Cost-Effective Scheduling Precedence Constrained Tasks in Cloud Computing

Bei Wang, Jun Li*, Chao Wang
Department of Automation
University of Science and Technology of China
Hefei, China
e-mail: Ljun@ustc.edu.cn

***Abstract*—Aiming at related task scheduling in cloud computing, a cost-effective precedence constrained tasks scheduling algorithm is presented. The algorithm takes into account the monetary cost and tries to fulfill a task scheduling balancing time and cost. In order to explore more possible solutions with high quality ignored by the deterministic algorithm, multi-population genetic algorithm is adopted to acquire the expected scheduling scheme. Besides, in order to improve the performance on time consuming of the algorithm, improved task duplication with monetary cost restriction is proposed. The validity of the algorithm is verified by experiments. Compared with deterministic scheduling algorithm, our algorithm lessens the cost greatly. Meanwhile, its performance on time consuming can also be guaranteed to some extent.**

Keywords-task scheduling; precedence constrained; cost-effective; cloud computing

I. INTRODUCTION

Strong computing capacity and convenient access have boosted employment of cloud computing in recent years. As the extension of grid computing [1], cloud computing hides underlying implements and provides consumers the services with an on-demand and pay-as-you-go way. Nowadays, many applications in cloud have emerged to be large graph processing, such as social networks and paralleling computing [2]. In general, such applications always consist of a great deal of tasks with precedence constraints and can be represented by directed acyclic graph (DAG). In order to increase the satisfaction of consumers, it's necessary to make full use of resources to get an efficient and reasonable DAG scheduling.

While the large graph application can be described as a directed acyclic graph, nodes in graph represent the tasks and edges denote the precedence constraints among tasks [3]. Before executing a task, its predecessors must complete execution and then deliver its execution results to the task [4]. Task scheduling is searching the mapping between tasks and processors while satisfying the precedence constraints among tasks. The scheduling problem in cloud has proven to be NP-Complete [5]-[7]. Concerning the issue, many researches have been made and various heuristics are developed. As one of the most classical and common heuristics, Heterogeneous Earliest Finish Time (HEFT) tried to reduce the makespan of the large graph by allocating the target task to the processor with the earliest finish time. The HEFT scheduling has short makespan and low complexity. However, without considering the inter-tasks communications, HEFT lost the opportunity to acquire a shorter scheduling scheme. Base on this, one duplication-based scheduling technique was proposed. It reduced the makespan of the whole graph by duplicating predecessors on the same processor with the target task to avoid inter-processors communications, which actually brought forward the execution start time of target task. Bozdag [8] improved the duplication-based algorithms to minimize the processor requirement with acceptable performance. To reduce finish time further, ZHANG Xiao-Lei [4] conducted the step of tasks duplication in advance and improved the priority calculation of tasks. It can be seen that, most DAG scheduling schemes were proposed to minimize tasks completion time but cost-oblivious. However, execution cost is also an important concern to consumer. Regarding to cost consuming, there are some cost-conscious scheduling strategies but making a compromise between monetary cost and time consuming. LI Jian [9] improved the Particle Swarm Optimization algorithm to gain balance between monetary cost and finish time, but without consideration on inter-tasks communications. Similarly, Convolbo [2] aimed to solve the cost optimization problem but was communications-oblivious. Moreover, there was trade-off between time and cost. Actually, an economic scheduling approach with high execution efficiency is what we target at.

In this paper, a cost-effective DAG scheduling scheme is proposed. Different with algorithms above, our strategy takes full account of the competition between time and cost consuming for a more economic scheduling, without sacrificing time performance. The scheme adopts Multi-Population Genetic Algorithm (MPGA) [10] to ensure global optimization and then imposes restrictions on task duplication to relieve the increase of monetary cost. In Section II, a description of DAG scheduling model in cloud computing is given. Details of the cost-effective DAG scheduling scheme are presented in Section III, followed by experimental results in Section IV. Finally, Section V concludes the paper.

II. DAG SCHEDULING MODEL

In directed acyclic graph, the tasks can be represented by the nodes and edges show the precedence constraints among tasks [11]. We define a DAG as $G = \{V, E, W, C\}$ where $V = \{v_i \mid i \in [1, n]\}$ is the set of n tasks and

$E=\{e_{ij} \mid v_i, v_j \in V\} \subseteq V \times V$ is the set of edges, denoting the dependencies among tasks; the $W=\{w_i \mid v_i \in V\}$ is the set of average time consuming of tasks; the $C=\{C(e_{ij}) \mid e_{ij} \in E\}$ is the set of communication cost among tasks and $C(e_{ij})$=0 when task v_i and v_j are executed on the same processor. For simplicity, the paper assumes that there are only one entry task v_{entry} which has no predecessor and one exit task v_{exit} without successor in standard DAG. Any DAG with more than one entry task or exit task can be converted to a standard one through the method in [4].

