IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 25, 2019, accepted April 22, 2019, date of publication April 25, 2019, date of current version May 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913175

An Energy-Aware Algorithm for Virtual Machine Placement in Cloud Computing

DA-MING ZHAO[1], JIAN-TAO ZHOU[1], AND KEQIN LI[2], (Fellow, IEEE)
[1]College of Computer Science, Inner Mongolia University, Hohhot 010021, China
[2]Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

Corresponding author: Jian-Tao Zhou (cszhoujiantao@qq.com)

The research is supported by Natural Science Foundation of China under Grant No. 61662054, 61262082, Natural Science Foundation of Inner Mongolia under Grand No.2015MS0608, Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering and Inner Mongolia Application Technology Research and Development Funding Project "Mutual Creation Service Platform Research and Development Based on Service Optimizing and Operation Integrating", Inner Mongolia Engineering Lab of Cloud Computing and Service Software and Inner Mongolia Engineering Lab of Big Data Analysis Technology.

ABSTRACT Virtualization technology, as a key technology in cloud computing, makes the virtual machine placement (VMP) play an important role in improving the energy efficiency of data centers. In this paper, an energy-aware algorithm named GATA is proposed for the VMP problem. It combines the genetic algorithm with the tabu search algorithm. The goal is to obtain an optimal VMP scheme to achieve energy efficiency while maximizing load balance among various resources. The algorithm is compared with two meta-heuristic algorithms and a newly proposed algorithm that is based on ant colony algorithm. The execution time of these algorithms is also discussed. The results show that the proposed algorithm is superior to those methods mentioned above.

INDEX TERMS Genetic algorithm, power-aware, tabu search algorithm, virtual machine placement.

I. INTRODUCTION

Cloud computing [1] is a special form of distributed computing that introduces utility models to remotely provide scalable and measurable resources, and it is also the commercial implementation of grid computing, parallel computing, and distributed computing [2]. The cloud environment, compromised of enormous inexpensive infrastructure that provides an IT pool of resources as a service leased on an "on demand" basis [3]. Consumers always deem that the cloud computing has the infinite computing and storage capabilities. They only need to pay for their actual IT resources usage without taking the management costs into account. Through the Internet, they can also access the cloud service anywhere.

Data center [4] is defined as an special IT infrastructure which is responsible for placing IT resources centrally, including servers, databases, network, communication devices and software systems. With the development of cloud computing technology, users' demand for cloud resource also increase. Therefore, cloud service providers such as Amazon EC2 [5], Alibaba cloud and Google take measures such as upgrading their hardware or increasing the number of servers. The more physical nodes to be created, the more energy will be consumed by the dada center. According to the U.S. department of Energy, the data center of cloud accounts for 1.5 of all U.S. energy consumption which increases at the speed of 12 each year [6].

Low resource utilization is another major reason for the energy inefficiency of data center [7]. Even if the workload is low, such as 10% CPU utilization, the power consumed is over 50% of the peak power. In order to effectively improve the utilization of the cloud and resources, virtualization technology becomes indispensable [8]. It has many advantages such as server consolidation [9], online migration [10], isolation, high availability, flexible deployment and low management overhead [11]. Virtualization brings a lot of solutions to the energy management of cloud computing. For example, at the data center level, virtualization technology can integrate multiple virtual machines (VMs) into the same physical machine through server consolidation and then shut down idle physical machines (PMs) to save energy.

As an important component of resource allocation and management in cloud computing, virtual machine placement (VMP) problem [12], [13] aims to choose a suitable physical machine for a virtual machine according to a certain method and strategy. With the premise of constraints and restrictions, it can achieve resource efficient and reduce energy consumption. And VMP is known as an NP-hard problem which is always regarded as a bin-packing problem.

Although turning on fewer hosts can save energy. However, it is well known that if too many VMs are hosted on the same host, although the resource utilization of

The associate editor coordinating the review of this manuscript and approving it for publication was Mustafa Servet Kiran.

the host increases, this also aggravates the competition of resources. The response time of VMs to requests will increase accordingly [14]. Conversely, hosting fewer VMs results in a waste of resources. Therefore, in order to minimize energy consumption, it is particularly important to reasonably allocate the resources of the PM to the VM so as to improve the overall resource utilization of the data center. In this paper, the VMP problem is considered as a multi-dimensional bin-packing problem which is also a multi-objective optimization problem. The memory and CPU of a physical host are treated as constraints. To reduce the energy consumption while maximizing the load balance of data center, a new algorithm combines genetic algorithms (GA) and tabu search algorithms, namely GATA, is proposed to determine the placement of virtual machines. This algorithm regards the tabu search algorithm as a mutation operator to improve the local search ability of genetic algorithm. To illustrate the effectiveness of this algorithm, it is compared with the existing virtual machine placement algorithms, such as the traditional genetic algorithm, the simulated annealing algorithm (SA) and a newly proposed method which is an improved ACS-based algorithm. The experimental results show that this method can turn off unnecessary hosts as much as possible in a short time based on energy-awareness and improve the load balance of the data center.

