Energy-and-Time-Saving Task Scheduling Based on Improved Genetic Algorithm in Mobile Cloud Computing

Jirui Li(✉), Xiaoyong Li, and Rui Zhang

Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China
ljrokyes@163.com

Abstract. With the collaboration of 5G network and mobile cloud computing(MCC), mobile devices can be offered important opportunities and new challenges in terms of energy saving and performance enhancement, sophisticated applications running on smart phones, which are called tasks in MCC environment, may be same or different. The paper studies the problem of task scheduling in MCC. Firstly, a task-virtual machine (VM) assignment strategy is presented; Secondly, on the basis of the strategy, we improve genetic algorithm (GA) which uses grouping multi-level encoding and dual fitness function (GMLE-DFF), GMLE means that the individual adopts hierarchical coding according to VMs grouping and tasks queuing. DFF refers to the reasonable combination of the optimal time span and the maximum resources utilization and minimum opened number of VMs. By simulating and realizing traditional GA, Sufferage algorithm and our improved GA, the results show the improved GA is superior to other two algorithms for reducing energy consumption while the task completion time is satisfied.

Keywords: Mobile cloud computing · 5G · Task scheduling · Genetic algorithm · Energy consumption

1 Introduction

Nowadays, the combination of 5G network and cloud computing will give rise to newer and hotter attention, and can give more additional advantages, in particular, longer lasting battery lifetime and more powerful computing ability for mobile devices. These merits have brought much changes of services and applications, significantly faster growth number and richer forms, both of which require offloading heavy computing from mobile terminals to cloud in order to realize longer lasting services [1]. But, if a cloud is used to provide services, in cloud server, a heavy energy consumption problem must be paid special attention. With the rapid growth of richer applications running on mobile devices, the energy consumption problem is worse and worse. The solutions of software and hardware are urgently needed to determine which methods will be greener or more energy efficient.

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 418–428, 2017.
DOI: 10.1007/978-3-319-59288-6_38

In MCC, the study of task scheduling is very important for reducing energy consumption and improving the system performance, and scheduling efficiency can largely affect the execution of applications running on mobile devices, especially in 5G network. The methods of task scheduling have been proposed in many studies. Two non-cooperative game methods (symmetric non-zero-sum game and asymmetry stein's game) were proposed by Kolodziej and Xhafa [2] by modeling the user requirements as the grid user's behavior, and designed and realized hybrid tuning scheduler based GA to balance the two game. Li Zhi-Yong et al. [3] implemented a multi-objective memetic algorithm for the problem of multi-objective scheduling optimization on the heterogeneous cloud, which used memetic local search technique based on the related information of solution structure to improve the local optimization ability, the technique could reduce the computational overhead of algorithm. Li et al. [4] presented a resource optimization mechanism, which included two online dynamic task scheduling algorithms: dynamic cloud list scheduling and dynamic cloud min-min scheduling. Heuristic algorithm [5] has two main goals, minimizing the task execution time and minimizing the task execution cost. Li et al. [6] designed a kind of Normal Best-Oriented Ant Colony Optimization by applying improved heuristic algorithm. Taheri et al. [7] proposed a job data scheduling algorithm based on colony. Xue Lin et al. [8] presented a task scheduling algorithm with dynamic voltage and frequency scaling (DVFC), which mainly solves the energy minimization problem in MCC environment. Neeraj Kumar et al. [9] proposed a Bayesian coalition Game-based VM migration method in vehicular mobile cloud to avoid wasting energy. In the aforementioned methods, energy consumption was the product of the finished time of tasks times the power consumption per unit, the computation method is partial and cannot objectively reflect the whole energy consumption of cloud server center, which is usually to maximize the resources utilization and minimize opened number of VMs, meanwhile, minimize the execution time of task sets [10].

Because of the mobility of terminals, tasks must be finished in the limited time, or else, the execution result may not return the terminals if the link between mobile device and cloud server is lost, and it must product large and additional energy consumption in cloud computing. In this paper, basing on the "intensive" characteristic of 5G, a task-VM allocation strategy is presented and applied to GMLE-DFF-based GA for improving the applications performance and the resource utilization of cloud server.