TABLE I. THE ETC OF A DAG TASK GRAPH

v_i	p_1	p_2	p_3
v_1	14	16	9
v_2	13	19	18
v_3	11	13	19
v_4	13	8	7
v_5	12	13	10
v_6	13	16	9
v_7	7	15	11
v_8	5	11	14
v_9	18	12	20
v_{10}	21	7	16

Assume there are *N* tasks and *M* processors and the task executions are non-preemptive. The execution time of tasks on different processors can be represented by matrix ETC [12]. The matrix ETC(i,j) denotes the execution time of task v_j on processor p_i. The calculation of ETC is as follow:

$$ETC(i,j)=\frac{MITask_j}{Capacity_i} \tag{1}$$

where $MITask_j$ is the length of task v_j and $Capacity_i$ refers to the computing capacity of processor p_i. A simple task graph example in [4] is shown as Fig. 1 with its corresponding ETC in Table I. For the processor p_k and task v_i, $ava(p_k)$ denotes the execution completion time of last task allocated to the processor, $st(v_i,p_k)$ and $ft(v_i,p_k)$ denote the execution start time and finish time of task v_i on processor p_k respectively. The execution start time of task v_i on processor p_k can be determined by (2).

$$st(v_i,p_k)=\max\{ava(p_k), \max_{v_j \in pred(v_i)}(ft(v_j,p_x)+C(e_{ji}))\} \tag{2}$$

where $pred(v_i)$ is the set of predecessors of v_i. Similarly, we use $succ(v_i)$ to denote the set of successors. Then the execution finish time goes as below.

$$ft(v_i,p_k)=st(v_i,p_k)+ETC(k,i) \tag{3}$$

Therefore, the scheduling length of the whole graph depends on the execution finish time of v_{exit}, which can be noted as

$$totalTime=\min_{j=1}^{M} ft(v_{exit},p_j) \tag{4}$$

If processor p_k is ready to execute task v_i before the arrival of the execution results from its predecessor, there will be idle-time on the processor which can be recorded as $slot(v_i,p_k)$ [4].

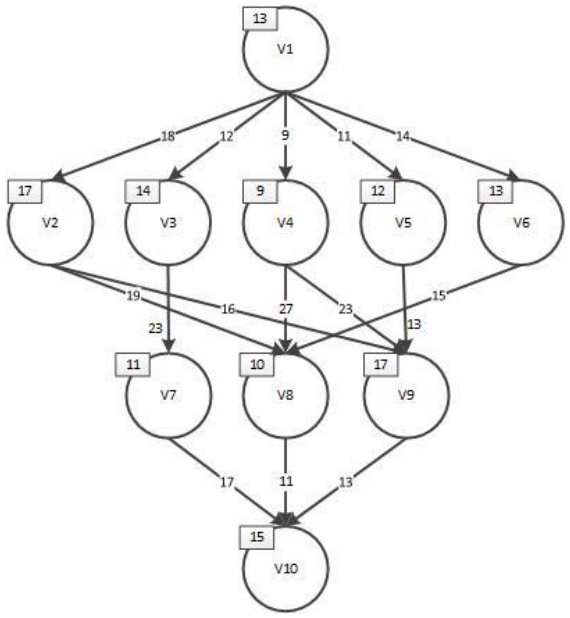

Figure 1. A simple task graph example

On the issue of price charged for processors in cloud, Google pricing model is adopted where charge for resources is based on the computing capacity and use time of resources [13]. The price charged for pi per unit time is shown as (5).

$$VM_{\cos t}(i)=VM_{base}\times \exp^{R_{base}} \tag{5}$$

where VM_{base} is the price charged for the slowest processor p_{base} and R_{base} denotes the speed ratio between them. The monetary cost for executing task v_j on processor p_i is

$$\cos t(i,j)=VM_{\cos t}(i)\times ETC(i,j) \tag{6}$$

The total cost for graph scheduling is

$$totalCost=\sum_j \cos t(i,j) \tag{7}$$

III. COST-EFFECTIVE DAG SCHEDULING SCHEME

Considering the likely falling into local optimum from genetic algorithm as well as particle swarm optimization, multi-population genetic algorithm is adopted to acquire the optimal scheduling scheme. As an improvement to standard genetic algorithm, MPGA takes into account the balance between global and local search ability and strikes the optimal solution by parallel search of multiple populations [14].