The rest of this paper is organized as follows. Section II discusses related work. The energy-aware model is built in Section III and the GATA algorithm is described in Section IV. Section V presents the experimental results and compares with other methods. In Section VI, the paper is concluded.

II. RELATED WORK

VMP has been proved as an NP-hard problem. In the traditional placement method, it is usually solved by a static placement algorithm or a dynamic placement algorithm. Verma *et al.* [15] proposed a static solution called CBP which considers the positive correlation constraints between programs in order to ensure VMs are not deployed on a server and verified it can reduce the possibility of the SLA violation. Besides, he also developed three dynamic placement algorithm named mPP, iFFD and pMaP that are aim at minimizing energy consumption, the operation of migration and the total costs respectively in [16].

Most researches about VMs' placement strategy are based on the single objective optimization with a certain criterion. Therefore, the optimal placement strategy obtained is only an optimal solution subjects to this criterion. Obviously, it is difficult to compare the optimal placement among these criterions. An effective way is to consider multiple objectives and make tradeoffs among them. Gao *et al.* [13] was committed to reducing both energy consumption and resource wastage. Thus, a multi-objective ant colony system algorithm is designed to obtain the Pareto set. Ibrahim *et al.* [17] presented an adaptive genetic algorithm to achieve energy efficient while considering the response time. Raju *et al.* [18] proposed an algorithm called EAMOCA in the hybrid cloud, which aimed to minimize execution time and energy consumption while maximizing the resource utilization.

This problem is usually abstracted as bin-packing problem or solved by using heuristic algorithms such as genetic algorithm, ant colony algorithm and simulated annealing algorithm. In previous years, Gao *et al.* [13] proposed a modified version of the ant colony system (ACS) algorithm called VMPACS, which is designed to deal with the potential large solution space for large-scale data centers effectively. Mi *et al.* [19] developed an online self-reconfiguration approach, named GABA, which aims to find the optimal reconfiguration policy based on genetic algorithm. Srikantaiah *et al.* [20] studied the inter-relationship among energy consumption, resource utilization and performance of consolidated workloads. Then he regarded the consolidation as a multi-dimensional bin-packing problem of allocating and migrating workloads to achieve the optimal energy consumption. Wu *et al.* [21] designed a Simulated Annealing based algorithm which combined with the FFD method to solve the problem of virtual machine placement.

In recent years, on the basis of the existing methods, some new algorithms with specific goals were proposed. Liu *et al.* [22] proposed an ACS-based approach named OEMACS. In the process of ant colony search, the state transition criterion is established according to the bottleneck of various resources from the perspective of global optimization. Meanwhile, for the allocation scheme that fails to find a feasible solution, order exchange and migration local search techniques are introduced, which turns an infeasible solution into a feasible solution and then updates the optimal solution. The energy consumption is greatly decreased and the utilization of various resources is improved. Xin *et al.* [23] presented an improved Knee Point-Driven Evolutionary Algorithm named EEKnEA. It aimed to improve the performance of KnEA further and an energy-efficient-oriented population initialization strategy was used to obtain a higher quality initial population. The experiment demonstrated that their method was able to minimize energy consumption and maximize load balance, resource utilization and robustness. Besides, reducing the possibility of physical machine overload. Hong and Ge [24] formulated the VMP problem as a combinatorial optimization problem and designed a heuristic approach based on an improved ant colony algorithm named GACA. Genetic algorithm is used to optimize the calculation of pheromone and support obtaining the global optimal solution with high convergence speed.

In general, the multi-objective optimization problem is a contradiction between each sub-objective. The improvement of a sub-objective may cause others' performance degradation. And it is impossible to achieve the optimal value simultaneously. An efficient way is to coordinate and compromise among them so that each sub-objective can be optimized as much as possible. For example, it is well known that turning off idle physical hosts is a simple and effective way to save energy. But it will increase the resource competition of other

servers, it may thus cause them to consume more energy and degrade their performance. An effective method should consider tradeoffs among all these sub-objectives.