2 Task Scheduling Based on Improved GA

In 1975, J.H. Holland professor put forward the basic theory and method of GA [11], which is based on nature selection and genetic theory and adopts the principle of "survival of the fittest" in nature. GA contains chromosome encoding, initial population, the design of fitness function, genetic operation (selection, crossover and mutation) and parameters settings. At present, GA has been widely used in artificial intelligence, information technology, computer science and engineering practice. With the in-depth theory study and practice proving, although GA develops rapidly, its shortcomings are gradually revealed. GA slows for the selection of the optimal solution and is easy to form a pseudo solution. To improve GA, the key is to avoid a false solution and shorten

the time for finding the optimal solution. In this paper, combining with resource characteristic in MCC system, an improved GA based on GMLE-DFF is proposed, which applies task-VM allocation strategy orienting multidimensional resources, and its flow figure is shown in Fig. 1.

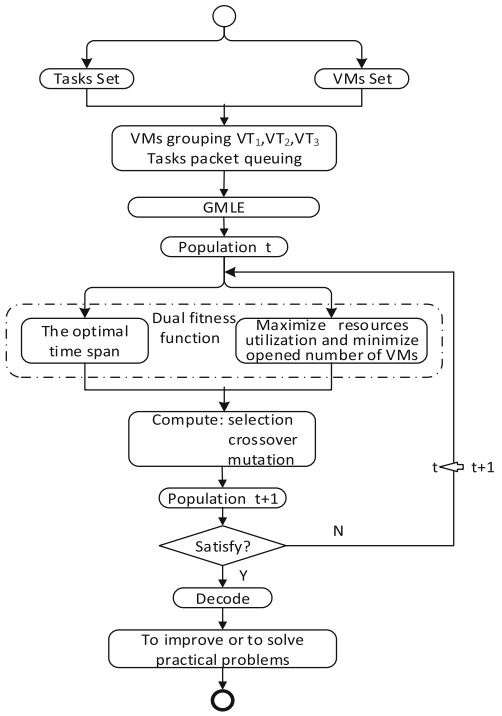

Fig. 1. The flow figure of improved GA based on GMLE-DFF.

2.1 Task-VM Assignment Strategy

Improved GA uses packet queuing method for tasks and VMs before encoding. In MCC environment, any one of mobile users may trigger tasks, and the distribution of tasks may be asymmetry. If considering only performance indicators of tasks, the uneven distribution of the tasks will happen, how to get good compromise between the task performance indexes and system load balancing is always a key point for task scheduling strategy. In order to solve this problem, we model task allocation process and present a two-stage task scheduling method, that is to say, task scheduling can be divided into VMs grouping and tasks allocation. The first stage, according to the computing power each VM provides, we divide VM collection into three groups, which are fixed sequence respectively: interaction intensive type (vt_1), computation intensive type (vt_2), and other type (vt_3). The second phase, for tasks waiting to be allocated, according to the demand for resources, each task selects the group and stands in line. Task-VM allocation strategy is shown in Fig. 2.

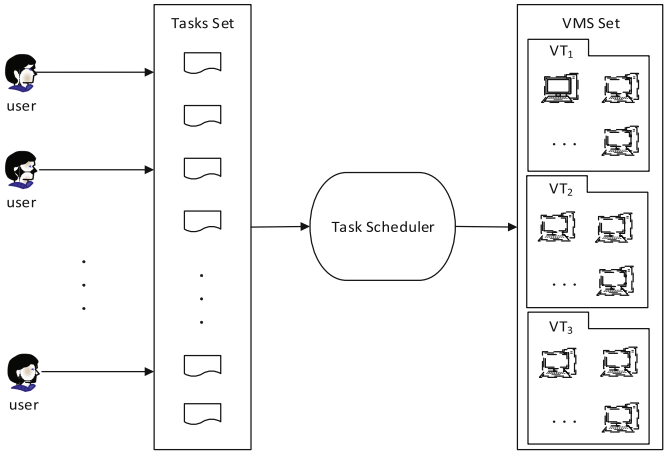

Fig. 2. Task-VM allocation strategy.

