2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)
March 29–31, 2018, Xiamen, China

An Improved Estimation of Distribution Algorithm for Cloud Computing Resource Scheduling

1st Haisheng Sun
School of Computer Science and Technology
University of Science and Technology of China
Hefei, China
haisheng@mail.ustc.edu.cn

2nd Chuang Liu and 4th Huaping Chen
School of Management
University of Science and Technology of China Hefei, China
chuang@mail.ustc.edu.cn, hpchen@ustc.edu.cn

3rd Rui Xu
School of Business
Hohai University
Nanjing, China
rxu@hhu.edu.cn

***Abstract*—This paper focuses on cloud computing resource scheduling on the Soft as a Service layer and aims at minimizing the user costs by regarding the deadline as a constraint for scheduling independent tasks. Existing works with evolutionary computation approaches fail to describe the interactions among independent tasks. To overcome this problem, an improved Markov-chain-based estimation of distribution algorithm is proposed, and the concept of virtual machine selection diversity is created to construct the probabilistic model rationally. Moreover, one heuristic rule related to the investigated problem is created to keep the population maintaining a high diversity in the evolution process. The experiment results show that the proposed algorithm not only obtains the best solution quality but also has competitive convergence among all compared algorithms.**

Keywords*—estimation of distribution algorithm; markov chain; cloud computing; resource scheduling

I. Introduction

As a novel mode of resource utilization in the form of Soft as a Service (SaaS), Platform as a Service, and Infrastructure as a Service, cloud computing offers considerable value to scientific research and real-world applications [1]. This technology enables end-users to easily purchase services on demand in the pay-per-use model, and the user's demand for cloud computing resource has dramatically increased recently. Therefore, achieving an optimal scheme for cloud computing resources scheduling (CCRS) that satisfies the various requirements of users and cloud providers has become a significant and valuable topic in cloud computing.

Under the pay-per-use model, researches in CCRS problem need to meet the user's requirements of Quality of Service (QoS), which has been seen as a significant issue [2]. So far, many related works have presented various user QoS including single or multiple QoS [3]–[5]. However, the previous works satisfying the user QoS still have certain shortcomings. For example, the various QoS levels expected by different users cannot be met when only single QoS is considered [6]. Moreover, most of the multiple QoS works weight and combine different QoS to form pseudo multi-objective optimization [7] so that the optimization results excessively depend on adjusting the weights. Different from the above studies, literature [8] considers multiple QoS by aiming at minimizing the user cost while ensuring the makespan meet deadline constraint for scientific workflows, and the results show that the scheduling scheme can be produced with a lower execution cost. Following aforementioned ideas, the particular problem investigated in this paper aims at minimizing the user costs by regarding the deadline as a constraint for scheduling independent tasks on the SaaS layer.

CCRS problem can be regard as a many-to-one mapping relation between the required tasks (cloudlets) and the available virtual machines (VMs). Author [9] indicates that the CCRS problem has been considered as an NP-hard problem. Evolutionary computation (EC) which have efficiency and effectiveness in tackling complex optimization problems are widely used to address CCRS issues [10]. Some related works adopting EC approaches are reviewed in the following. For minimizing the makespan, [4] schedules independent and divisible tasks adapting to different computation and memory requirements via the genetic algorithm (GA) in early time, but there is not much improvement in the GA, and concurrently, the simulation is conducted without a cloud environment. Additionally, [11] proposes a hybrid ant colony optimization (ACO) based particle swarm optimization (PSO) to minimize the makespan, but the comparative experiment in this article is not sufficient.

However, the apparent deficiency of previous works such as [3], [4], [11] is assuming that each independent task is entirely irrelevant to its adjacent neighbors when allocating them to virtual machines (VMs). In practice, the arrangement for the current task i will certainly have a certain impact on the VM selection for its subsequent tasks i+1. If task i is first assigned to VM j and then task i+1 is scheduled to the same VM j, the completion time of the latter will be increased because of waiting in the queue. Attempting to capture the dependence information among components of a canditate, this study adopts estimation of distribution algorithm (EDA) [12] whose behaviour is characterized by the probabilistic model applied. EDA which is also a stochastic optimization algorithm shares the advantages of EC approaches and is considered effective at dealing with multi-constraints and multi-criteria decision making processes [13]. Note that, the existing EDA