III. PROBLEM STATEMENT AND MODEL FORMULATION

In a cloud environment, there is a resource pool consisting of many physical hosts which are all fully virtualized. And it is assumed that all hosts are homogeneous and each of them contains a certain number of VMs which are heterogeneous. The problem of VMP can be regarded as a two-dimensional bin-packing problem. The bin represents a physical host which has variable length and width. And the items are the VMs to be deployed. The width of the bin is the host's memory, and the length of the bin represents the host's utilization of CPU. If a host contains multiple VMs, it can be deemed that the host's memory and CPU utilization as the sum of these VMs. For example, if the CPU utilization and memory of two virtual machines on a physical host are $(10, 2)$ and $(20, 3)$ respectively, then it can be considered that the CPU utilization of the physical host is 30% and the memory size is 5GB at this time.

A. VM PLACEMENT

The problem of VMP in the cloud computing can be briefly described as follows:

1) $V = \{v_1, v_2, ..., v_n\}$ is the set of virtual machines to be deployed. n is the number of VMs. Each element in the set is a virtual machine, where $v_i = \{r_i, c_i\}$ are the memory size and the CPU utilization of the ith virtual machine respectively.

2) $H = \{h_1, h_2, ..., h_m\}$ is the set of destination hosts. m is the amount of PMs. Each element in the set is a physical host, where $h_j = \{R_j, C_j\}$. R_j and C_j are the memory size and CPU utilization of jth physical host respectively.

3) The objective is to find a subset of H called H' which is represented as $H' = \{h'_1, h'_2, ..., h'_m\}$. And it is necessary to find a map $f : V \rightarrow H'$ that should not only meet the resource requirements of VMs, but also increase the amount of idle physical hosts so that the energy consumption can be reduced. At the same time, the map should also ensure that the total requirements of the VMs do not exceed the capacity of the physical host it deployed.

Figure 1 shows three possibilities for resource utilization. Figure 1(a) illustrates that three virtual machines are currently deployed on a physical host. It can be seen that there is a large amount of space left due to the unreasonable deployment. Figure 1(b) is another case where the memory of the physical host is not fully utilized. Figure 1(c) shows the purpose of the proposed method in this paper, that is, improve the energy efficient through reasonable deployment so that resources can be fully utilized.

B. POWER CONSUMPTION MODELING

The energy consumption of a physical host consists of idle energy consumption, CPU energy consumption, memory energy consumption, hard disk energy consumption, network energy consumption and other system components.

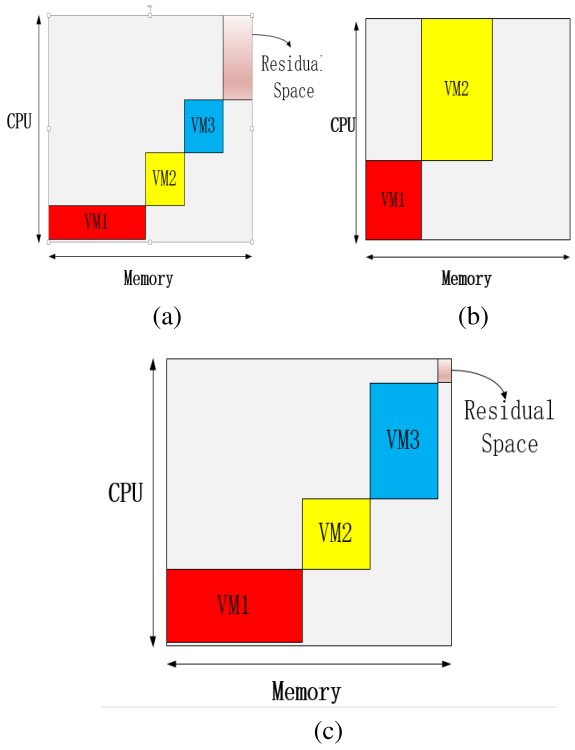

FIGURE 1. Resource utilization.

Most literatures have demonstrated that the energy consumption of CPU is the largest contributor to the energy consumption of a system, and the relationship between the energy consumption of a physical host and the utilization of CPU can be expressed linearly [25]. In the experiment, each physical host is divided into three states which can be presented as E_j^{busy}, E_j^{idle} and E_j^{off}. And the energy consumption model proposed in [26] is also applicable to the experimental environment of this paper. Therefore, the energy consumption of j th physical host can be calculated as:

$$E_j = \begin{cases} (E_j^{busy} - E_j^{idle}) \times U_j + E_j^{idle} & U_j > 0 \\ E_j^{off} & otherwise \end{cases} \quad (1)$$

where E_j^{busy} and E_j^{idle} are the average measured power consumption of the jth host when it is idle and the CPU utilization reaches 80%. The values of them are assumed to 215 and 162 Watt in the simulation experiments according to the [13]. Since each host in the data center is assumed to be homogeneous, these values are same among servers. E_j^{off} represents the energy consumption of the jth server when it is shut down, with the value of 0. U_j represents the CPU utilization of the jth host, which is equal to the sum of the CPU utilization of all VMs deployed on that server. It can be formalized as: $U_j = \sum_{i=1}^{n} X_{ij} C_i$. X_{ij} is a binary number and it is set to 1 if the ith VM is placed on jth PM. If there are no VMs placed on the jth PM, then X_{ij} is set to 0. C_i is the CPU utilization of the ith VM.

