Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 8958170, 8 pages
http://dx.doi.org/10.1155/2016/8958170

Research Article

A Trusted Real-Time Scheduling Model for Wireless Sensor Networks

Weizhe Zhang, Boyu Song, and Enci Bai

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China

Correspondence should be addressed to Weizhe Zhang; wzzhang@hit.edu.cn

Received 24 November 2015; Accepted 19 January 2016

Academic Editor: Josep Samitier

Heterogeneous multicore and multiprocessor systems have been widely used for wireless sensor information processing, but system energy consumption has become an increasingly important issue. To ensure the reliable and safe operation of sensor systems, the task scheduling success rate of heterogeneous platforms should be improved, and energy consumption should be reduced. This work establishes a trusted task scheduling model for wireless sensor networks, proposes an energy consumption model, and adopts the ant colony algorithm and bee colony algorithm for the task scheduling of a real-time sensor node. Experimental result shows that the genetic algorithm and ant colony algorithm can efficiently solve the energy consumption problem in the trusted task scheduling of a wireless sensor and that the performance of the bee colony algorithm is slightly inferior to that of the first two methods.

1. Introduction

Energy consumption has become a major problem in wireless sensor networks. Sensor nodes are often equipped with small batteries [1]. Therefore, the reduction of energy consumption to prolong the lifetime of networks is a widespread concern. Wireless sensor networks perform mainly real-time tasks. We thus need to focus on reducing energy consumption under the premise of meeting specified task deadlines.

The real-time task scheduling of wireless sensor networks involves forming a mapping relationship between the real-time task scheduling and the processors within the acceptable time scope of the system. The real-time tasks are then assigned to the processors according to the mapping relationship. In this way, the tasks can be executed within the deadline. The mapping relationship should effectively meet time and resource constraints.

In the process of establishing time scheduling models, the operating environment of the real-time calculation must be considered because calculation resources consist of different heterogeneous hosts, workstations, and even PCs that are distributed in different locations. These resources can be run in a variety of operating systems, such as UNIX, Linux, and Windows, and the storage media may be large storage devices, databases, or other devices. Therefore, a scheduling model and energy consumption model must be built specifically for heterogeneous platforms.

Heterogeneity is mainly reflected in the following aspects. (1) Computer hardware heterogeneity: a server processor that executes calculation operations may be heterogeneous. It may cause considerable differences in energy consumption when processing the same tasks in unit time. (2) Operating system heterogeneity: the heterogeneous components of an operating system are assigned to a particular machine, the sequence of tasks to be executed may be disrupted, and the execution time could either increase or decrease. (3) Heterogeneity of communication network: the topological structure of a network may resemble items such as a bus, ring, and tree, and the communication medium may be a cable, optical fiber, microwave, and so on. Such variance results in different transmission rates and different transmission time periods. Heterogeneity degree can be used to evaluate the degree of difference among heterogeneous computing systems. A large heterogeneity degree is associated with obvious system heterogeneity. Hence, heterogeneity degree can serve as an important reference and basis for selecting task scheduling strategies.

Research results [2] indicate that the execution power and response time of different computers vary in the implementation of different computing tasks. For example, when a CPU and GPU execute the same image processing task, their execution power and response time, as well as their total energy consumption for completing the task, will vary. Therefore, as a result of different energy consumption factors, a given task may be completed at different periods and with different levels of energy consumption when different scheduling methods and supply voltages are used. Under such circumstances, the energy consumption problem should be considered when carrying out parallel task scheduling in a real-time computing environment.

Several studies have established energy consumption models on the basis of resource utilization rates. Li and John [3] proposed a hardware counter-based linear model to estimate the energy consumption of an operating system. Topcuoglu et al. [4] proposed a simple linear regression model based on CPU utilization rate and hard disk utilization, proved that the model is accurate, and used the model to reduce the energy consumption of the server cluster. By establishing an inquiry table for system power and resource utilization rate, the authors [4] established an energy consumption model for a server system and proposed several methods for applying the model to different systems of the same series. Currently, energy consumption optimization management technologies based on distributed parallel computing systems are divided into three categories: off/sleep technology, dynamic voltage scaling (DVS) technology, and virtualization technology. The off/sleep technology is mainly used to reduce idle energy consumption. The other two technologies are mainly used to reduce the execution power consumption. Execution power consumption is defined as the power consumption generated by the operation of computer hardware driven by commands and data when using a computer to complete a task. The power consumption of the operated hardware in this case is referred to as execution power. In the execution of the same task, the execution power may constantly change under different implementation phases and given different task characteristics [5].