At Fig. 2, after tasks arrive at the request queue (Tasks set), in accordance with the principle of "First Come, First Service (FCFS)" [12], tasks are sent to the task scheduler, then, according to the running status of each VM, the scheduler calculates the corresponding VM set, if the current task is not to violate the Service Level Agreement (SLA) of VM, and deploys task on the VM, if the collection which does not violate the SLA is empty, then creates a new VM for the task.

2.2 Grouping Multi-level Encoding (GMLE) Mode

In MCC, mobile devices differ in thousands ways, different mobile devices may complete different tasks, so, it will trigger a greater difference of resources utilization between different VMs, even lead to load imbalance of cloud computing system. In order to solve the problem and promote the resources utilization, based on 2.1, we build a multi-level chromosome encoding rules, which use genetic box to signify VM and use gene to expresses task, one genetic box may include many genes, genes in box are no order, but the box capacity is limited, lots of genetic boxes make up a type of VM sets. These genetic boxes arrange in fixed order according to the category to form a chromosome. There are three genetic box sets, which are respectively vt_1,vt_2 and vt_3, tasks coming from mobile devices will be also classified following the same criterion. Table 1 illustrates the encoding rules.

At Table 1, it refers to the scheduling of m tasks allocated to n VMs. Firstly, n VMs are divided into three queues according to the special criterion; there is p, q, and $n - p - q$ VMs respectively in vt_1, vt_2, vt_3; then, on the basis of the same rules, the tasks are classified and fall in. The corresponding encoding is shown in formula (1).

$$\begin{aligned} D_i &= \{vt_1, vt_2, vt_3\} = \{\{v_1 \dots v_p\}, \{v_{p+1} \dots v_{p+q}\}, \{v_{p+q+1} \dots v_n\}\} \\ &= \{\{v_1{:}\{2, 5, 7, 38, 79\}, \dots, v_p{:}\{9, \dots, m\}\}, \{v_{p+1}{:}\{3, 52, m-2\}, \dots, v_{p+q}{:}\{4, \dots, 32\}\}, \\ &\quad \{v_{p+q+1}{:}\{m-3, m-1\}, \dots, v_n{:}\{14, \dots, m-10\}\}\}, \end{aligned} \tag{1}$$

Table 1. The Chromosome Coding.

Interaction intensive type vt_1	VM ID	v_1	v_2	…	v_p
	Task ID	{2,5,7,38,79}	{1,20,21,22}	…	{9,15,28,44,101,116, m}
Computation intensive type vt_2	VM ID	v_{p+1}	v_{p+2}	…	v_{p+q}
	Task ID	{3,52, m-2}	{18,27,49,63,70,92}	…	{4,8,66,87,32}
Other type vt_3	VM ID	v_{p+q+1}	v_{p+q+2}	…	v_n
	Task ID	{m-3,m-1}	{25,56,43,120,10}	…	{14,77,96,131,158,m-10}

After the chromosome encoding is determined, the population will be initialized. Because of the diversity and uncertainty of the initial population, the paper uses the random method product the initial population, which size is *S*.

2.3 Energy-and-Time-Aware Double Fitness Function

Improved GA makes use of DFF to reality the compromise between promoting the performance and reducing energy consumption. DFF refers to the optimal time span and the maximum resources utilization and minimum opened number of all VMs.

