Elitist task scheduling adaptive algorithm in cloud system based on genetic algorithm

Jingheng Ye

***Abstract*—In heterogeneous distributed computing systems like cloud computing, the problem of calculating the shortest time by mapping tasks to resources is a major issue which can have much impact on system performance. For some reasons such as heterogeneous and dynamic features and the dependencies among requests, task scheduling is known to be a NP-complete problem.**

In this paper, we proposed a feasible and efficient method (ETSAA) to find a suitable scheduling for workflow graph based on genetic algorithm.

At first, the elitist task scheduling adaptive algorithm (ETSAA) makes tasks prioritization in complex graph considering their impact on others, based on graph topology. This technique is efficient to reduction of completion time of application. Then, it merges FCFS and Best-Fit methods to make an optimal initial population in order to obtain a good solution quickly, and apply some suitable operations such as mutation and elitist selection to control and lead the algorithm to optimized solution. This algorithm evaluates the solutions by considering efficient parameters in cloud environment.

Finally, the proposed algorithm presents the better results with increasing number of tasks in application graph in contrast with FCFS greedy algorithm and traditional generic algorithm (GA).

***Index Terms*—Heterogeneous distributed computing systems; Cloud task scheduling; Adaptive generic algorithm; Elitist generic algorithm**

I. INTRODUCTION

Cloud computing is the further development of parallel algorithm, distributed computing and grid computing. It includes a collection of heterogeneous computing nodes, virtualized computers, and software services that are dynamically provisioned. These software services compete end-user's application by their availability, performance capability, and Quality of Service (QoS) requirements. The cloud computing provisions service in main three levels Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (laaS) [1]. In laas level, it takes Virtual Machines (VMs) as scheduling units for mapping physical heterogeneous resources to tasks. Each VM is an abstract unit which has computing and storage capacities in cloud. Because of heterogeneous and dynamic properties of resources, in addition to many numbers of entry tasks with different characteristics, this issue of task scheduling is known to be a NP-Complete problem. The software running in cloud environments is called an application. Each application is a workflow that consists of a set of tasks which are connected to each other. These tasks would be scheduled to run over different processors in cloud environments.

Many methods are proposed for this problem. Some methods have already been used in the scheduling of workflows [2, 3]. But they don't consider heterogeneous components characteristics and are not suitable for cloud computing which is heterogeneous distribute computing system. Some proposed algorithms take heterogeneous components characteristics into account [4], but they use simple-structure directed acyclic graph (DAG). The relationships between tasks of workflow in cloud environments are complex, so after workflow converts to DAG graph, the stratification is not obvious. For this reason, the complex structure DAG graph is more suitable for cloud environment. HSGA algorithm [5] takes heterogeneous of resources in cloud environment into consideration, moreover it uses complex structure graph to describe workflow, but HSGA does not introduce elitist strategy, crossover rate and mutate rate in HSGA is fixed, which is easy to cause premature phenomenon too early to find the global minimum. This paper mainly targets complex computation-intensive tasks, I improve HSGA algorithm to propose ETSAA algorithm. The ETSAA algorithm tries to decrease the number of algorithm operation iteration with tasks preprocessing to obtain an optimized initial population, considering the load balance on resources, then execute crossover and mutation operation according to the adaptive crossover rate and mutation rate to control premature phenomenon, thus lead algorithm to optimal solution.

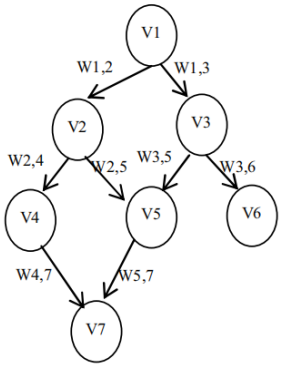
II. RELATED WORK

In this section, we mention simple-structure and complex structure of DAG graph. In this meantime, we introduce HSGA algorithm.