C. LOAD BALANCE MODELING

The purpose of load balancing is to ensure that the resources of each physical host are effectively allocated and equitably used so as to meet the user's QoS requirements [19]. In order to avoid excessive resource competition among virtual machines on the same host, L_j is used to denote the distance between the resource demand of the jth PM and the resource threshold of the jth PM. L_j can be calculated as:

$$L_j = \sqrt{\left(\left(U_j - T_C\right)^2 + \left(R_j - T_R\right)^2\right)} \tag{2}$$

where R_j is the memory utilization of the jth host, which is equal to the sum of the memory utilization of all VMs deployed on that host's. It can be formalized as: $R_j = \sum_{i=1}^{n} X_{ij} r_i$. r_i is the memory utilization of the ith VM. The CPU and memory thresholds for each host are set as T_C and T_R. These two values are usually set to 80 and 100 respectively according to the research in [27]. The smaller the value of L_j, the more reasonable the allocation of resources on PM j.

D. PROBLEM DEFINITION

Based on the previous analysis, the optimization model of the VMP problem can be described as:

Minimize:

$$E = \sum_{j=1}^{m} E_j \tag{3}$$

$$L = \sum_{j=1}^{m} L_j \tag{4}$$

Subject to:

$$\sum_{j=1}^{m} X_{ij} = 1 \quad \forall i \in V, \forall j \in H \tag{5}$$

$$\sum_{i=1}^{n} c_i X_{ij} \leq T_C \times y_j \quad \forall i \in V, \forall j \in H \tag{6}$$

$$\sum_{i=1}^{n} r_i X_{ij} \leq T_R \times y_j \quad \forall i \in V, \forall j \in H \tag{7}$$

$$y_i, X_{ij} \in \{0, 1\} \tag{8}$$

Constraint (6) and (7) ensure the capacity of each host is not exceed their thresholds. y_j is a binary number which represents whether a host is in use or not. It can be calculated as:

$$y_j = \begin{cases} 0 & \sum_{i=1}^{n} X_{ij} = 0 \\ 1 & otherwise \end{cases} \tag{9}$$

As mentioned in the above, VMP is a process of mapping n VMs to m PMs. Therefore, there will be m^n possible mappings. It is necessary to design an algorithm which can obtain the optimal solution of placement and is able to solve this multi-objective problem.

IV. THE DESCRIPTION OF THE PROPOSED GTBA ALGORITHM

The algorithm proposed to solve the problems formulated in Section III is mainly based on a genetic algorithm. The genetic algorithm is a random search optimization algorithm based on biological evolution and molecular genetics. It is selforganizing, adaptive and learning. However, the traditional genetic algorithm has the disadvantages of ''premature''. Tabu search algorithm employs an individual serial search method to record the trajectory of searching and guides the search direction by setting a tabu table to prevent ''inbreeding'', maintains the diversity of the population and overcomes the defect of ''premature convergence.'' Therefore, the local searching ability of the GA can be improved by converting the tabu search algorithm into a mutation operator to search for the optimal solution.

The pseudocode of the proposed GATA algorithm is described in Algorithm 1. The algorithm works as follows: In the initialization phase, the parameters involved in the genetic algorithm and the tabu search algorithm are set. The VMP problem to be solved is encoded by real-coding and the fitness of the initial population is evaluated. During iterative phase, individuals are selected, crossed, and mutated. For the mutation operation, a tabu search operation is performed, the candidate solution is generated from the neighborhood of the solution. Moreover, the local search ability is improved according to the aspiration criteria. After each iteration, the minimum number of hosts obtained by the current generation is passed to the next iteration as the number of hosts. When the genetic algorithm reaches the number of iterations, the global optimal solution will be obtained as the final placement scheme.