2.3.1 The Optimal Time Span

For a task-VM allocation scheme, its optimal time span means the completion time of a task which is the longest in all the tasks of VM. $cost_j^i$ expresses the completing time of the task t_i running on VM v_j. If the number of tasks allocated to v_j is *ats*, the finishing time of total task is shown in formula (2).

$$T_{j,total} = \sum_{i=1}^{ats} cost_j^i, \tag{2}$$

Therefore, the fitness function based on the optimal time span is formula (3).

$$f(j) = max_{1 \le k \le n} T_{k,total}, 1 \le j \le S, \tag{3}$$

In the initial evolution stage of the population, the part individuals which have superior fitness value may mislead the development direction of population; near the convergence phase, the individuals which fitness values are close are difficult to continue to evolve, thus, it is difficult to find the optimal solution. In the above two cases, the fitness function must be adjusted, after adjusting, $f_{ad}(j)$ is formula (4).

$$f_{ad}(j) = \frac{f(j) + |f_{min}|}{f_{max} + f_{min} + \delta}, 1 \le j \le S, \tag{4}$$

Of which, $f(j)$ is the original fitness value; f_{max} is an upper bound of $f(j)$; f_{min} is a lower bound of $f(j)$; δ is a real number,$\delta \in (0, 1)$, which aim is to prevent the

denominator as zero and increase the randomness of GA. If f_{max} or f_{min} is unknown, their value is replaced by the maximum or minimum value of the current generation so far, respectively; $|f_{min}|$ is to ensure that the fitness values are not negative after the calibration; if there is larger difference between f_{max} and f_{min}, the discrepancy is smaller after adjusting, this can prevent the exceptional individuals from ruling the whole population; if f_{max} and f_{min} are more close, the discrepancy is bigger after adjusting, thus this can increase the differences between individuals in the group and expand the search space to find the optimal solution.

2.3.2 Maximizing Resources Utilization and Minimizing Opened Number of VMs

Firstly, defining the fitness of the genetic box(VM). Consider the average utilization of all resources on VM as the comprehensive utilization(ϑ_c), as shown in formula (5).

$$\overline{\vartheta}_c = \frac{\sum \vartheta_k^c}{4}, \quad k \in \{mem, cpu, stor, net\}, \tag{5}$$

Among, ϑ_k^c denotes the utilization of the k-dimensional resource on VM v_c.

However, $\overline{\vartheta}_c$ cannot completely measure VM utilization, for example, v_c has two dimensions, the utilization of one is 85%, the other is 15%, then $\overline{\vartheta}_c = 50\%$; if the utilization of two dimensions on v_i is 45% and 55% respectively, then $\overline{\vartheta}_i = 50\%$. It is obvious that the resources usage of v_i is superior to v_c. Considering the balance between each resource dimension on VM, based on $\overline{\vartheta}_c$, designing a new genetic box evaluation parameter θ_c, which is expressed in formula (6).

$$\theta_c = \frac{\overline{\vartheta}_c}{\sqrt{\sum \left(\vartheta_k^c - \overline{\vartheta}_c\right)^2}}, \quad k \in \{mem, cpu, stor, net\}, \tag{6}$$

θ_c reflects the utilization of a genetic box and the equilibrium degree between the resources. θ_c is higher, corresponding VM has a higher efficiency, at the same time, the difference of the utilization between multiple resource dimensions on VM is small.

If using $usage_j$ to depict the overall utilization and resource equilibrium degree of the chromosome j (VM sets), as shown in formula (7).

$$usage_j = \frac{\sum \theta_c}{open\_num_z}, \tag{7}$$

Therein to, $open\_num_z$ signifies the number of opening VMs in the allocation scheme. $usage_j$ reflects an average utilization of all VMs which are deployed the tasks in the task migration plan corresponding with chromosome j.

Secondly, the fitness of chromosome j is made up of two parts: (1) overall utilization and resource equilibrium degree $usage_j$; (2) the number of opening VMs $open\_num_z$. Then, the fitness function of chromosome j is formula (8).

$$f(chr_j) = \varepsilon_1 * usage_j - \varepsilon_2 * \frac{open\_num_z}{n}, \tag{8}$$

n is the number of VMs in the whole system; ε_1 and ε_2 are the weight coefficient.

Finally, in order to facilitate screening of individuals in the phase of selection stage, before selection, using the normalization and information entropy methods [13] to handle DFF, the purpose is to control the production of population objectively.