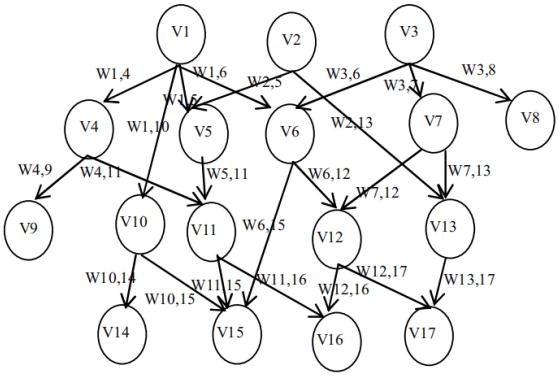
A. Simple structure and complex structure of DAG

graph

Each application in cloud environments is a workflow that contains a set of tasks, which tasks are connect to each other by precedence constraints. Each task will be executed and give an output data. This data will be then sent to the next task as defined by the structure of workflow. Most workflow can be represented by a DAG graph whose nodes represent tasks and whose edges represent precedence constraints. According to different workflow projects, the workflow application structures can be classified as either simple structure like Fig. 1(a) or complex structure as Fig. 1(b). In the simple-structure graph, nodes are related together considering certain level, but the complex-structure graph like [6] has more relation among nodes where the level of some nodes is not certain.



(a)A simple-structure graph



(b)A complex-structure graph

Fig. 1. The simple and complex graphs in workflow

B. HSGA algorithm

1) Encoding

In GA algorithm, every solution is encoded as a chromosome. There's a lot of ways to encode chromosome, how to encode that depends on the problem we will to resolve. Fig. 2 depicts a chromosome in HSGA algorithm. Here, first, the tasks of graph are ordered on priority according to section 2), then map tasks to processors.

2) Task prioritization

Tasks on complex application graph can't be partitioned into level easily, and each task's length and successors are different with others, so the task selection based on graph topology is an important problem. Each task's completion time influences the completion time of application, so executing order of tasks, in other words, an ordering method of tasks has important influence on total completion time. Thus HSGA algorithm computes the priority of tasks based on task influence, according to (1):

V2	V3	V1	V4	V6	V5
P1	P3	P1	P2	P3	P2

Fig. 2. A simple encoding of a schedule as a chromosome

$$T_{priority}(v_i) = w(v_i) + \sum_{\substack{d(v_j)=\alpha \\ v_j \in T_{pred}(v_j)}}^{\beta} \left(d(v_j) \times w(e_{i,j})\right) \tag{1}$$

Where task v_j is one of predecessors of task v_i , $d(v_j)$ is the depth of task v_j in critical path. The critical path for each task is the longest path from it to an exit task. $T_{pred}(v_j)$ is a set of predecessors of task v_i , β is the most depth of successors with the longest sequence and α is computed as:

$$\alpha = \beta - \lfloor \beta/2 \rfloor \tag{2}$$

3) Initial population

HSGA algorithm sorts tasks and puts that in all chromosomes' first line. Then each task is mapped to a selected resource from a virtual list of available resources, according to performance (computational speed, load, etc.) of virtual machines.

4) Evaluation of fitness function

The fitness value of each solution is computed through:

$$Fitness = \sum_{i=1}^{m} \left(p_i^t \times p_i^f\right) \tag{3}$$

Where p_i^t is the time when resource p_i completes all its assigned tasks of workflow in current scheduling, p_i^f is probability of task failure per unit time on resource p_i as failure rate of resource. The chromosome with minimum fitness value is considered as the best solution among the others.

Target is: minimizing Fitness (4)

5) Selection, crossover, mutation operation

Some of the best of chromosomes will be selected by elitism method for next iteration. And here a random gene selection crossover is used.

A chromosome and one of its genes will be chosen randomly and a resource will be selected randomly from virtual list of resources in mutation operation. The selected resource will be replaced with selected gene if its failure rate is better than the last candidate resource as [7] and this candidate resource is not the resource with most workload in current schedule. The failure factor of resource can be computed as:

$$p_i^{fr} = p_i^f / p_i^{MIPS} \tag{5}$$

Where p_i^{MIPS} is the computing power of resources as number of machine instructions per second.

III. THE IMPROVED ALGORITHM

This paper mainly focuses on task prioritization, select, crossover and mutation operation of ETSAA algorithm to execute corresponding improvements.