In 2010, Wang et al. [6] proposed a scheduling algorithm for energy consumption perception and a method for calculating idle time. The proposed scheduling algorithm considers the method for reducing voltage in the communication process. In 2009, Khan and Ahmad [7] studied task allocation for energy consumption perception and assigned tasks to a computing grid with DVS features. The goal of such task assignment is to reduce energy consumption and response time [8]. In China, Zhu et al. [9] considered the situation in which copying a scheduling algorithm can reduce the waiting time and time delay but increase energy consumption and subsequently proposed a heuristic processor reduction optimizing method. Kwok and Ahmad [10] studied network energy consumption models and algorithm networks from a global perspective, established a systematic optimization model of network energy consumption, proposed a building network system for optimizing energy consumption models, and proposed a network packet routing algorithm for optimizing energy consumption. Lee and Zomaya [11] established the HCS model according to DVS technology and developed the ECS algorithm, which is an innovation of the energy consumption model with good usability [12, 13].

The privacy-preserving techniques for WSNs are classified into two for protecting two types of private information: data-oriented and context-oriented privacy [14]. KIPDA [15] is a data-oriented privacy-preserving aggregation method, which can hide a wide range of aggregation functions. Ozdemir et al. present a polynomial regression based data aggregation protocol that preserves the privacy of sensor data [16].

The rest of the paper is organized as follows. Section 2 introduces the system model and formulates the problem of real-time task scheduling in a heterogeneous wireless sensor network. Section 3 introduces the genetic algorithm and ant colony algorithm, which are adopted to obtain the shortest scheduling length and lowest energy consumption. Section 4 presents a new bee colony algorithm. Section 5 provides an analysis of the performance and results of the proposed algorithms. Section 6 concludes the paper with a summary and details of future research directions.

2. System Model

2.1. Real-Time Task DAG Model. A real-time parallel task model can be abstracted as a DAG diagram $\mathrm{DAG} = (T, E, W, D)$, in which $T = \{t_1, t_2, \dots, t_n\}$ represents the collection of tasks; n indicates the number of tasks; $E = \{e_{ij} \mid t_i, t_j \in T\} \subseteq T \times T$ denotes the collection of directed edges that describe the dependency relationship between tasks; $W = \{w_1, w_2, \dots, w_n\}$ is the computational quantity collection of tasks, with $w_i \in W$ representing the number of clock ticks of task t_i in the processor; and D is the communication between tasks $\{d_{ij}\}$, with $d_{ij} \in D$ representing the duration of communication between the two ends of e_{ij}.

The earliest starting time of each task in the parallel task map is defined as $\mathrm{Est}(t_{\mathrm{entry}}) = 0$, $\mathrm{Est}(t_i) = \max_{t_p \in \mathrm{pred}(t_i)}\{\mathrm{Est}(t_p) + \mathrm{Met}(t_p) + \mathrm{Act}_{pi}\}$, where t_{entry} represents the entry subtask without a precursor node, $\mathrm{Met}(t_p)$ represents the minimum execution time of task t_p, and Act_{pi} represents the communication time between task t_p and task t_i. The earliest completion time of each task $\mathrm{Eft}(t_i)$ is defined as $\mathrm{Eft}(t_i) = \mathrm{Est}(t_i) + \mathrm{Met}(t_i)$. $\mathrm{Lft}(t_i)$ is the latest completion time of task t_i, $\mathrm{Lft}(t_{\mathrm{exit}}) = D$, and $\mathrm{Lft}(t_i) = \min_{t_c \in \mathrm{Succ}(t_i)}\{\mathrm{Lft}(t_c) - \mathrm{Met}(t_i) - \mathrm{Act}_{ic}\}$, where t_{exit} denotes the exit subtask without a successor node. The deadline of each task is $\mathrm{Lft}(t_i)$, which is the latest completion time of task t_i.

In Figure 1, t_i in the inner circle represents the serial number of the task node, the digit adjacent to the circle represents the computation amount of the task node, and the digit on the arrow line represents the amount of communication between the two tasks. After processing, the DAG diagram of the parallel task is assumed to have only one entry node and one exit node.