TABLE II. CORRESPONDENCE BETWEEN PROCESSORS AND TASKS

Processors	Corresponding tasks
1	1, 6, 7, 10
2	2, 4
3	3, 5, 8, 9

A. *Priority Definition*

Given the DAG, a list of tasks can be constructed according to priority of tasks in descending order. To satisfy the precedence constraints among tasks, the calculation of tasks priority in (8) is conducted by traversing the graph upwards.

$$SL(v_i) = \alpha \bullet OD(v_i) \times SL^{'}(v_i) + \beta \bullet B(v_i) \tag{8}$$

where $OD(v_i)$ is the number of predecessors of task v_i and $B(v_i)$ is the sum of communication cost between v_i and its successors, while α together with β represents the weight. The value of $SL'(v_i)$ is the priority definition in HEFT [15].

$$SL^{'}(v_i) = w(v_i) + \max_{v_j \in succ(v_i)} (C(e_{ij}) + SL^{'}(v_j)) \tag{9}$$

Then schedule tasks according to the order in priority list. According to the definition, the entry task v_{entry} has the highest priority and is scheduled firstly, while exit task v_{exit} is the last.

B. *Chromosome Encoding*

After constructing priority list, we number tasks in sequence so the number of task v_{enry} is 1. In the paper, a direct encoding scheme is adopted. The value of each gene in a chromosome denotes the corresponding processor ID assigned to a task. A chromosome represents a scheduling scheme [16]. For the example with 3 processors and 10 tasks above, one possible chromosome can be encoded as

{1, 2, 3, 2, 3, 1, 1, 3, 3, 1}

The correspondence between tasks and processors is illustrated in Table II.

C. *Initialization and Fitness Function*

For the multiple populations, random initialization method is adopted. In the example above, the initial value of each gene in chromosome is a random integer between 1 and 3.

Considering the influence on scheduling scheme from monetary cost, fitness of individuals can be calculated by (10).

$$F = \omega \bullet \frac{T_{\max} - T}{T_{\max} - T_{\min}} + (1-\omega) \bullet \frac{C_{\max} - C}{C_{\max} - C_{\min}} \tag{10}$$

where T is the scheduling length of the current scheme and its corresponding monetary cost is C. T_{max} and C_{max} are the maximum makespan and monetary cost in local population respectively, while T_{min} and C_{min} represent the minimum. The weight for time consuming is denoted as w and the corresponding one for cost is 1-w. The value of w can be adjusted according to the bias for time and cost from users.

D. *Genetic Operations*

Standard genetic algorithm includes selection, crossover and mutation operation [17], [18]. Besides, MPGA introduced immigration and elite reservation operation.

1) Selection operation

A deterministic sampling selection operator is used in the paper. Suppose there are *NIND* individuals in each population, then the expected number of the i^{th} individual existed in next generation can be given by (11). The value of *NIND* which is referred to the scale of population can be set in experiments.

$$N_i = NIND \bullet \frac{F_i}{\sum F_i} \tag{11}$$

where F_i is the fitness of the i^{th} individual. There are $\sum_{i=1}^{NIND} [N_i]$ individuals can be determined now. Then we sort individuals in descending order according to the fraction part of N_i, and add the first $NIND - \sum_{i=1}^{NIND} [N_i]$ individuals into the next generation. So far, *NIND* individuals have been determined.

2) Crossover and Mutation operation

In order to improve the search efficiency, adaptive crossover and mutation operation are adopted. The probability of crossover is set as follows:

$$P_c = \begin{cases} k1 * \dfrac{f_{\max} - f'}{f_{\max} - f_{avg}}, & f' \geq f_{avg} \\ k2 \qquad\qquad\quad , & f' < f_{avg} \end{cases} \tag{12}$$

where *f'* is the higher fitness of the two crossed chromosomes and f_{max} is the maximum fitness in the population, f_{avg} is the average value. The probability of mutation is set as:

$$P_m = \begin{cases} k3 * \dfrac{f_{\max} - f}{f_{\max} - f_{avg}}, & f \geq f_{avg} \\ k4 \qquad\qquad\quad , & f < f_{avg} \end{cases} \tag{13}$$

where f is the fitness of the mutated individual. As for the value of $k1$, $k2$, $k3$, $k4$, they can be identified in experiments.