A. Task prioritization

Assume sorting tasks according to priority value of HSGA algorithm directly, it will have the following problems: if parent nodes of the child node is excessive or the edge of these parent nodes have a large value, that will result in a priority value of child node greater than parent nodes, child node perform prior to its parent nodes, thus is not in conformity with the topology sequence of DAG graph. Based on the above problems, this paper adopts the following sort method to calculate tasks' weight, as shown in (6):

$$\begin{aligned} T_{priority}(v_i) = w(v_i) &- \sum_{d(v_j)=\alpha}^{\beta} \left(d(v_j) \times w(v_j)\right) \\ &+ \sum_{d(v_k)=\alpha}^{\beta} \left(d(v_k) \times w(v_k)\right) \\ v_j \in T_{pred}(v_i), v_k &\in T_{succ}(v_i) \end{aligned} \tag{6}$$

Where $T_{succ}(v_i)$ is a set of successors of task v_i, $T_{pred}(v_i)$ is a set of predecessors of task v_i, $d(v_j)$ is the depth of v_j, $w(v_j)$ is the average completion time of v_j, representation of α is same as (2).

Here, the tasks are ordered by weight in descending then depth in descending. In this method, it will ensure the reasonableness of task execution, but also optimize the total completion time.

B. Encoding

In GA method, every solution is encoded as a chromosome. Each chromosome has N genes, as the chromosome length, as the same as the number of tasks to be finished. In workflow scheduling each schedule appears in a chromosome form. Each schedule contains the tasks of application and the related candidate resources.

C. Initial population

In this paper, a set of multiple possible solutions (chromosomes) is assumed to be referred to as a population. The initial population is made randomly in normal genetic algorithm. [9]

Making a good and goal oriented initial population that would lead to find the response in a rapid manner is the concern here. For this purpose, to build the initial population, after the tasks are sorted by priority, they will be placed in the first row of genes in the chromosome, and for each task, a suitable resource will be select with minimum running time for task from virtual resource list.

This process is repeated for all genes as Best-Fit. In this manner, in first chromosome for each gene, the algorithm selects the fittest resource from the first place in virtual list, but for the second chromosome, it finds the best resource from the second place in virtual list. The fittest candidate-resource will be searched from the next point, after last chromosome was started from the last place in virtual list and so on like Round-Robin method but for resource selection. This process, continue to making a population. But in making each chromosome, if the counter is finished, the resource selection will be continued from first place.

This method assures that all resources will be selected for making population. Thus, all possible solutions can almost be made, and attended the balance the load on resources.

D. Evaluation

To recognize the value of a solution, we should evaluate it by a fitness function with efficient parameters in quality of solution. This paper chooses function D as fitness function which can intuitively indicate the efficiency of scheduling plan.

$$Fitness = D(S) \tag{8}$$

Where S is a scheduling plan which can be represented by a chromosome. Function $D(S)$ is the duration from the first task started to the last task finished with scheduling plan S.

E. Crossover and mutation operation

These operations are quite important in genetic algorithm, because these implement the principles of evolution. In this paper, adaptive crossover and mutation rate are adopted to

execute crossover and mutation operation. Crossover and mutation rate of HSGA algorithm are determined in advance and remain unchanged in the evolutionary process, which will likely lead to premature, thereby reducing the search efficiency. Literature [8] proposed a new evaluation index to evaluate the premature length of population:

Assume that t_{th} generation of populations consists of M individuals, the best individual fitness value is $F_{t\,best}$, the worst individual fitness value is $F_{t\,worst}$.
then:

$$\Delta = \frac{F_{t\,worst} - F_{t\,best}}{F_{t\,worst}} \quad (7)$$

Crossover rate is computed as:

$$P_c = \Delta \cdot P_{c\,max} \quad (?)$$

Where $P_{c\,max}$ is the maximum of P_c which is usually in the range of [0.2, 0.8].

Mutate rate is computed as:

$$P_m = \Delta \cdot P_{m\,max}$$

Where $P_{m\,max}$ is the maximum of P_m which is usually in the range of [0.1, 0.5].

Random point crossover is used in this paper, shown in fig. 3 and fig.4:

Before Crossover:

V1	V8	V16	V22	V45	...	V34
P3	P7	P2	P1	P9	...	P4

V1	V8	V16	V22	V45	...	V34
P7	P4	P4	P9	P2	...	P3

Fig. 3.

After Crossover:

V1	V8	V16	V22	V45	...	V34
P3	P4	P2	P9	P9	...	P3

V1	V8	V16	V22	V45	...	V34
P7	P7	P4	P1	P2	...	P3

Fig. 4.

Here, a random gene selection crossover is used. Two parents randomly and their some genes are selected randomly. Then two other solutions by change in resource sections of selected genes are created. For example, some genes are selected randomly such as second, fourth and last genes, the candidate resources are changed by each ether, in two random selected chromosomes. Figure 3 and Figure 4 illustrates the before and after crossover in mentioned example.