2.2. Processor Model. The processor is represented by the symbol P. We assume that each processor P only executes one command within one clock tick and that different processors have different operating speeds. A processor has two properties, namely, processing frequency and corresponding voltage. Let $s_{i,j}$ denote the clock rate of task T_i in processor P_j,

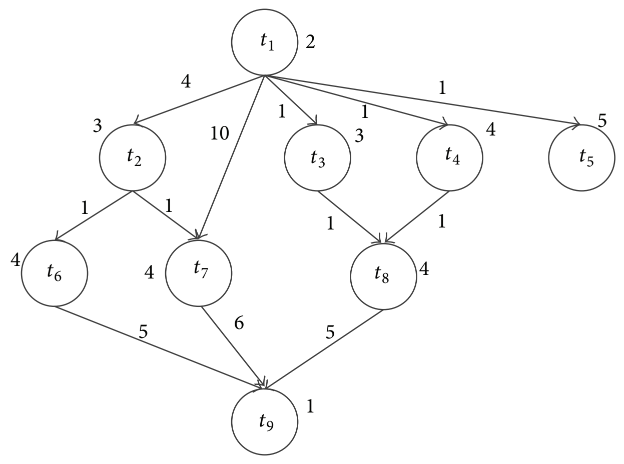

Figure 1: Example of DAG task map.

and let $e_{i,j}$ denote the execution time of task T_i in processor P_j. Then, $e_{i,j} = c_i/s_{i,j}$, in which c_i represents the calculation amount of the corresponding T_i.

The platform (heterogeneous multiprocessor platform, HMP) comprising m heterogeneous processors is expressed as HMP $= \{P_1, P_2, \ldots, P_m\}$; m processors have different clock rate-voltage pairs.

2.3. Scheduling Model. A scheduling model is represented by a scheduling matrix $X_{n\times m}$. In $X_{n\times m}$, each element $x_{i,j}$ represents the corresponding relation between task T_i and processor P_j, and $x_{i,j}$ can only be set to 0 or 1, with $x_{i,j} = 1$ indicating that task T_i is assigned to processor P_j and with $x_{i,j} = 0$ indicating that task T_i is not assigned to processor P_j.

2.4. Constraint Model. The effect matrix of the task scheduling problem in a heterogeneous processor is $U_{n\times m}$. The element $u_{i,j}$ of $U_{n\times m}$ is defined as $u_{i,j} = e_{i,j}/p_i$, which represents the consumed computing ability of P_j when performing T_i. The value of $u_{i,j}$ is a real number within the scope of $(0, 1) \cup +\infty$. If T_i cannot be operated in P_j, then $u_{i,j}$ can be set to $+\infty$.

According to the definition of the effect matrix $U_{n\times m}$, the constraints [17] of task scheduling in the heterogeneous processor are given. The constraints are as follows:

(a) $\sum_{j=1}^{m} x_{i,j} = 1$, $(i = 1, 2, 3, \ldots, n)$;

(b) $\sum_{i=1}^{n} x_{i,j} * u_{i,j} \leq U$, $(j = 1, 2, 3, \ldots, m)$, $U = 1$, is the maximum computing amount allowed by each processor;

(c) $x_{i,j}$ is set as 0 or 1 $(i = 1, 2, 3, \ldots, n;\ j = 1, 2, 3, \ldots, m)$.

In (a), each task is fully allocated to the processor. In (b), the total calculation amount of the tasks assigned to each processor is not greater than the maximum allowable amount of calculation. In (c), the task is either assigned or not assigned to a processor.

Each specific element value of the scheduling matrix is 0 or 1. Under the restriction of the constraint model, the size of each task is 10 5 × 10 scheduling matrixes $X_{5\times10}$, as shown in Table 1.

Table 1: Scheduling matrix example $X_{5\times10}$.

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
P_0	0	0	1	0	0	1	0	0	0	0
P_1	0	0	0	0	1	0	0	0	1	0
P_2	1	0	0	0	0	0	0	1	0	0
P_3	0	0	0	1	0	0	1	0	0	1
P_4	0	1	0	0	0	0	0	0	0	0

In the scheduling matrix of $X_{5\times10}$, $x_{i,j} = 1$ indicates that task T_i is assigned to processor P_j. To apply the matrix to the swarm intelligence algorithm, the compressed scheduling matrix is $\{2, 4, 0, 3, 1, 0, 3, 2, 1, 3\}$, the compressed scheduling matrix in the programming application becomes an index, and the task serial number is mapped on the corresponding processor according to the order of the compressed scheduling matrix.

