics, and Informatics, pp. 46–50 (2006)

9. Liu, A., Wang, Z.: Grid Task Scheduling Based on Adaptive Ant Colony Algorithm. In: International Conference on Management of e-Commerce and e-Government, pp. 415–418 (2008)
10. He, X., Sun, X.H., Laszewski, G.V.: Qos Guided Min-Min Heuristic for Grid task Scheduling. Journal of Computer Science and Technology 18(4), 442–451 (2003)
11. Entezari-Maleki, R., Movaghar, A.: A Genetic-Based Scheduling Algorithm to Minimize the Makespan of the Grid Applications. In: Kim, T., Yau, S., Gervasi, O., Kang, B., Stoica, A., Slezak, D. (eds.) GDC and CA 2010. CCIS, vol. 121, pp. 22–31. Springer, Heidelberg (2010)
12. Davidovic, T., Selmic, M., Teodorovic, D.: Scheduling independent tasks: Bee Colony Optimization Approach. In: 17th Mediterranean Conference on Control and Automation, pp. 1020–1025 (2009)
13. Wong, L.P., Low, M.Y.H., Chong, C.S.: A Bee Colony Optimization Algorithm for Traveling Salesman Problem. In: 6th IEEE International Conference on Industrial Informatics, pp. 1019–1025 (2008)
14. Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.L.: A Bee Colony Optimization Algorithm to Job Shop Scheduling Simulation. In: Perrone, L., Wieland, F., Liu, J., Lawson, B., Nicol, D., Fujimoto, R. (eds.) The Winter Simulation Conference, pp. 1954–1961 (2006)
15. Quijano, N., Passino, K., Univ, M.: Honey Bee Social Foraging Algorithms for Resource Allocation. In: American Control Conference, pp. 3389–3394 (2007)
16. Lucic, P., Teodorovic, D.: Computing with Bees: Attacking Complex Transportation Engineering Problems. International Journal on Artificial Intelligence Tools 12(2), 375–394 (2003)