A. ENCODING

An equal-length real-coding is used to represent a placement scheme which is also known as an individual in a group. The terms ''physical machine'' and ''host'' will be used interchangeably in this paper. The order of the numbers of the physical machines loaded by each virtual machine v_i constitutes the chromosome's coding of the problem. For example, 23421124, represents a placement scheme. It indicates that the first, the fourth and the seventh virtual machine are deployed on the second host. The second is deployed on the third host. The third and the last virtual machines are deployed on the fourth host. The fifth and sixth virtual machines are deployed on the first host. The initial population can be generated by the random arrangement from 1 to m where m is the number of the physical machine. The number of hosts used for the optimal solution is set as m^{min}. In the initial stage, since the optimal solution is unknown, it is assumed that $n = m = m^{min}$. In other words, the worst solution in the initial stage is to place n virtual machines on n hosts with one VM mapping to one PM. In addition, from the individual's chromosome encoding string, it can be easily counted how many physical hosts are utilized in

Algorithm 1 GATA

Input: The number of VMs n and minimal physical hosts m^{min} that are same initially, Parameters required in GATA

Output: The optimal placement scheme of VMs based on energy-aware

```
/*Initialization*/
Set values of parameter:
pc,pm,pop_size,chrome_length,iteration,tabuLen,tsGen,
neighbourLen
/*Generate the initial population*/
population ← geneEnconding(pop_size, chrome_length)
/*Evaluate each individual generated in the initial population*/
calFitVal(pop_size, population)
/*Iterative loop*/
for i = 0 → iteration do
    /*Select individuals */
    Adapt the Roulette Wheel choice to remain individuals
    se ← selection(pop_size, population)
    /*Crossover operation*/
    Select individuals according to the crossover rate
    Adapt the Two-point crossover to remain individuals
    cro ← crossover(pop_size, pc, se)
    /*Mutation operation*/
    Select individuals according to the mutation rate
    Transform the tabu search algorithm into a mutation operator
    mu ← mutation(pop_size, pm, cro)
    Compute the fitness of the mutation individual as an asp
    asp ← calFitVal(pop_size, mu)
    Perform the tabu search algorithm for a mutation
    /*Tabu Search operation*/
    for j = 0 → tsGen do
        for k = 0 → neighbourLen do
            newPopulation ← tabuSearch(mu, asp)
            bestNeighbour ← sortNeighbour(newPopulation)
        end
    end
    /*update m^min*/
    m^min ← countNumber(bestNeighbour)
    i ← i + 1
end
Evaluate the fitness of the population after iterations and obtain the optimal placement scheme.
```

a placement scheme. Therefore, it is feasible to solve this problem by using genetic algorithm. Figure 2 illustrates this placement scenario.

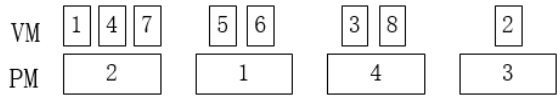

FIGURE 2. VM Placement.

B. FITNESS CALCULATION

In the genetic algorithm, the fitness of individuals are used to evaluate the degree of individual fitness, so as to determine their possibility of inheritance. Due to limitations of the host's CPU utilization and memory thresholds, the penalty function is used to efficiently accelerate the process of searching to the optimal. To efficiently eliminate the unsatisfactory solution, for the placement where the CPU utilization or the memory utilization exceeds its thresholds, the penalty value is set to ensure this individual has very little chance to be selected and it is a constant with a value of 50. Therefore, the punish function can be calculated as:

$$F_{punish\_j} = \begin{cases} 50 & U_j > T_C \vee R_j > T_R \\ 0 & otherwise \end{cases} \tag{10}$$

In the objective function, it is considered that the energy consumption and the resource utilization of the data center are equally important. Therefore, the product of them is used as the objective function's value of the lth chromosome which is defined as:

$$f_l = E \times L \tag{11}$$

In this paper, the objective function is a non-negative value, and is based on the minimum value of the function as the optimization goal, so the fitness function of the lth chromosome is defined as:

$$F_{fitness\_l} = C_{max} - f_l - n \times F_{punish\_j} \times F_{punish\_j} \tag{12}$$

where C_{max} can be calculated as: $C_{max} = n \times s^2$. The value of C_{max} is changeable which is depend on the number of VMs. The larger the number of VMs, the larger the value of C_{max}. s is a constant with value of 100 and it is obtained through many experiments of different scales. Its function is to guarantee that the fitness function always takes a non-negative value and it can ensure that the smaller the objective function value is, the greater its fitness will be. $F_{fitness\_l}$ is the fitness of lth chromosome in a iteration where $1 \leq l \leq N$, and N is the number of chromosomes in this iteration.

C. THE SELECT OPERATOR

The main purpose of selection operator is to avoid gene deletion, improve global convergence and computational efficiency. In this paper, the Roulette Wheel choice is adopted to remain the best fitness individual which will appear in the next generation as much as possible. Although the selection error of this method is large, sometimes the individual with higher fitness may even not be selected, it is still a commonly used selection operator. The probability P_l of the individual l

is selected as:

$$P_l = \frac{F_{fitness\_l}}{\sum_{l=1}^{N} F_{fitness\_l}} \tag{13}$$

Next, the crossover operation is performed to increase the diversity of a population, which is described in the next subsection.