This work was support in part by the National Natural Science Foundation of China (No.71671168 / No.71433003), the China Postdoctoral Science Foundation (No.2015M581707), the Jiangsu Planned Projects for Post-doctoral Research Funds (No.1501006B).

for tackling CCRS in the literature [14], [15] pertains to the particular EDA that assumes the problem variables are independent. Thus these existing EDA approaches fail to consider the interactions among independent tasks. Inspired by the Markov chain model that can well describe the interactions among variables [16], [17], an improved EDA based on a 1-order Markov chain model (MCEDA) is constructed to cope with the investigated problem efficiently. Moreover, several heuristic rules are created to keep the population maintaining a high diversity in evolution process. Finally, an extremely comprehensive experiment is conducted to prove the effectiveness of the MCEDA. The results show that our MCEDA obtains the best solution quality and has competitive convergence among all the compared algorithms.

The remainder of this paper is organized as follows: Section 2 presents the problem formulation. Section 3 gives the design of the MCEDA in detail. The comparative experiment is provided in Section 4. Finally, we end the paper with conclusions and future work in Section 5.

II. Problem Formulation

This section describes the problem investigated by defining the notation used before presenting the associated mathematical equations in detail. In the following section, the VMs and cloudlets represent resources and tasks, respectively.

A. Notations

Notation	Description
n	Number of cloudlets.
m	Number of VMs.
v_i	The assigned VM index of cloudlet i.
V	(v_1,v_2,,v_n) is the decision vector.
D	Deadline for all cloudlets.
PT_i	Actual processing time of cloudlet i.
WT_i	Queue waiting time of cloudlet i.
MT_i	Completion time for VM j.
Q_j	Waiting queue for VM j.
T	Completion time for all cloudlets.
C_j	Cost per unit time for VM j.
P_j	Processing capability of VM j.
Z_l	The lth generation populations.
Z_l^s	The lth generation dominant (selected) populations.
μ_k	Opportunity coefficient in equations, k=1,2,3.
$V_{l-1,i}$	The set of v_i of each selected individuals in Z_{l-1}^s.
$V_{l-1,i}^x$	The set of v_i when the value of the (i-1)th component (variable) is x in $V_{l-1,i}$.
$v_{l-1,1}^t$	The tth element of set $V_{l-1,1}$.

B. Constraints

Our scheduling is subject to the VMs allocation and time constraints. Then, the decision vector V need to be identified, which is also the solution for the problem. Note that the cloudlet scheduling sequence is irrelevant to the solution. Therefore, the natural increment sequence (1,2,...,n) is adopted in each solution for convenience. The following notations and explanation are employed in the following parts.

- This paper agrees that a static pool of heterogeneous resources that are available to process the tasks is provided.

$$v_i \in \{1, 2, ..., m\}, i = 1, 2, ..., n \tag{1}$$

- We designate a pre-configured processing capability to each VM by floating point operations per second (FLOPS); thus, the actual processing time of each cloudlet is determined according to the VM to which it is assigned.

$$PT_i^{v_i} = L_i / P_{v_i}, i = 1, 2, ..., n \tag{2}$$

  where L_i represents the size of cloudlet i in terms of floating point operations (FLOP).
- Each VM executes the cloudlets according to the FIFO rule. Whenever a new cloudlet i is scheduled on a particular VM j, the cloudlet needs to enter the waiting queue of VM j. Naturally, the queue waiting time for new cloudlet i can be calculated by equation (3).

$$WT_i = \sum_{k=1}^{i-1} PT_k^{v_i}, 2 \leq i \tag{3}$$

  where k represents the number of previous cloudlets before current new cloudlet i.
- The completion time for each VM equals the completion time for the last cloudlet of its waiting queue.

$$MT_j = \sum_{i=1}^{s} PT_i, i \in Q_j, j = 1, 2, ..., m \tag{4}$$

  Where, s is the size of the waiting queue. Thus, the completion time for all cloudlets is equal to the longest completion time of all the VMs. Each solution will give a scheduling program, and only if the maximum completion time T is not more than deadline D can we state that the solution is feasible.

$$T = max\{MT_j\} \leq D, j = 1, 2, ..., m \tag{5}$$

C. Objective

The objective is to minimize the user (execution) cost for processing independent tasks with the deadline constraint on the SaaS layer.

$$\textbf{\textit{min}} \quad Obj(V) = \sum_{j=1}^{m} MT_j \cdot C_j \tag{6}$$

III. Proposed MCEDA

In this section, we elaborate the MCEDA proposed to tackle the problem investigated.