2.4 Genetic Operation and the Stopping Criterion

In GA, the goal of selection operator is to simulate the evolution model in the nature, excellent individuals have greater provability to survive to the next generation and poor individuals are likely to be eliminated. this paper uses the selection method of roulette wheel to select individuals, the computation of the individual fitness is formula (9).

$$p_k = \frac{\varphi_{a_k}(i)}{\sum_{k=1}^{S} \varphi_{a_k}(i)}, \quad k = 1, 2, \ldots, S, \tag{9}$$

i shows the i generation chromosomes set, $\varphi_{a_k}(i)$ is the fitness value of individual which will variant.

The crossover operator is the main method to generate new individual, it determines the global search ability of GA and looks for the crossover pair from the whole population. The selection of the crossover position uses formula (10) to calculate.

$$p_c = \begin{cases} p_{c_1} * e^{\frac{\varphi'_{a_k}(i) - \varphi^{avg}_{a_k}(i)}{\varphi^{max}_{a_k}(i) - \varphi^{avg}_{a_k}(i)}}, & \varphi'_{a_k}(i) \geq \varphi^{avg}_{a_k}(i) \\ p_{c_1}, & \varphi'_{a_k}(i) < \varphi^{avg}_{a_k}(i) \end{cases} \tag{10}$$

$\varphi^{max}_{a_k}(i)$ is the maximum value of the population fitness, $\varphi^{avg}_{a_k}(i)$ is the average value of the population fitness, $\varphi'_{a_k}(i)$ is the one of two crossover individuals which fitness value is bigger, p_{c_1} is set as the maximum crossover probability.

The mutation operator is the aiding method to produce new individuals. The mutation operator is as follow.

$$p_m = \begin{cases} p_{m_1} * e^{\frac{\varphi_{a_k}(i) - \varphi^{avg}_{a_k}(i)}{\varphi^{max}_{a_k}(i) - \varphi^{avg}_{a_k}(i)}}, & \varphi_{a_k}(i) \geq \varphi^{avg}_{a_k}(i) \\ p_{m_1}, & \varphi_{a_k}(i) < \varphi^{avg}_{a_k}(i) \end{cases} \tag{11}$$

Among, p_{m_1} is set as the maximum mutation probability in the algorithm.

In this paper, using the standard deviation of the fitness function value of the optimal time span as the terminal convergence conditions.

$$tc = \sqrt{\frac{\sum_{k=1}^{S}(\varphi_{a_k}(i) - \varphi_{a_k}^{avg}(i))^2}{S}} < \sigma, \tag{12}$$

σ is the convergence threshold, σ ∈ (0, 1), if the standard deviation *tc* is smaller than σ, the iteration terminates, or to continue their genetic evolution.

3 Performance Evaluation

In the simulation experiment, the host is configured as follow, CPU is Intel 3.2 GHz, Memory is 12.0 GB, and the capacity of hardware is 500 GB. The parameters and their value are shown at Tables 2 and 3.

Table 2. The number settings of tasks and VMs.

Configuration groups	conf1	conf2	conf3	conf4	conf5	conf6
Number of VMs	20	60	100	200	300	500
Number of tasks	60	180	300	600	900	1500

Table 3. The parameters settings of simulation environment.

Parameters	Settings	Illustration
Population	44	Population size
ε_1	0.75	Weight coefficient of resources utilization
ε_2	0.25	Weight coefficient of the number of opening VMs
p_{c_1}	0.69	Maximum crossover probability
p_{m_1}	0.008	Maximum mutation probability
σ	0.1	Convergence threshold

At Table 2, the number of VMs in vt_1, vt_2, vt_3 is respectively gotten by summarizing the numbers of VMs after which are partitioned according to the VM parameters generated randomly; the number of tasks in each VM type vt_1, vt_2, vt_3 is also gained by using the same summarizing method.