2.5. Energy Consumption Model. With the development of processor hardware technology, the CMOS integrated circuit has become the digital circuit of widely used processors. The power consumption of the CMOS integrated circuit mainly consists of dynamic power consumption and internal short-circuit power consumption. As the internal short-circuit power consumption of an ordinary processor is less than 10% of the dynamic power consumption, the internal short-circuit power consumption of a processor is not considered when utilizing a DVS technology-based processor as a research platform to study the scheduling problem and reduce energy consumption. The dynamic power consumption equation [11] of the processor that adopts DVS technology is

$$P_d = ACV^2 f, \tag{1}$$

where A is the circuit flip frequency, C is the load capacitance, V is the supply power voltage, and f is the clock rate. In digital circuits, voltage V and clock rate f are positively correlated; the formula indicates that supply power voltage is a determining factor of power consumption [11]. As for a given processor, $A \cdot C$ is generally considered a constant, and different supply voltages correspond to different levels of power consumption. When the supply voltage of a computing resource is reduced, the calculating speed and power of the computing resource decrease accordingly.

In the present study, the execution time of a single task is $e_{i,j} = w_i/f_j$, and the scheduling length (makespan) in a single processor is

$$m_j = \sum_{k=1}^{p} \left(e_{k,j} + d_{k,\text{next}}\right), \tag{2}$$

where p is the number of tasks assigned to processor P_j, $e_{k,j}$ represents the execution time assigned to processor P_j, and $d_{k,\text{next}}$ is the communication time between the current task and the next task. Thus, the total scheduling length of a task graph is

$$M = \max\left\{m_1, m_2, \ldots, m_j, \ldots, m_m\right\}. \tag{3}$$

The total execution power consumption of the scheduling system can be expressed with the following formula:

$$E = \sum_{i=1}^{n} ACV_i^2 f \cdot w_i^* = \sum_{i=1}^{n} ACV_j^2 f_j \cdot \frac{w_i}{f_j} = \sum_{i=1}^{n} V_j^2 w_i, \quad (4)$$

where w_i is the corresponding computation amount of each task and j is the symbol indicating that task T_i is assigned to processor P_j.

Scheduling success rate is the number of successful scheduling tasks completed within the deadline divided by the total number of tasks. This index is used to evaluate the performance of the real-time task scheduling.

3. Ant Colony Scheduling Algorithm

The ant colony algorithm is based on the foraging behavior of ants, especially their means of finding the shortest paths between food sources and their nest. While moving food to their nest, ants leave a pheromone trail on the ground. Ants can smell pheromones. When choosing the path to search for food, they tend to choose the path with the highest concentration of pheromones. Pheromones have a positive feedback effect. When a path is taken by an increasing number of ants, the corresponding pheromone concentration increases. This condition entices the other ants to take such path. Ants follow this pheromone trail to find the shortest path between the food source and their nest.

On the basis of this observation, Dorigo et al. first presented ant colony algorithms to resolve difficult combinatorial optimization problems. The present work applies this algorithm to the energy consumption optimization problem.

In the actual simulation and application process, the algorithm design should consider the definition of ants, pheromone trail representation, design of pheromone deposition and volatilization, and relevant design of pheromone and path selection.

Each ant is defined according to the definition in the compressed scheduling matrix and effect matrix, and each scheduling matrix represents a walking path of ants. However, because pheromones are gradually released, the compressed scheduling matrix cannot be randomly generated and should instead be constantly selected and allocated by ants. According to the effectiveness of the path, the status of ants can be divided into three categories: partial solution, feasible solution, and infeasible solution.

Several methods can be used to represent a pheromone trail. In the present work, the pheromone trail is represented by the support rate $\sigma(i, j)$, which indicates the support rate of task T_i assigned to processor P_j in the ant colony. In the scheduling matrix, the ants move from the upper left corner to the lower right corner; $x_{i,j} = 1$ indicates that the ants pass through the path. The movement of the ants follows the constraints.

3.1. Design of Ant Pheromone Deposition and Volatilization Rate. The design of a scheduling algorithm should comply with three objectives, the methods for which should be exactly the same as those used for the calculation of the genetic algorithm but are referred to as optimal solutions in separate representations. First, the maximum information concentration of the ant colony algorithm is defined as

$$\delta_{\max} = \frac{f\left(s^{\text{best}}\right)}{\rho}, \quad (5)$$

where s^{best} is the optimal solution of the current calculation, ρ is the volatilization rate of the pheromones, and $f(s)$ is the corresponding evaluation function of each solution, the definition of which corresponds to the three objectives of this research.

The minimum information concentration is defined as

$$\delta_{\max} = \frac{\delta_{\min}}{(\omega \cdot \ln(\theta + 1))}, \quad (6)$$

where ω is a constant that is greater than or equal to 1 and θ is the number of iterations of the ant colony algorithm (starting from 1).