A. Markov-EDA Framework

This paper adopts a 1-order Markov-EDA [12], [16] in which the configuration of the variable z^{n+1} depends on the configuration of the previous variable, which can be formulized as equation (7). Note that, our proposed MCEDA without variables permutation is different from mutual information maximing input clustering (MIMIC) [18] whose main

Algorithm 1 General Framework for EDAs

1: $Z_0 \leftarrow$ Generate N individuals (the initial population) randomly;
2: **for** $l = 0 \rightarrow stopping\ criterion$ **do**
3: $Z_{l-1}^s \leftarrow$ Select $M\,(M \leq N)$ individuals from Z_{l-1};
4: Compute the marginal and conditional probabilities corresponding to each factor of factorization;
5: $Z_l \leftarrow$ Sample N individuals from $p_l(x)$;
6: **end for**

idea is to choose the optimal permutation to produce the lowest pairwise entropy with respect to the true distribution, although, both of them belong to a chain topology. The pseudo code of the Markov-EDA is shown in Algorithm 1.

$$p(z^{n+1}|z^1, z^2, ..., z^n) = p(z^{n+1}|z^n) \tag{7}$$

B. Encoding Scheme

The integer (decimal) encoding scheme is employed in this article. There are m kinds of resource allocation options for each component of a candidate, which can be seen from equation (1).

C. Probabilistic Model

The primary idea of the MCEDA is to construct a probabilistic model to take full advantage of the interaction among independent tasks. We suppose that each cloudlet (variable) in the problem follows the 1-order Markov chain model and regard equation (7) as a factorization of the univariate marginal. In this sense, the joint probability distribution in the MCEDA can be factorized as follows:

$$p(V) = p(v_1) \cdot \prod_{i=2}^{n} p(v_i|v_{i-1}) \tag{8}$$

First, to make the VM allocation process more rational and enrich the population diversity during the evolution, this paper tactfully presents the concept of VM selection diversity $\rho_{l-1}(x)$, which means that a different VM x assigned to execute the current ith cloudlet has a diverse effect on the VM selection of the following (i+1)th cloudlet in the lth generation. Considering the significance of the first component and its unrestricted VM selection space, the first variable is chosen as the sample space to count the number denoted as $N(x)$ of the first variable from S_l^s when $v_1 = x$. Then, we add up the number denoted by $d(x)$ of variant VMs in the second component under the premise of $v_1 = x$. In the case that no $v_1 = x$ exists, a greater value tolerated is assigned. Thus, we measure the VM selection diversity by the equation (9). The smaller $\rho_{l-1}(x)$ is, the stronger the influence of VM x is and the greater the probability value for the appearance of x in the lth generation.

$$\rho_{l-1}(x) = \begin{cases} N(x)/d(x), & x \in V_{l-1,1} \\ \mu_1 \cdot max\{N(x)/d(x)|x \in V_{l-1,1}\}, & otherwise \end{cases} \tag{9}$$

Second, the relative frequency is considered to compute the marginal probabilities $p(v_1)$. The frequencies based on the statistics of the selected individuals about the first component stand for the marginal probabilities. Here, $f_{l,1}(v_1 = x)$ means the frequencies of the first variable x, and $p_{l,1}(v_1 = x)$ is the normalization probability in the lth generation.

$$f_{l,1}(v_1 = x) =$$

$$\begin{cases} N(x)/M, & x \in V_{l-1,1} \\ \mu_2 \cdot min\{N(x)/M|x \in V_{l-1,1}\}, & otherwise \end{cases} \tag{10}$$

$$p_{l,1}(v_1 = x) = f_{l,1}(v_1 = x) / \sum_{x=1}^{m} f_{l,1}(v_1 = x) \tag{11}$$