D. THE CROSSOVER OPERATOR

Crossover operators in genetic algorithms are used to generate new individuals. A pair for individuals whose crossover rate are less than the set value are chosen and two-point crossover is carried out which is shown in Figure 3. Genes on each locus of two paired individuals are exchanged at the same crossover probability to form two new individuals. Specifically, two positions pos_1, pos_2, are randomly selected on the chromosome as cross points, and then the portions between the two chromosomes pos_1 and pos_2 are exchanged.

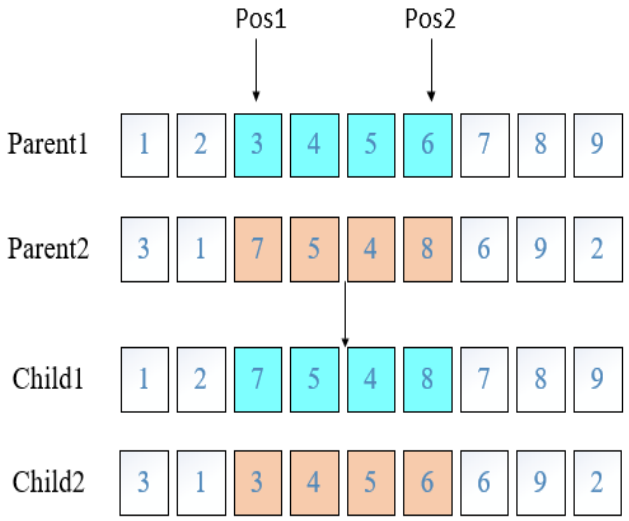

FIGURE 3. Two-point crossover.

E. THE MUTATION OPERATOR

The mutation operation is a method to prevent the algorithm from premature. However, genetic algorithm is easy to fall into local optimum when solving. In addition, the convergence speed is slow and the efficiency of searching is low. Therefore, the tabu search algorithm is transformed into a mutation operator to improve the local search ability of genetic algorithm. The operation process is as follows:

Step 1: Set parameters of the tabu search algorithm. The chromosome which meets the mutation condition is regarded as the initial solution of the tabu search. The value of the fitness generated by this solution is taken as the aspiration criterion and this solution is regarded as *Sgb*. The tabu table has its length and it is empty initially.

Step 2: Determine whether the tabu search achieves the maximum number of its iterations. If so, the algorithm is terminated and the optimization result is obtained. Otherwise, continue with the following steps.

Step 3: Perform tabu search for the selected chromosome which firstly generates some of its neighbors. In this paper, two genes of a chromosome are randomly selected, and the positions of these two genes pos_1, pos_2 are exchanged. Then this chromosome can be viewed as a neighbor.

Step 4: Sort neighbors according to their fitness and then choose those neighbors with high fitness as candidate solutions. In particularly, the number of candidate solutions should less than the number of neighbors.

Step 5: Select the chromosome with the highest fitness in candidate solutions. Then judging whether its fitness is higher than the aspiration criterion. If it is true, then no matter whether it is in tabu table, the *Sgb* is replaced with this chromosome. If the tabu table is not full, then this chromosome can be added directly. Otherwise, this chromosome is put into the tabu table according to FIFO criterion which replaces the chromosome that first entered the tabu table. Then it goes to step 7. Otherwise, continuing with the following steps.

Step 6: Choose the best chromosome in the candidate solution which is not in tabu table as current solution if the aspiration criterion is not satisfied. At the same time, if the tabu table is not full, then this chromosome can be added directly. Otherwise, the corresponding placement of the candidate chromosome is used to replace the placement that first entered the tabu table according to the FIFO criterion.

Step 7: Return to step 2.

Step 8: Return a new population to the selection operation after performing the above operations on all chromosomes involved in the mutation.

When the algorithm finishes, the individual fitness within the population is evaluated and the placement of VMs with the highest fitness is selected.

V. EXPERIMENTAL EVALUATION

In this section, a series of simulation experiments are conducted on the proposed GATA algorithm from the aspects of the energy consumption, the load balance and the execution time of the placement scheme. The experimental results are compared with the simulated annealing algorithm (SA) proposed in [21], the traditional genetic algorithm and a newly proposed algorithm based on ant colony algorithm [22].