After the assignment of task T_i to processor P_j, the pheromone trail is updated according to the relationship between (T_i, P_j) and s^{best}. The updating formula is

$$\begin{aligned} \sigma(i, j) &= (1 - \rho) \cdot \sigma(i, j) + f\left(s^{\text{best}}\right), \quad \left(T_i, P_j\right) \in s^{\text{best}}, \\ \sigma(i, j) &= (1 - \rho) \cdot \sigma(i, j), \quad \left(T_i, P_j\right) \notin s^{\text{best}}. \end{aligned} \quad (7)$$

If $\sigma(i, j) > \delta_{\max}$, then $\sigma(i, j) = \delta_{\max}$; if $\sigma(i, j) < \delta_{\min}$, then $\sigma(i, j) = \delta_{\min}$ to maintain the information concentration within a certain range.

3.2. Ant Colony Algorithm Design. The purpose of programming a specific ant colony algorithm is to find feasible solutions for the three objectives of this work and to output corresponding targets to find the optimal path for ants. The ant colony algorithm can be terminated under two conditions: (1) a feasible solution is found and (2) the algorithm execution reaches the maximum number of iterations.

The specific process flowchart for designing the ant colony algorithm is shown in Figure 2.

4. Bee Colony Scheduling Algorithm

The bee colony algorithm is an emerging swarm intelligence algorithm that has been widely used in recent years. It explores the optimal value according to the role assignment and comprises conversion and information exchange methods that bee colonies adopt to search for food. The application methods of the bee colony algorithm and ant colony algorithm are almost the same. However, the bee colony algorithm adopts its own unique calculation formula to find the adaptation value and selection probability of food sources. The following section of this paper introduces the role of ants in the bee colony algorithm according to the implementation process of the bee colony algorithm.

The algorithmic frameworks of the bee colony algorithm include the initial period, employed bee period, and scout bee period. In the last two periods of a cycle, employed

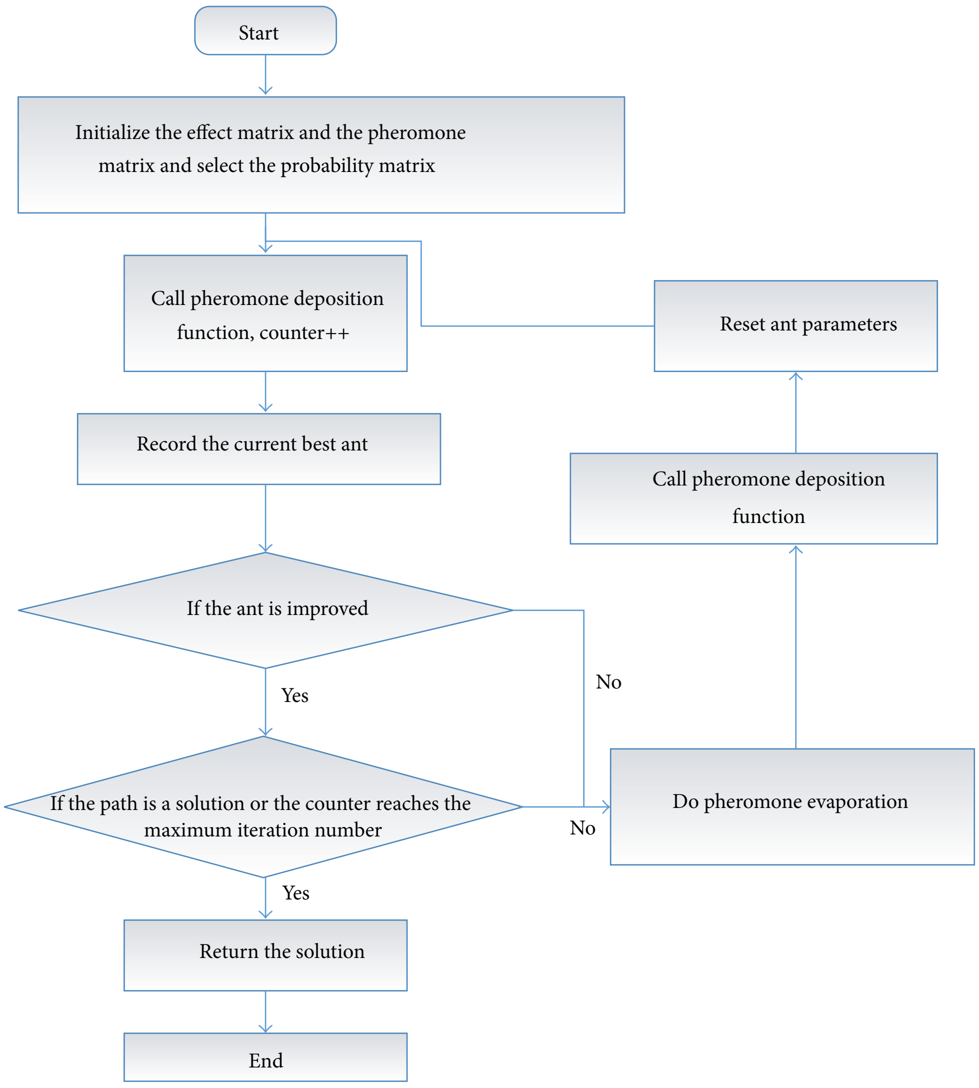
Start
Initialize the effect matrix and the pheromone matrix and select the probability matrix
Call pheromone deposition function, counter++
Record the current best ant
If the ant is improved
Yes
No
If the path is a solution or the counter reaches the maximum iteration number
Yes
No
Return the solution
End
Reset ant parameters
Call pheromone deposition function
Do pheromone evaporation