Next, considering the characteristics of the studied problem, the approach for calculating the conditional probabilities need to be ameliorated. According to the diverse conditions of the current variable (component) in S_l^s , the traditional conditional probabilities are separately divided into three cases: *1)* both $v_{i-1} = x_2$ in $V_{l,i-1}$ and $v_i = x_1$ in $V_{l,i}^{x_2}$ are available, i=2,3,$\cdots$,n; *2)* there is no $v_{i-1} = x_2$ existing in $V_{l,i-1}$; *3)* otherwise. Accordingly, the improved conditional probability denoted by $f_{l,i}(v_i = x_1|v_{i-1} = x_2)$ can be expressed as

$$f_{l,i}(v_i = x_1|v_{i-1} = x_2) =$$

$$\begin{cases} N(x_1)/N(x_2), & \\ 1/m, & x_2 \notin V_{l-1,i-1} \\ \mu_3 \cdot min\{N(x_1)/N(x_2)|\psi\}, & otherwise \end{cases} \tag{12}$$

where $\psi = x_2 \in V_{l-1,i-1} \& x_1 \in V_{l,i}^{x_2}\ i = 2, 3, ..., n$

Furthermore, a real-time heuristic information is introduced to help define the conditional probabilities $p(v_i|v_{i-1})$ appropriately and to precisely select the available resources. As described, the problem noted in this paper is limited by deadline D such that the total processing time needs to be minimized as much as possible. Let $F_{l,i}(x)$ expresses the inverse queue waiting time expressed as equation (11). Thus, the larger $F_{l,i}(x)$ is , the shorter WT_i is, and the greater the probability value for assigning to VM x in the lth generation is.

$$F_{l,i}(x) = 1/WT_i, v_i = x \& i = 2, 3, ..., n \tag{13}$$

Finally, the normalized transition probability can be determined by combining $p_{l,1}(v_1 = x)$, $F_{l,i}(x)$ and $f_{l,i}(v_i = x_1|v_{i-1} = x_2)$ using a set of exponents, which is expressed as follows:

$$\begin{aligned} &p_{l,i}(v_i = x_1|v_{i-1} = x_2) = \\ &f_{l,i}(v_i = x_1|v_{i-1} = x_2)^a \cdot p_{l,1}(v_1 = x)^b \cdot F_{l,i}(x)^c/\eta \end{aligned} \tag{14}$$

where $\eta = \sum_{x_1=1}^{m} f_{l,i}(v_i = x_1|v_{i-1} = x_2)^a \cdot p_{l,1}(v_1 = x)^b \cdot F_{l,i}(x)^c$ and parameters a, b and c are the importance of each item to the selection of the offspring and are all positive.

D. Population Sampling

From the above, the marginal probabilities $p(v_1)$ of the proposed model can be acquired by equation (11), and the conditional probabilities $p(v_i|v_{i-1})$ can be obtained by using equation (14). Then, the sampling processing for each component in the solution can be conducted with a straightforward approach from these probability model. In order to enhance the exploration capacity of the proposed algorithm, a small part of new individuals will be generated by cross and mutation of the best superior individual of the last generation.

E. Algorithm Presentation

The infeasible solutions, which appear because of the deadline constraint, should be corrected after the sampling processing is performed. Consequently, a correction algorithm (CA), whose main idea is to migrate the most time-consuming cloudlet of each solution to the most unoccupied VM, rises in response to the proper time and conditions. Here, the most unoccupied VM indicates that the reborn MT_j can be calculated by equation (15) and is at a minimum while scheduling cloudlet k to VM j. Note that, in the following simulation experiments, the parameter D will be set by considering both the processing capability of VMs and the length of cloudlets to ensure that there is at least one feasible solution can be corrected by CA. The pseudo code of CA is presented in Algorithm 2.