3) Immigration and Elite Reservation Operations

The immigration operation in MPGA introduces the optimal individuals appearing in evolutionary process into other populations regularly. It realizes the information exchange among populations. At the stage of elite reservation, optimal individual of each population will be picked out to constitute the quintessence population.

E. Improved Task Duplication

After constructing a possible scheme, task duplication is considered. Duplicating the predecessors on the same processor with the target task during the slot time can reduce the communication time efficiently, so as to reduce makespan of the whole graph [19], [20]. Suppose the task v_i will be executed on processor p_k and its predecessor is task v_j. Without violating precedence constraints among tasks, task duplication will be considered if conditions in (14) are satisfied.

$$slot(v_i, p_k) \geq ETC(k, j) \,\&\&\, ft(v_j, p_k) < st(v_i, p_k) \tag{14}$$

For shorter makespan, copying more predecessors is considered if the task has more than one predecessor. Sorting its predecessors in descending order according to the arrival time of their execution results, then duplicating predecessor task in sequence until the conditions above cannot be satisfied. However, duplicating tasks blindly can lead to a great increase in cost. For a more economic task duplication, imposing cost restrictions on task duplication is considered. The target predecessor task will be duplicated only when (15) is satisfied:

$$-\frac{\Delta \cos t}{\Delta time} < k \bullet Maxprice \tag{15}$$

where $\Delta cost$ and $\Delta time$ are changes in monetary cost and makespan after task duplication respectively. *Maxprice* refers to the maximum price charged for processors and k is the coefficient. Through experiments, it is found that the strategy works well when $k = (\frac{\omega}{1-\omega})^2$.

IV. Results and Analysis

In order to verify the feasibility of our algorithm for large graph scheduling, comparisons with algorithm in [4] are given. We simulate the cloud environment in MATLAB. Related parameter settings are shown in Table III, where *MP* is the quantity of populations. The *Capacity* and *MITask* are generated randomly.

If no significant change for optimal individual's fitness in quintessence population happens for 10 generations, the iteration ends. Moreover, the iteration will be forced to terminate if it has been for 150 times.

TABLE III. Parameter Settings of the Experiment

Parameters	Value
MP	10
NIND	40
k1	0.6
k2	0.8
k3	0.1
k4	0.05
w	0.25
α	1
β	1
VM_{base}	0.1

For the example in Fig. 1, the prices charged for processors per unit time are generated randomly. The scheduling solution given by [4] can be illustrated in Fig. 2 with solution given by this paper in Fig. 3. In illustrations, the shaded tasks denote the task copy. Their scheduling lengths are both 66. The monetary cost are 42.357 and 33.5839 respectively. It is can be seen that with a same scheduling length, the scheme in Fig. 3 has less monetary cost obviously by avoiding possible uneconomic task duplication.