3.1 Evaluate Experimental Results

3.1.1 Comparing Tasks Completion Time

Under the different number configurations of tasks and VMs, the completion time of tasks in the three algorithms are recorded at Table 4.

Table 4. Task completion time.

Algorithms	The completion time under the difference configurations(t)					
	conf1	conf2	conf3	conf4	conf5	conf6
GMLE-DFF	198	630	6465	40720	125608	300225
Sufferage	150	558	6278	40610	125608	300255
GA	180	594	6398	40640	125698	301650

Known from Table 4, task completion time of the three algorithms is nearly the same from conf1 to conf6 respectively, Improved GA is no obvious advantage. Based on Table 2, maintaining the number of tasks is constant at 4500, respectively recording task completion time, the result is in Fig. 3.

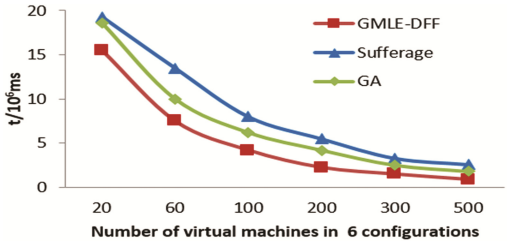

Fig. 3. Task completion time with the number of VMs changing.

Known from Fig. 3, when the number of VMs is less, the completion time of GMLE-DFF is far less than Sufferage and GA. But, as the number of VMs increasing, task completion time of all the three algorithms are declining and roughly equal finally. The cause is when the number of VMs increases, handling ability of VMs can well meet tasks demand, the conditions of task preemption resources and FCFS do not exist.

3.1.2 Comparing Resources Utilization

Under 6 configurations of Table 2, comparing resources utilization of improved GA and Sufferage algorithm, the average resources utilization is recorded in Fig. 4.

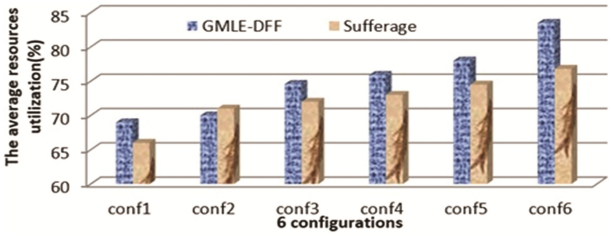

Fig. 4. Average resources utilization.

Seen from Fig. 4, when the number of tasks and VMs is both small, the difference of improved GA and Sufferage is not large in terms of average resources utilization, even in the conf2, Sufferage is better than improved GA. However, as the number of tasks and VMs increases gradually, the advantage of improved GA emerges little by little, for example, to conf6, the average resources utilization of improved GA is 83.5%, and Sufferage is 76.8%, the former is 6.7% higher than the latter. This is ascribed to redundant free VMs closed in improved GA based on GMLE-DFF.

3.1.3 Algorithm Convergence Situation

Supposing the number of tasks is 4500 in conf2, algorithm convergence result of improved GA based on GMLE-DFF and GA is shown in Fig. 5.

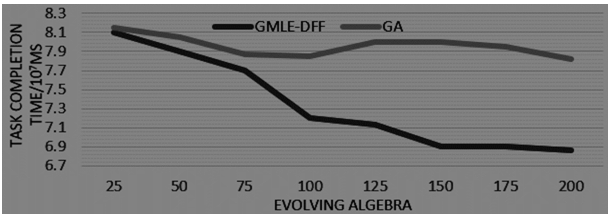

Fig. 5. Algorithm convergence result.

Seen from Fig. 5, at the early stage of algorithm iteration, the performance of two algorithms is more or less, but the convergence speed of improved GA based on GMLE-DFF is faster. After 150 generations, improved GA has started to restrain oneself, while GA presents the tendency of convergence when it is close to 200 generations. In addition, in the initial phase of evolution, it is about 65 generations, in GA, parts of exceptional individuals mislead the population evolution direction and make GA evolve to the direction of the extraordinary value from 65 to 100 generations, but the superior value is not the ideal optimal solution; Improved GA does not emerge the situation, meanwhile, improved GA adopts adaptive crossover probability and mutation probability, and adjusts the fitness function in the close to the algorithm convergence, therefore, finally the optimal solution gained by improved GA based on GMLE-DFF is more ideal than GA.