Algorithm 2 CA

Begin

1: **for** $i = 1 \rightarrow N$ **do**
2: **while** $D \leq T[i]$ **do**
3: Order all VMs in the MT_i in non-increasing order;
4: Select the cloudlet with maximal PT_i from the first VM;
5: Migrate it to the optimal VM obtained with equation (15);
6: Calculate $T[i]$.
7: **end while**
8: **end for**

End

$$MT_j = min\{MT_j + PT_k | v_k = j\} \tag{15}$$

With the declarative probabilistic model and sampling procedure, our MCEDA is proposed as Algorithm 3. As for the time complexity, Z_0 is generated with the time complexity of $O(Nn)$ and the time complexity of Algorithm 2 for each individuals is $O(n + mlogm)$. Computing the objective of all solutions costs $O(Nn)$ time and sequencing all the solutions requires $O(NlogN)$ time. The operation of constructing equation (9) has a time comlpexity of $O(Mm)$. Calculating the probability values by equation (11) or (14) can be done in $O(nmM)$ and the time complexity of sampling step is $O(Nn)$ and the elitism criterion only need $O(1)$ time. Therefore the total time complexity of all steps in Algorithm 3 is $O(N(n + mlogm) + nmM)$.

Algorithm 3 MCEDA

Begin

1: Set the initial population $Z_0 = \emptyset$, selected population $Z_0^s = \emptyset$, generation counter $l = 0$;
2: Generate $Z_0 = \{V_1, V_2, , V_N\}$ at random;
3: Applying Algorithm 2 to make correction procedure;
4: **for** $l = 1 \rightarrow l_{max}$ **do**
5: Calculate (6) for each solution V;
6: Sequence all the solutions in term of objective values in non-decreasing order;
7: $Z_{l-1}^s \leftarrow$ select the top M individuals;
8: Construct (9);
9: **for** $i = 1 \rightarrow n$ **do**
10: **if** $i == 1$ **then**
11: Construct and solve (11);
12: **else**
13: Construct and solve (14);
14: **end if**
15: **end for**
16: Sample new population and applying Algorithm 2;
17: elitism criterion
18: **end for**
19: The best individual in Z_l is the final solution.

End

The specific parameter settings employed for MCEDA are as follows. The rational population size of MCEDA is set to $N = n \times m$, and $n \times m$ represents the assignation that schedules n cloudlets to m VMs. Then, the truncation selection strategy is applied, which chooses the top $M = 20 \times N$ individuals. Furthermore, an improved elitism criterion is employed, which converts the worst individual $worst_l$ in this generation to the best individual $best_{l-1}$ in the last generation after sampling. Finally, the maximum iteration $l_{max} = 5 \times n$ is seen as a termination criterion.

IV. Performance Evaluation

In this section, the CloudSim framework [19] was used to simulate a cloud environment that has a single data center. In the following experiments, the policy that determines the share of processing power among cloudlets in every VM is space-shared stated in CloudSim. Research [20] states that there is no optimal algorithm for CCRS because of its NP-hard complexity and EC algorithms are the most powerful of heuristic methods. Therefore, abundant experiments are conducted to compare the proposed MCEDA with several existing related ECs and EDAs.

A. Parameter Settings

The VM configurations are based on the most recent computation and optimization resources of the current Amazon elastic compute cloud and are given in Table I. The method proposed by the [21] is employed to estimate the VM processing capacity based on the number of ECUs. We specify the unit time on which the pay-per-use model is based as one

hour, which means that any partial utilization of the leased VM is charged as if the full one-hour period was occupied.

TABLE I. CONFIGURATIONS OF VM

Name	ECUs(Cores)	GFLOPS	Cost($/h)
c4.large	8	35.2	0.212
c4.xlarge	16	70.4	0.423
c4.2xlarge	31	136.4	0.847
c4.4xlarge	62	272.8	1.694
c4.8xlarge	132	580.8	3.571
c3.large	7	30.8	0.209
c3.xlarge	14	61.6	0.418
c3.2xlarge	28	123.2	0.836
c3.4xlarge	55	242.0	1.672
c3.8xlarge	108	475.2	3.344

Because of having no benchmark for CCRS, the size of cloudlets is generated from a discrete uniform distribution ranging from 5×10^5 to 1×10^6 gigaflops. Given the provisions that the total number of VMs in the experiment is K ($K \leq 10$), then the top K types of VM in Table I are selected and the deadline parameter D is set by equation (16) with $\omega = 0.5$ unless otherwise indicated, where P_{max} and P_{min} represent the maximum and minimum processing capability of all VMs, respectively. Meanwhile, the top t cloudlets are chosen in term of the size of L_i to obtain D.