A. EXPERIMENTAL SETUP

In the experiment, the implementations of all algorithms were written in Python and ran on a PC which has an Intel Core i7-8750H processor with 2.2GHz CPU and 8 GB of RAM usable. Besides, the programs for the proposed algorithm were coded in Python and its operating environment is PyCharm 3.3. The operating system is Windows10. Using real workload to do experiments is extremely significant for VM placement. In this paper, Google Cluster Data (GCD) dataset which provides traces over a 29 days period in May 2011 [28] is adopted. The GCD workload comprises 672003 jobs, each with one or more tasks. To create the CPU

and the memory utilization of VMs, the tasks of each job are aggregated by summing their CPU and memory consumption every five minutes in a period of 24 hours. n VMs with CPU and memory utilization less than their threshold value are filtered out from these jobs. Those VMs are used to be deployed.

TABLE 1. Parameters of genetic algorithm in GATA.

pop_size	pc	pm	chromo_length	iteration
20	0.8	0.2	n	30

In GATA, because of the combination of tabu search algorithm, the setting and combination of multiple parameters in these two algorithms have various effects on the experimental results. Through a large number of experiments on different scales, the final parameters were determined. For parameters related to the genetic algorithm in the proposed algorithm, it is shown in Table 1. The population size is 20. The rate of crossover and mutation is 0.8 and 0.2 respectively. The *chromo_length* is equal to the number of VMs and its iteration is 30. For parameters related to tabu search algorithm in the proposed algorithm, it is shown in Table 2. The neighbor size is 20 and the iteration of tabu search is 15. The length of tabu table is set to $\sqrt{n}$ according to experience, n is the number of VMs. And the aspiration criterion is determined by the fitness of the optimal placement strategy in all iterations which is changeable. In addition, the initial population is generated randomly and it is assumed that physical hosts in the data center are homogeneous. The upper bound of CPU utilization and memory utilization are set as 80 and 100 to avoid resource competition according to [27]. Furthermore, the setting of each parameter in the SA algorithm is the same as those in [21].

TABLE 2. Parameters of tabu search algorithm in GATA.

tsGen	tabuLen	neighborLen
15	$\sqrt{n}$	20

B. EVALUATION OF MULTI-OBJECTIVE PLACEMENT OF TWO-DIMENSION PACKING

Obviously, the size of the VMP problem varies with the number of VMs and PMs. Specifically, the amount of VMs are 10, 30, 50, 100 and 200 in the experiment. In the initial stage, the amounts of VMs is assumed to be equal to the amounts of PMs. And in the worst case, the upper bound of PMs should not exceed the amounts of VMs.

Table 3 lists the results of the proposed algorithm compared with two heuristic algorithms GA, SA and an improved algorithm based on ant colony algorithm named OEMACS. The first column of the table is the scale of the problem. The third column of the table corresponds to the sum of load balance of each placement strategy. The load balance improved by GATA compared with other algorithms is shown in the last column. Taking GA as an example, the value of improved load balance can be calculated as:

$$LI = (1 - L\_GATA/L\_GA) \times 100\% \tag{14}$$

where L_GATA and L_GA represent the value of load balance of GATA and GA respectively. For each algorithm, it was performed 20 times and the average value was taken as the final result.

TABLE 3. Comparison results of GATA with two evolution algorithms of load balance.

Problem	Algorithm	Load Balance	LI(%)
10×10	GA	127.21	1.7
	SA	126.16	0.9
	OEMACS	128.75	2.9
	GATA	125.02	0
30×30	GA	749.97	32.1
	SA	652.72	21.9
	OEMACS	535.98	4.9
	GATA	509.55	0
50×50	GA	1661.95	45.9
	SA	1324.50	31.1
	OEMACS	929.29	1.8
	GATA	912.58	0
100×100	GA	4011.65	51.7
	SA	3258.76	40.6
	OEMACS	2015.78	3.9
	GATA	1936.36	0
200×200	GA	10077.41	58.8
	SA	7335.12	43.4
	OEMACS	4461.20	6.9
	GATA	4154.31	0

From Table 3, it is obvious that compared with these heuristic algorithms, GATA has the lowest value of load balance which means that this kind of allocation scheme can make the resource allocation more reasonable so that it can avoid excessive resource wastage. For example, *hostA*'s cpu utilization is 70 and its memory utilization is 10, *hostB*'s cpu utilization and its memory utilization are all 40. Then *hostA*'s load balance is 90.5 and *hostB*'s load balance is 84.8 according to the Equation (2). It is obvious that *hostA*'s cpu resource is close to its threshold while its memory resource has large free space. This type of allocation is not desirable, so it has higher load balance value. In contrary, the other host has enough space to response new requests, so it is more likely to be selected. Here, the load balance of all hosts in the data center is taken into considered instead of one.