FIGURE 2: Flowchart for designing the ant colony algorithm.

TABLE 2: Experimental parameters of the genetic algorithm.

Parameter serial number	Parameter name	Parameter value
1	Number of chromosomes	200
2	Crossover rate	0.8
3	Mutation rate	0.15
4	Maximum number of iterations	1,000

TABLE 3: Experimental parameters of the ant colony algorithm.

Parameter serial number	Parameter name	Parameter value
1	Number of ants	10
2	ρ	0.02
3	ω	20.00
4	Maximum number of iterations	1,000

bees become onlooker bees according to the food sources found until they find the optimal solution or the maximum number of cycles. In the bee colony algorithm, the bees are divided into three categories: employed bees, onlooker bees, and scout bees. Employed bees are related to a particular food source, onlooker bees observe the dance information of employed bees to select the food source, and scout bees randomly search for food. In the initial stage, the scout bees find all the food sources. Then, the employed bees and onlooker bees make use of the food sources. The employed bees that deplete the food sources become scout bees.

In the scheduling matrix, when a bee is a scout bee, it randomly searches the path from left to right of the matrix and selects one point in each column until $x_{i,j} = 1$, which indicates that the task has been assigned. The scout bee then becomes an employed bee. When a bee is transformed into an onlooker bee, it selects employed bees to follow its

Table 4: Experimental parameters of the bee colony algorithm.

Parameter serial number	Parameter name	Parameter value
1	Number of bees	10 (twice the number of food sources)
2	Maximum number of iterations	1,000

Table 5: Performance test data.

Task quantity	Genetic algorithm		BCO ant colony	ABC bee colony
10	Scheduling success rate	92.33%	Increase by 9.54%	Increase by 10.22%
	Scheduling length	13.125	Decrease by 8.23%	Decrease by 6.14%
	Energy consumption	64.024	Decrease by 5.24%	Increase by 5.22%
30	Scheduling success rate	90.35%	Increase by 8.43%	Decrease by 13.65%
	Scheduling length	63.254	Decrease by 7.49%	Increase by 19.55%
	Energy consumption	322.112	Decrease by 4.32%	Decrease by 5.32%
50	Scheduling success rate	91.67%	Increase by 11.24%	Decrease by 3.25%
	Scheduling length	150.763	Decrease by 11.23%	Decrease by 13.22%
	Energy consumption	579.245	Decrease by 10.25%	Increase by 6.88%

path according to the selection probability. When a bee is transformed into an employed bee, it repeats the path in the previous round (i.e., a compressed scheduling matrix).

4.1. Fitness Value and Selection Probability of the Bee Colony Algorithm. The value of the objective function determines the method for calculating the fitness value of the employed bee period. The definition of the bee colony algorithm takes the positive and negative values of the objective function. However, the values of the objective function proposed in this work are positive. Thus, the fitness function is calculated as

$$\text{fit}(m) = \frac{1}{(1 + f(m))}, \tag{8}$$

where m is a bee and $f(m)$ is the target path value of the bee.

In the scout bee period, employed bees are chosen to follow the path. The selection probability ρ_m of each employed bee is calculated as

$$\rho_m = \frac{\text{fit}(m)}{\sum_{i=1}^{\text{SN}} \text{fit}(i)}, \tag{9}$$

where SN is the number of employed bees.

4.2. Bee Colony Algorithm Design. The bee colony algorithm program has two termination conditions: (1) a feasible solution is found and (2) the maximum cycle index is reached. During the cycle, onlooker bees choose to follow employed bees according to the selection probability value, which is calculated according to the path of the employed bees. The flowchart is shown in Figure 3.