$$D = \sum_{i=1}^{t} \frac{L_i}{P_{max}} + \omega \cdot (\sum_{i=1}^{t} \frac{L_i}{P_{min}} - \sum_{i=1}^{t}) \frac{L_i}{P_{min}}), t = \lceil \frac{n}{m} \rceil \quad (16)$$

B. Comparative Algorithms

1) EC Approaches: First, the most commonly used GA was selected. To maintain the diversity of the population, we use the concept of "generation gap" and set the Gap=10 %. The GA parameter settings in [22] are employed. The next EC approach is the ACO and the special parameter settings in [23] are employed.

2) EDA: First, the earliest population based incremental learning (PBIL) recognized as initially solving the binary optimization is selected. Our experiment takes the top $20\%\times N$ individuals to learn the model and sets the learning rate $alpha$=0.01. MIMIC [18] is selected as the second comparative method, in which the conditional probabilities are obtained from simple statistics on dominant population.

C. Cost Evaluation

In this section, the average user costs obtained for each instance of all algorithms are compared on various configurations: (A) 20×4, (B) 50×6, (C) 80×8, (D) 80×10, (E) 100×10, (F) 150×10. For each configuration, three instances are generated and all algorithms are executed ten times for each instance. The parameters recorded are $AD \pm SD$, where $AD = \sum_{i=1}^{10}(Obj_i - Opt)/10$ stands for the average deviation and Opt represents the optimal (best) solution obtained during all the executions for each instance. The standard deviation of the solution is denoted as SD. Table II shows that the comparative results varies with instances. The results in bold indicate the algorithm that obtains Opt; the results with strikethrough denote that the algorithm that achieves the lowest AD; the results marked with * represent that the corresponding algorithm obtains the solution in the optimal region (OR) at least once during ten executions, where $OR = [Opt, Opt \times (1 + 2\%)]$

TABLE II. COST EVALUATION ON VARIOUS SCALE INSTANCES

	MCEDA	GA	ACO	PBIL	MIMIC
A	0.19±0.1*	~~**0.17**~~±0.1*	0.24±0.1*	0.23±0.1*	0.25±0.1*
	0.21±0.0*	0.11±0.1*	~~**0.10**~~±0.0*	0.21±0.0*	0.21±0.0*
	0.17±0.1*	~~**0.02**~~±0.1*	0.22±0.0*	0.23±0.0*	0.23±0.1*
B	**0.63**±0.2*	0.62±0.1*	~~0.60~~±0.1*	1.04±0.1*	0.52±0.1*
	~~**0.25**~~±0.2*	0.69±0.0*	0.62±0.1*	0.95±0.1*	0.76±0.2*
	0.50±0.2*	~~0.48~~±0.1*	0.69±0.1*	1.20±0.1*	1.07±0.2*
C	~~**0.76**~~±0.2*	2.91±0.1	2.30±0.1	3.30±0.3	2.28±0.2
	~~**1.52**~~±0.6*	2.73±0.1	2.07±0.1*	3.12±0.3	1.90± 0.2*
	~~**0.90**~~±0.3*	2.67±0.1	2.04±0.1*	2.90±0.4	2.02± 0.3*
D	~~**2.73**~~±1.4*	7.10±0.7	6.10±0.2	6.95± 0.2	5.05±0.6
	~~**2.38**~~±0.4*	6.79±0.5	5.98±0.2	6.71± 0.3	5.25±0.3
	~~**.26**~~±0.2*	7.07±0.8	6.04±0.2	6.95± 0.1	5.66±0.4
E	~~**3.34**~~±0.3*	7.97±0.7	6.91±0.3	8.06± 0.2	5.85±0.3
	~~**2.28**~~±0.3*	8.09±0.7	7.09±0.2	8.19± 0.2	6.08±0.5
	~~**2.48**~~±0.5*	8.21±0.5	7.27±0.1	8.32± 0.2	5.45±0.5
F	~~**4.31**~~±0.0*	11.6±0.6	10.3±0.3	11.9± 0.2	7.58±0.5
	~~**3.62**~~±0.6*	11.4±0.9	9.87±0.3	11.8± 0.2	7.81±0.5
	~~**4.77**~~±0.8*	11.7±0.9	10.3±0.1	8.8±0.3	8.63±0.5