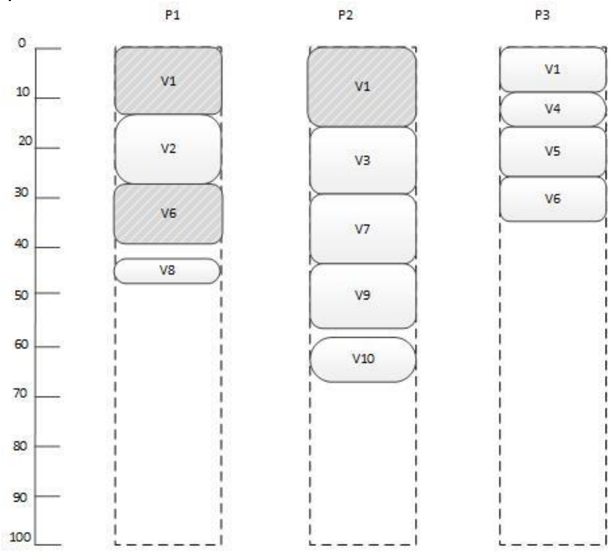

Figure 2. The scheduling illustration for comparison

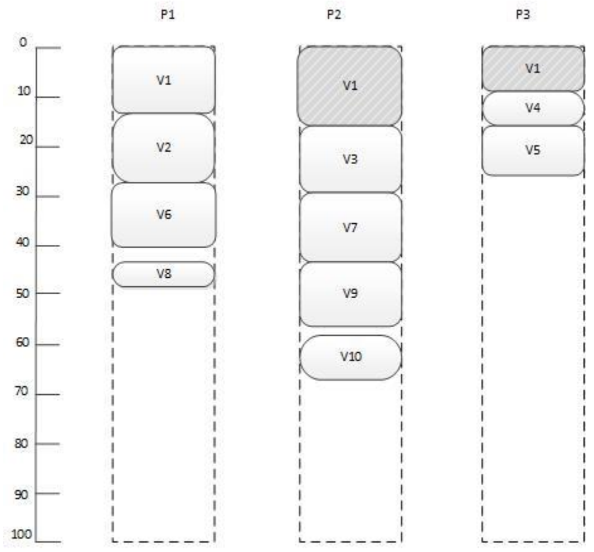

Figure 3. The scheduling illustration of our algorithm

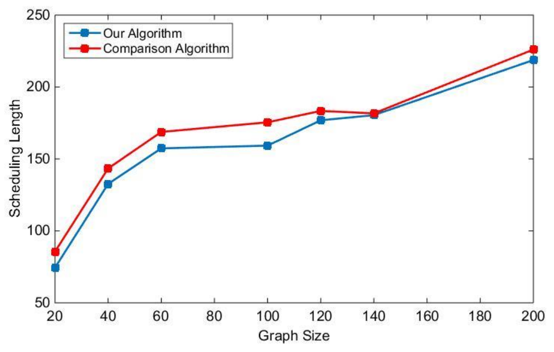

Figure 4. The scheduling length comparison

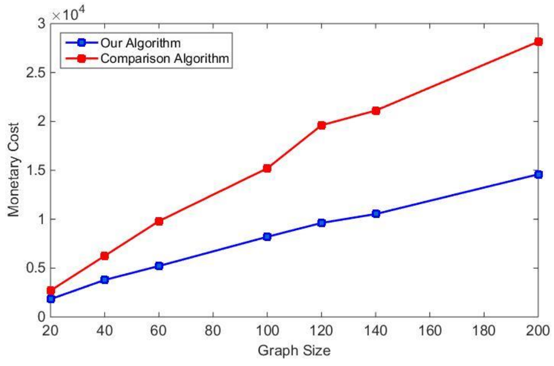

Figure 5. The monetary cost comparison

For a more general and objective evaluation, the following experiments are conducted. The simulated cloud environment is set with 200 heterogeneous virtual machines and the communication network is assumed to be fully connected [21]. The lengths of tasks and computing capacities of virtual machines are generated randomly [22], [23]. The performance comparisons between two are shown in Fig. 4 and Fig. 5.

The data in figures above are the average. Obviously, our algorithm outperforms the one in [4]. For determined algorithm like [4], its pattern to determine the solution is too absolute and only one solution can be constructed. Unlike [4], our algorithm searches for multiple solutions and are likely to get a better one. As the experiments reveal, our algorithm has excellent performance on cost consuming. This can be attributed to its cost consciousness and restriction on task duplication to avoid excessive increase in monetary cost. As for performance requirements, time and monetary cost are competitive and there can inevitably be some concessions in time for lower cost. That's the reason why the weakness on time performance exists sometimes. Besides, randomness of the initial population may also bring some instability to time performance. Even so, our algorithm performs well in general and its effectiveness and superiority to [4] are proved.

V. CONCLUSIONS

In this paper, a cost- effective DAG scheduling strategy is presented. The strategy expands the search scope and adopts multi-population genetic algorithm to acquire the optimal scheduling scheme. In order to utilize slot time in scheduling process efficiently, an improved cost-effective task duplication is considered. As a result, expected scheduling length and reasonable cost can be guaranteed. The simulation experiment results show that the cost-effective DAG scheduling algorithm has better performance than [4], especially in monetary cost. The algorithm reduces the monetary cost greatly with a proper scheduling length. Considering the contradiction between time and cost consuming, there can be some concessions in time. Therefore, win-win between time and cost consuming with more stable time performance is expected in further research. Besides, optimizing the algorithm for lower complexity and speeding up the search are also directions of future work.

ACKNOWLEDGMENT

The authors would like to thank the lab colleagues for their help in the study. Besides, the same thanks to anonymous reviewers for their valuable comments and suggestions on improving the quality of the paper.

REFERENCES

[1] N. Mohan and E. Raj, "Resource Allocation Techniques in Cloud Computing--Research Challenges for Application," in Proceedings of 4th International Conference on Computational Intelligence and Communication Networks, 2012: 556-560.