Experimental results show that mutation operation makes big impact on the algorithm. If a condition that fitness of chromosomes becoming worse is allowed, the population may be on the decline. Thereby, the mutation operation of ETSAA only allows better mutation, which means if a chromosome becomes worse after mutation, the mutation would be cancelled.

F. Elitist selection solution

Though HSGA algorithm work pretty well in most of environments, but it may not convergence when it has run some generations where the best fitness of this running generation may be far away from the global best fitness. Thereby, this paper proposes an elitist selection solution in order to preserve the excellent chromosomes that have been found.

Elitist selection solution is easy to understood intuitively, in selection step for every generation, it chooses the best k chromosomes for next generation. This step can effectively reserve excellent individuals and find the global best chromosomes as soon as possible. The parameter k is usually set to 10% of the population number. Next, elitist chromosomes will not be changed in crossover operation in order to preserve their genes but other chromosomes can get their genes randomly.

G. Pseudo code

Pseudo code of IAHA algorithm

Input: Set of available resources and unmapped tasks
Output: An optimized generated schedule
1. Compute depth of each task in DAG graph
2. Compute the priority of each task according to formula (6), and sort tasks by rule
3. Update resource properties
4. Make a list of available resources
5. Initial population
6. While satisfy stop condition
7. Compute fitness value of every chromosome, adaptively adjust crossover rate and mutation rate according to formula (7),(8),(9)
8. Elitist selection solution is used in selection operation
9. Crossover and mutate to get a new generation of population
10. End while
11. Save the best solution
12. Dispatch all mapped tasks on candidate resources due to obtain the best scheduling

IV. SIMULATION AND RESULT ANALYSIS

To verify correctness and rationality of the algorithm results, this section presents the comparative evaluation ETSAA with traditional GA and FCFS algorithm, and evaluates their completion time with different tasks and processors. The simulation parameters are listed in TABLE 1.

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Number of tasks	30-100
Number of processors	5

Number of generations	1000

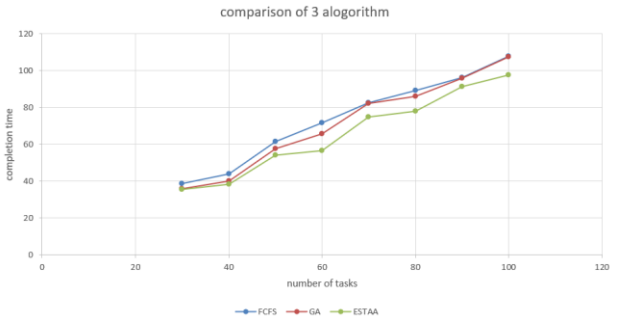


Fig. 5.

In fig.5, other algorithm are compared with ETSAA. The results show, the completion time of tasks for ETSAA is less than traditional GA and FCFS.

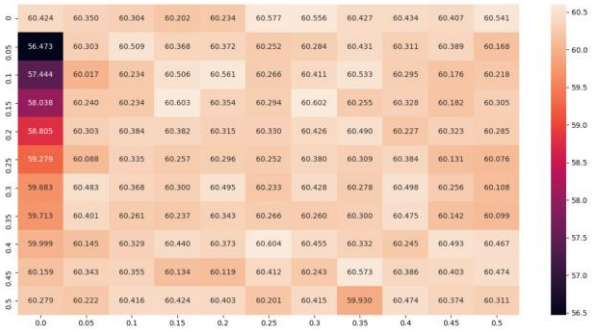


Fig. 6.

Fig. 7.

In fig.6, completion time of different crossover rate and mutation rate for traditional GA are compared. It indicates that parameters selection can make big impact for results.

In fig.7, completion time of different max crossover rate and max mutation rate for traditional GA are compared. It indicates that parameters selection of ETSAA is very loose, which is more capable of dealing with complex task scheduling.

V. Conclusions and Future Work

Elitist task scheduling adaptive algorithm (ETSAA) is proposed in this paper. This algorithm focus to process tasks with complex computation which are running in cloud environment, using the algorithm to map resources to tasks reasonably, thereby give a quickly and effectively response to tasks which are users submitted and handled .