For obtaining Opt, the MCEDA had significant superiority over the other algorithms, achieving 15 $Opts$ out of the total 18 instances. In contrast, only 2 $Opts$ were obtained by GA and one by ACO. Compared to the other algorithms, the MCEDA still maintained a clear advantage regarding the lowest AD, acquiring 13 best ADs of all the instances. Remarkably, the AD of MIMIC is lower than any other algorithm except MCEDA as a whole. which means that EDA with dependance is more efficient. Moreover, the MCEDA satisfied the OR for all the instances, i.e., the MCEDA obtained all solutions in OR within ten repetitions. In addition, ACO and MIMIC have more ability of acquiring the OR than GA and PBIL. Note that MCEDA and MIMIC show better performance when the instances are large-scale and have abundant VMs, which can be validated by analyzing the AD values.

In conclusion, most of Opts were obtained by the MCEDA, and over half of the lowest ADs were also contributed by the MCEDA. Additionally, the MCEDA appeared to be strongly capable of obtaining all the solutions in the OR within ten executions. Accordingly, these distinct advantages indicate that the MCEDA should be seen as an effective alternative and applicable to solve the investigated problem in this paper.

D. Convergence Analysis

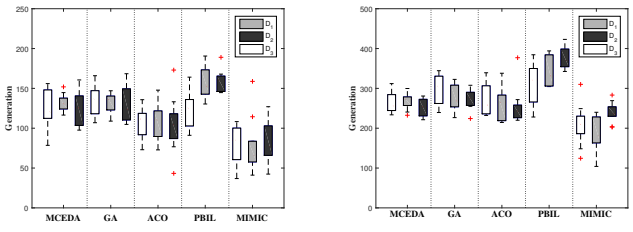
This section is devoted to the analysis of the convergence of the comparative algorithms under configurations 50×6 and 100×10. Ten instances were generated for each configuration, and each algorithm was executed ten times for each instance.

This experimentation was carried out with three deadline parameters D_1, D_2 and D_3, which are generated with $\omega = 0.25$, $\omega = 0.5$, and $\omega = 0.75$ by equation (16).

The generation in which the final result first appeared was recorded for each execution and could reflect the convergence speed since all algorithms share the same termination criterion for each instance. The span of each box represents the stability of the corresponding algorithm.

1) Convergence Speed: As seen in Fig. 1, it is clear that MIMIC has the fastest convergence speed and MCEDA had a lower median generation than the corresponding result of either the GA or PBIL. As to the 50×6 instance, the median generation of the MCEDA was not inferior to that of the ACO. However, for the more-VMs instance 100×10, the convergence speed of the MCEDA was similar to that of the ACO. Furthermore, the deadline parameters had less effect on the MCEDA when the scale of instance increases.

2) Convergence Stability: When compared on the 50×6 instances, the MCEDA did not have the advantage of convergence stability. For the 100×10 instances, the MCEDA can achieve a smaller box span than the others. Moreover, the deadline parameters had also less influence on the MCEDA than the others.



(a) 50 × 6. (b) 100 ×10.

Fig. 1. Box plots of convergence analysis

In conclusion, the MCEDA achieved a competitive convergence speed and performed relatively stable among the comparative algorithms on two configurations.

V. Conclusion

In this paper, a cloud computing resource scheduling problem for minimizing the user costs for processing independent tasks with a deadline constraint on the SaaS layer was analyzed and addressed. To solve the problem, an improved variable-dependent approach MCEDA was proposed. The experiment showed that the MCEDA has significant superiority over other algorithms for minimizing the user cost, particularly for large-scale and more-VM instances. In addition, MCEDA was also competitive on convergence speed and stability.

There are still many valuable research points worth addressing to improve the solving algorithm or the CCRS problem in this paper, i.e., the way to construct and solve the marginal probability of the first component can be improved. At the present stage, CCRS is still a front-burner issue, and the design of the scheduling algorithm that has outstanding performance is also a challenge faced by researchers.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," J. Int. Services. and Apps., vol. 1, no. 1, pp. 7–18, Apr. 2010.