[2] M. W. Convolbo and J. Chou. "Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources," The Journal of Supercomputing, 2016, 72(3): 985-1012.

[3] K. L. Li, X. Y. Tang, B. Veeravalli and K. Q. Li. "Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems," IEEE Transactions on Computers, 2015, 64(1): 191-204.

[4] X. L. Zhang. Study on scheduling algorithm of the independent and associated tasks for cloud computing. Chongqing: College of Computer of Chongqing University, 2014.

[5] J. Li and J. Peng, "Task Scheduling algorithm based on improved genetic algorithm in cloud computing environment," Journal of Computer Application, 31(01): 184-186, 2011.

[6] ST. Maguluri, R. Srikant and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in 2012 Proceedings IEEE INFOCOM, 2012: 702-710.

[7] L. Zheng, The Research of resource scheduling key technology in cloud computing. Beijing: Beijing University of Posts and Telecommunication, 2014.

[8] D. Bozdag, F. Ozguner and U. V. Catalyurek. "Compaction of schedules and a two-stage approach for duplication-based DAG scheduling," IEEE Transactions on Parallel and Distributed Systems, 2009, 20(6): 857-871.

[9] J. Li, Q. J. Huang, Y. Y. Liu and S. Su. "Task scheduling for large graph processing based on particle swarm optimization in the cloud," Journal of Xi'an Jiaotong University, 2012,46(12): 116-121.

[10] T. Pencheva and M. Angelova, "Purposeful Model Parameters Genesis in Multi-population Genetic Algorithm," Global J Technol Optim, 5(1): 164, 2014.

[11] J. Mei, K. Li, A. Ouyang and K. Li. "A profit maximization scheme with guaranteed quality of service in cloud computing," IEEE Transactions on Computers, 2015, 64(11): 3064-3078.

[12] Z. Zhu and Z. Du, "Improved GA-based task scheduling algorithm in cloud computing," Computing Engineering and Applications, 49(5):77-80, 2013.

[13] J. Li, S. Su, X. Cheng, Q. J. Huang and Z. B. Zhang. "Cost-conscious scheduling for large graph processing in the cloud," //High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on. IEEE, 2011: 808-813.

[14] B. Wang and J. Li. "Load balancing task scheduling based on Multi-Population Genetic Algorithm in cloud computing," //Control Conference (CCC), 2016 35th Chinese. IEEE, 2016: 5261-5266.

[15] X. Chen, Y. C. Mao, Q. Jie and L. L. Zhu. "Related task scheduling algorithm based on task hierarchy and time constraint in cloud computing," Journal of Computer Applications, 2014, 34(11): 3069-3072.

[16] T. Wang, Z. Liu, and Y. Chen, "Load Balancing Task Scheduling Based on Genetic Algorithm in Cloud Computing," in Proceedings of 12th International Conference on Dependable, Autonomic and Secure Computing, 2014: 146-152.

[17] P. Babu and T. Amudha, "A novel genetic algorithm for effective job scheduling in grid environment," in Computational Intelligence, Cyber Security and Computational Models, M. Senthilkumar, V. Ramasamy, S. Sheen, C. Veeramani, A. Bonato, and L. Batten, Eds. New Delhi: Springer India, 2014: 385-393.

[18] L. Huang, A Research on Task Scheduling Algorithm of Cloud Computing Based on Genetic Algorithm. Xiamen: Xiamen University, 2014.

[19] L. Z. Guo, S G. Zhao, S. G. Shen and C. Y. Jiang. "Task scheduling optimization in cloud computing based on heuristic algorithm," Journal of Networks, 2012, 7(3): 547-553

[20] J. G. Barbosa, B. Moreira. "Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters," Parallel computing, 2011, 37(8): 428-438.

[21] R. K. Sharma, N. Sharma. "A Dynamic optimization algorithm for task scheduling in cloud computing with resource utilization," International Journal of Scientific Engineering and Technology, Volume, 2013 (2): 1062-1068.

[22] Q. Zhang, W. W. Niu, C. Z. Xing and H. Liang. "A scheduling algorithm of related tasks based on DAG graph in grid," Journal of Chinese Computer Systems, 2012, 33(5): 971-975

[23] C. Huang, D. Hu, and X. Yu, "Application of multiple-objective genetic algorithm for task scheduling in cloud environment," Information Technology, 38(5): 130-134, 2014.