5. Experiment

5.1. Experiment Test Program Design. As for the three algorithms implemented in the heterogeneous platform, the unity of scheduling tasks and the differences in the parameter setting of each algorithm should be considered, and each algorithm should select the optimal parameters to achieve the best performance. The optimal state of each algorithm should then be used to compare the scheduling efficiencies of different algorithms.

The configuration of the experiment platform used in this work is as follows:

(1) operating system: Windows 7 Ultimate,

(2) CPU: Intel® Core™ i3 2.27 GHz,

(3) memory: 2.00 GB.

The three tests of the algorithm on the heterogeneous platform are mainly performance tests. Small-scale tasks are adopted in each performance test, and the main reference indicators are the algorithm's scheduling success rate, scheduling length, and energy consumption. Each test is repeated 10 times. The test results are the average values of each test.

The parameters of the three algorithms on the heterogeneous platform are shown in Table 2 (genetic algorithm), Table 3 (ant colony algorithm), and Table 4 (bee colony algorithm), separately.

5.2. Experiment Results. The genetic algorithm is taken as a standard to compare the performance of the ant colony algorithm and bee colony algorithm. The test results are shown in Table 5, in which the values are the average values of 10 experiments.

As shown in the table, the ant colony algorithm is superior to the genetic algorithm in terms of achieving the three objectives. As for the bee colony algorithm, the scheduling results indicate its potential use for the combinatorial optimization of task scheduling; however, this algorithm needs to be improved further, or the program needs to be optimized to obtain convincing results. The results shown in the table cannot verify the advantages and disadvantages of the bee colony algorithm in task scheduling.

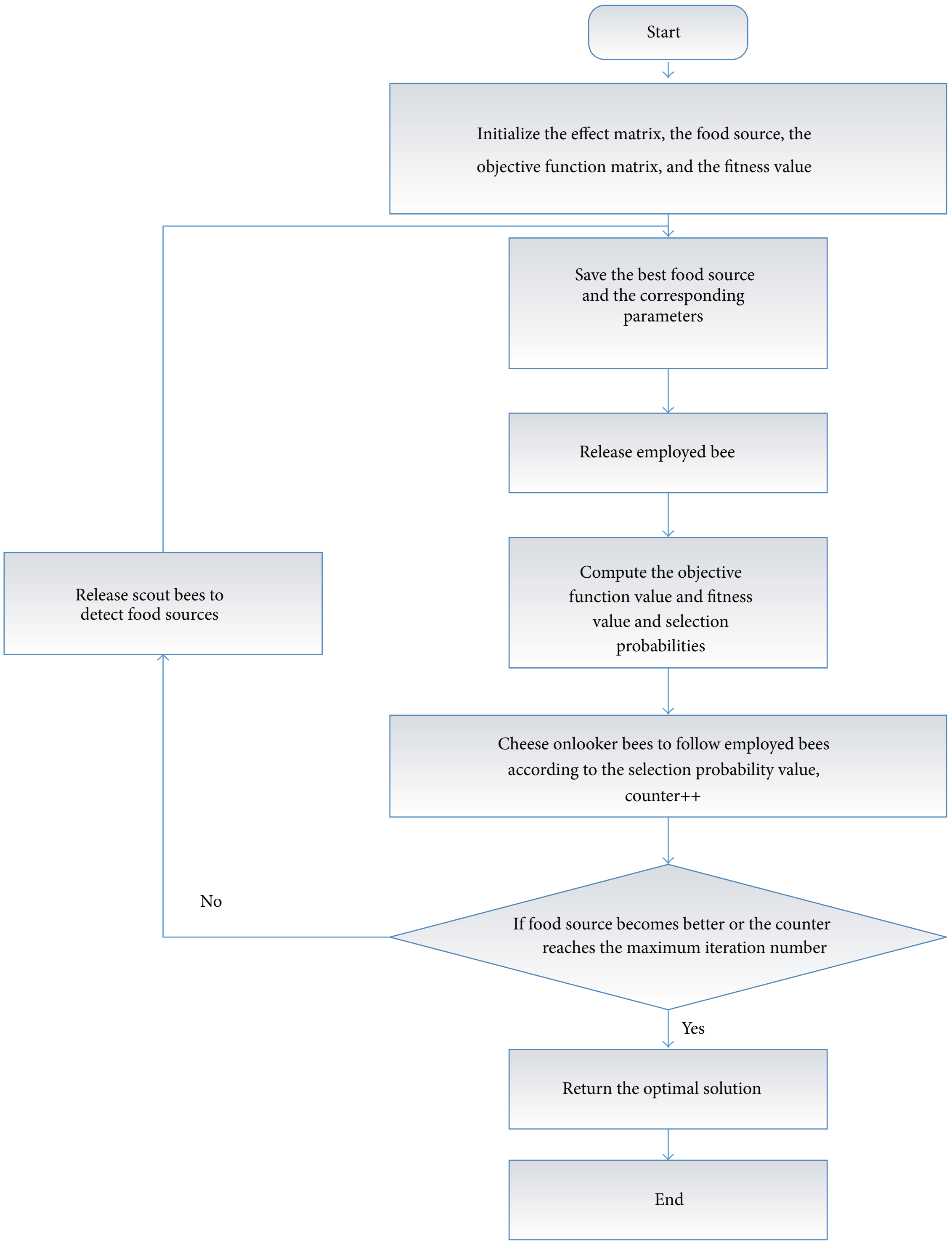
Start
Initialize the effect matrix, the food source, the objective function matrix, and the fitness value
Save the best food source and the corresponding parameters
Release employed bee
Compute the objective function value and fitness value and selection probabilities
Cheese onlooker bees to follow employed bees according to the selection probability value, counter++
If food source becomes better or the counter reaches the maximum iteration number
No
Release scout bees to detect food sources
Yes
Return the optimal solution
End