[2] J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," Proc. IEEE, vol. 99, no. 1, pp. 149–167, Jan. 2011.

[3] P. Kumar and A. Verma, "Independent task scheduling in cloud computing by improved genetic algorithm," Int. J. Adv. Res. Comput. Sci. Softw., vol. 2, no. 5, May. 2012.

[4] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent tasks scheduling based on genetic algorithm in cloud computing," Proc. Int. Conf. Wireless Commun. Netw. Mobile. Comput., pp. 1–4, Sept. 2009.

[5] Z. G. Chen, K. J. Du, Z. H. Zhan, and J. Zhang, "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm," Proc. IEEE Congr. Evol. Comput., pp. 708–714, May. 2015.

[6] S. Singh *et al.*, "Qrsf: Qos-aware resource scheduling framework in cloud computing," J. Supercomput., vol. 71, no. 1, pp. 241–292, Jan. 2015.

[7] G. Guo-ning, H. Ting-lei, and G. Shuai, "Genetic simulated annealing algorithm for task scheduling based on cloud computing environment," Proc. Int. Conf. Intell. Comput. Integr. Syst., pp. 60–63, Oct. 2010.

[8] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 222–235, Jun. 2014.

[9] T. A. Genez *et al.*, "Workflow scheduling for saas/paas cloud providers considering two sla levels," IEEE. NOMS., pp. 906–912, Apr. 2012.

[10] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," J. Netw. Syst. Manage., vol. 23, no. 3, pp. 567–619, Jul. 2015.

[11] X. Wen, M. Huang, and J. Shi, "Study on resources scheduling based on aco allgorithm and pso algorithm in cloud computing," Proc. 11th Int. Symp. DCABES, pp. 219–222, Oct. 2012.

[12] P. Larranaga and J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation. Springer Science and Business Media, 2002, vol. 2.

[13] J. Ceberio *et al.*, "A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem," IEEE Trans. Evol. Comput., vol. 18, no. 2, pp. 286–300, Apr. 2014.

[14] N. Chen, X. Fang, and X. Wang, "A cloud computing resource scheduling scheme based on estimation of distribution algorithm," Proc. 2nd Int. Conf. Syst. Informat., pp. 304–308, Nov. 2014.

[15] K. Niu, J.-D. Wang, H.-W. Zhang, and W. Na, "Cloud resource scheduling method based on estimation of distirbution shuffled frog leaping algorithm," 3rd Int. Conf. Cybers. Technol., pp. 1–6, Oct. 2015.

[16] Z. Yuan, "Prediction of protein subcellular locations using markov chain models," FEBS letters, vol. 451, no. 1, pp. 23–26, May. 1999.

[17] R. Santana, P. Larraaga, and J. A. Lozano, "Protein folding in simplified models with estimation of distribution algorithms," IEEE Trans. Evol. Comput., vol. 12, no. 4, pp. 418–438, Jul. 2008.

[18] J. S. De Bonet, C. L. Isbell Jr, and P. Viola, "Mimic: Finding optima by estimating probability densities," Adv. Neural Inf. Proc. Syst., vol. 51, no. 4, pp. 233–241, 1999.

[19] R. Calheiros *et al.*, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw. Pract. Exper., vol. 41, no. 1, pp. 23–50, Jan. 2011.

[20] Z.-H. Zhan *et al.*, "Cloud computing resource scheduling and a survey of its evolutionary approaches," ACM. Comput. Surv., vol. 47, no. 4, pp. 1–33, Jul. 2015.

[21] S. Ostermann *et al.*, "A performance analysis of ec2 cloud computing services for scientific computing," Int. Conf. Cloud Comput., pp. 115–131, Berlin,Germany: Springer,2009.

[22] J. W. Ge and Y. S. Yuan, "Research of cloud computing task scheduling algorithm based on improved genetic algorithm," Appl. Mecha. Mater., vol. 347, pp. 2426–2429, Aug. 2013.

[23] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 53–66, Aug. 1997.