4 Conclusion

Reducing energy consumption is an important method to implement green computing. With the development of MCC and the coming of 5G, the related energy consumption problems give rise to new and hot attention for green MCC. Based on the intensive characteristic of 5G, the paper studies the problem of task scheduling in MCC. While guarantying system performance, in order to solute load imbalance and promote the resources utilization of the cloud computing system, a task-VM assignment strategy is put forward and applied to GA, which is improved based on GMLE-DFF. Experiments

show that convergence speed of improved GA based on GMLE-DFF is faster than GA, and its performance is more outstanding, not only the task execution time is the shortest, but also it could meet resources utilization maximizing and load balancing of VMs. Therefore, it can be well applied to task scheduling for reducing energy consumption and shortening the execution time in MCC.

Funding Acknowledgments. The work is supported by the National Nature Science Foundation of China (No. 61370069, 61672111), Fok Ying Tung Education Foundation (No. 132032), Beijing Natural Science Foundation (No. 4162043), and the Cosponsored Project of Beijing Committee of Education.

References

1. Liu, F., Shu, P., et al.: Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. Wirel. Commun. IEEE **20**(3), 14–22 (2013)
2. KołOdziej, J., Xhafa, F.: Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids. Int. J. Appl. Math. Comput. Sci. **21**(2), 243–257 (2011)
3. Li, Z.-Y., Chen, S.-M., Yang, B., et al.: Multi-objective memetic algorithm for task scheduling on heterogeneous cloud. Chin. J. Comput. **2016**(2)
4. Li, J., Qiu, M., Ming, Z., et al.: Online optimization for scheduling preemptable tasks on IaaS cloud systems. J. Parallel Distrib. Comput. **72**(5), 666–677 (2012)
5. Guo, L., Zhao, S., Shen, S., et al.: Task scheduling optimization in cloud computing based on heuristic algorithm. J. Netw. **7**(3), 547–553 (2012)
6. Li, J., Qiu, M., Niu, J., et al.: Feedback dynamic algorithms for preemptable job scheduling in cloud systems. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 1, pp. 561–564. IEEE (2010)
7. Taheri, J., Lee, Y.C., Zomaya, A.Y., et al.: A Bee Colony based optimization approach for simultaneous job scheduling and data replication in grid environments. Comput. Oper. Res. **40**(6), 1564–1578 (2013)
8. Lin, X., Wang, Y., Xie, Q., et al.: Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Trans. Serv. Comput. **8**(2), 175–186 (2015)
9. Kumar, N., et al.: Performance analysis of Bayesian coalition game-based energy-aware virtual machine migration in vehicular mobile cloud. Netw. IEEE **29**(2), 62–69 (2015)
10. Zomaya, A.Y., Teh, Y.H.: Observations on using genetic algorithms for dynamic load-balancing. IEEE Trans. Parallel Distrib. Syst. **12**(9), 899–911 (2001)
11. Thede, S.M.: An introduction to genetic algorithms. J. Comput. Sci. Coll. **20**(1), 115–123 (2004)
12. Prasad Acharya, G., Asha Rani, M.: Fault-tolerant multi-core system design using pb model and genetic algorithm based task scheduling. In: Satapathy, S.C., Rao, N.B., Kumar, S.S., Raj, C.D., Rao, V.M., Sarma, G.V.K. (eds.) Microelectronics, Electromagnetics and Telecommunications. LNEE, vol. 372, pp. 449–458. Springer, New Delhi (2016). doi: 10.1007/978-81-322-2728-1_41
13. Li, X., et al.: Service operator-aware trust scheme for resource matchmaking across multiple clouds. IEEE Trans. Parallel Distrib. Syst. **26**(5), 1419–1429 (2015)