Figure 3: Flowchart of the bee colony algorithm program.

6. Conclusion

This work takes a real-time parallel task based on the DAG model as a scheduling object, adds the energy consumption model for a heterogeneous platform, pursues the goal of reducing energy consumption, and designs the ant colony algorithm and bee colony algorithm for a heterogeneous platform. Through a series of experiments, this work verifies the combinatorial optimization ability of the genetic algorithm in optimizing task scheduling and proves that the ant colony algorithm is superior to the genetic algorithm in terms of combinatorial optimization. The bee colony algorithm can potentially be used for combinatorial optimization.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61173145 and by the Doctoral Program of Higher Education of China (RFDP) under Grant no. 20132302110037.

References

[1] V. Kianzad and S. S. Bhattacharyya, "Efficient techniques for clustering and scheduling onto embedded multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 667–680, 2006.

[2] K. Rzadca and F. Seredynski, "Heterogeneous multiprocessor scheduling with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2840–2847, IEEE, September 2005.

[3] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 160–171, 2003.

[4] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[5] P. Ranganathan and P. Leech, "Simulating complex enterprise workloads using utilization traces," in *Proceedings of the 10th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW '07)*, February 2007.

[6] L. Wang, G. Von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '10)*, pp. 368–377, Melbourne, Australia, May 2010.

[7] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 346–360, 2009.

[8] J. Kolodziej, S. U. Khan, and F. Xhafa, "Genetic algorithms for energy-aware scheduling in computational grids," in *Proceedings of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC '11)*, pp. 17–24, IEEE, Barcelona, Spain, October 2011.

[9] Y. Zhu, J.-Z. Luo, and W. Li, "An approach for energy aware multipath service composition based on workflow," *Jisuanji Xuebao*, vol. 35, no. 3, pp. 627–638, 2012.

[10] Y.-K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 5, pp. 506–521, 1996.

[11] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, pp. 92–99, IEEE, Shanghai, China, May 2009.

[12] W. Zhang, E. Bai, H. He, and A. M. Cheng, "Energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms," *Sensors*, vol. 15, no. 6, pp. 13778–13804, 2015.

[13] W. Zhang, H. Xie, B. Cao, and A. M. K. Cheng, "Energy-aware real-time task scheduling for heterogeneous multiprocessors with particle swarm optimization algorithm," *Mathematical Problems in Engineering*, vol. 2014, Article ID 287475, 9 pages, 2014.

[14] N. Li, N. Zhang, S. K. Das, and B. Thuraisingham, "Privacy preservation in wireless sensor networks: a state-of-the-art survey," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1501–1514, 2009.

[15] M. M. Groat, W. Hey, and S. Forrest, "KIPDA: k-indistinguishable privacy-preserving data aggregation in wireless sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '11)*, pp. 2024–2032, Shanghai, China, April 2011.

[16] S. Ozdemir, M. Peng, and Y. Xiao, "PRDA: polynomial regression-based privacy-preserving data aggregation for wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 15, no. 4, pp. 615–628, 2015.

[17] H. Chen, A. M. K. Cheng, and Y.-W. Kuo, "Assigning real-time tasks to heterogeneous processors by applying ant colony optimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 1, pp. 132–142, 2011.