In this paper, we use elitist selection solution and adaptive crossover and mutation rate, and its adaptive extend is not prominent. With elitist selection solution, excellent chromosomes can be preserved for next generation, thereby, ETSAA can find global minimum more quickly. With adaptive parameters, ETSAA can act differently in different stages, which makes it more capable of working in complex task scheduling.

In the meantime, the algorithm proposed in this paper is mainly for complex compute-intensive applications, we hope to find a reasonable method to map tasks to resources in case of communication-intensive applications or compute-intensive and communication-intensive mixing in the future.

References

[1] Ghorbannia Delavar,A.,Aryan,Y.:A Synthetic heuristic algorithm for independent task scheduling in cloud systems.Int.J.Comput.Sci.Issues 2011,8(6),289-295

[2] Yoo,M. :Real-time task scheduling by multi objective genetic algorithm.J.Syst.Softw.2009,Vol 82,619-628

[3] Omara,F.A.,Arafa,M.M.:Genetic algorithms for task scheduling problem.J.Parallel Distrib.Comput.2010,vol.70,13-22

[4] Li Jinmei, Sun Dongwei, Wu Yanxia, Global comparatively optimum static task scheduling algorithm (in Chinese), Application Research of Computes: 2014,Vol.31 No.4, 1027-1030

[5] Ghorbannia Delavar, A., Aryan, Y.: HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems. Cluster Comput, 2014, Vol.17:129-137

[6] Ghorbannia Delavar, A., Aghazarian, V., Litkouhi, S., Khajeh Naeini, M.: A scheduling algorithm for increasing the quality of the distributed systems by using genetic algorithm. Int. J. Inf.Educ. Technol.2011, 1(1), 58–62

[7] Wang, X., Yeo, C.S., Buyya, R., Su, J.: Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. Future Gener. Comput. Syst.2011, Vol. 27,1124–1134

[8] Li Xin, The Improvement of the Adaptive Genetic Algorithm and its Application (in Chinese),NanJing University of Information Science&Technology, May 2008,32-34

[9] Delavar A G , Aryan Y . HSGA: A hybrid heuristic algorithm for workflow scheduling in cloud systems[J]. Cluster Computing, 2014, 17(1).

[10] Ghorbannia Delavar, A., Aryan, Y.: A synthetic heuristic algorithm for independent task scheduling in cloud systems. Int. J. Comput. Sci. Issues 8(6), 289 (2011)

[11] Tanga, X., Li, K., Li, R., Veeravalli, B.: Reliability-aware scheduling strategy for heterogeneous distributed computing systems. J. Parallel Distrib. Comput. 70, 941–952 (2010)

[12] Nicod, J.-M., Philippe, L., Toch, L.: A genetic algorithm to schedule workflflow collections on a SOA-Grid with communication costs. LIFC Laboratoire D'informatique de l'Universite de Franche-COMTE, EA 4269 (2011)

[13] Tang, X., Li, K., Liao, G., Li, R.: List scheduling with duplication for heterogeneous computing systems. J. Parallel Distrib. Comput. 70, 323–329 (2010)

[14] Li, J., Qiu, M., Niu, J., Gao, W., Zong, Z., Qin, X.: Feedback dynamic algorithms for preemptable job scheduling in cloud systems. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (2010)

[15] Casanova, H., Desprez, F., Suter, F.: On cluster resource allocation for

multiple parallel task graphs. J. Parallel Distrib. Comput. 70, 1193–1203 (2010)

[16] Pandey, S.: Scheduling and management of data intensive application workflflows in grid and cloud computing environments. Doctoral thesis, Department of Computer Science and Software Engineering, the University of Melbourne, Australia (December 2010)

[17] Porto, S., Ribeiro, C.: A tabu search approach to task scheduling on heterogeneous processors under precedence constraints. Int. J. High Speed Comput. 7, 45–72 (1995)

[18] Kalashnikov, A., Kostenko, V.: A parallel algorithm of simulated annealing for multiprocessor scheduling. J. Comput. Syst. Sci. Int. 47, 455–463 (2008)

[19] Yoo, M.: Real-time task scheduling by multi objective genetic algorithm. J. Syst. Softw. 82, 619–628 (2009)

[20] Yu, J., Buyya, R., Ramamohanarao, K.: Workflflow scheduling algorithms for grid computing. Department of Computer Science and Software Engineering, The University of Melbourne, VIC 3010, Australia (2009). http://www.cloudbus.org/reports