Energy consumption is also an objective in this paper. According to Equation (1), it can be known that if the number of hosts obtained by the optimal solution of any two algorithms is equal, then their energy consumption will also be the same. Lower power consumption means fewer hosts are active in the data center. The energy consumption of these algorithms is shown in Table 4 under different problem scales.

TABLE 4. Comparision results of GATA with three evolution algorithms of energy consumption(W).

VMs	GATA	GA	SA	OMEACS
10	11150.35	11150.35	11150.35	11150.35
30	35633.34	35939.49	35763.34	35661.89
50	57752.15	58741.25	58334.15	57759.25
100	110921.80	113498.20	111183.14	111020.35
200	228765.32	230635.64	229872.16	229020.54

From Table 4, it can be found that the energy consumption is proportional to the number of VMs. However, for algorithms with the same energy consumption when the problem scale is small such as 10×10, their load balances are different in Table 3. And this issue has also been discussed above. For different energy consumption of the same problem scale, GATA always gets the lowest energy consumption with the lowest load balance value compared with the other three algorithms.

As the problem size increases, the execution time of each algorithm also changes accordingly. Figure 4 shows the execution time of these algorithms to obtain the final deployment solution.

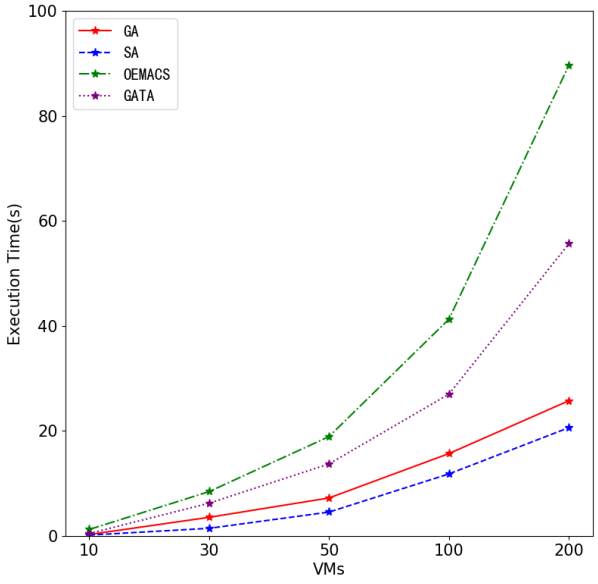

FIGURE 4. Execution time.

It can be seen from Figure 4 that the execution time of each algorithm is proportional to the size of the problem. And it is obviously that among the changes in the slope of these curves, the growth rate of OEMACS is significantly faster than the other three algorithms. The reason behind this is that at the end of each iteration of OEMACS, the order exchange and migration local search are innovatively introduced. This avoids the algorithm falling into the local optimal solution and turning an infeasible solution into a feasible solution, but it also increases the number of searches, resulting in an increase in time and the gap among these algorithms also increases. Besides, the result of OMEACS does not superior to the proposed algorithm GATA. That's the why the problem scale is set to 200 instead of increasing. Compared with the proposed GATA, GA has a faster execution time due to the lack of the tabu search process. Compared with GA, SA algorithm has no selection and crossover operation, and the search process of SA can be regarded as a simplification of tabu search, that is, searching neighbors of the initial individual and accepting the suboptimal solution with a certain probability.

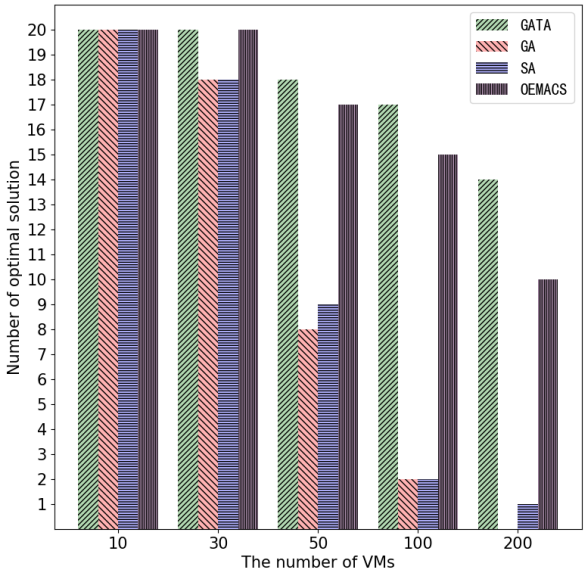

FIGURE 5. The number of optimal solution.

Figure 5 shows the number of each algorithm obtaining the optimal solution in 20 experiments. When deploying a large number of VMs in a data center, providers are often concerned with the reliability of the placement scheme. As can be seen from Figure 5 that GATA has the most optimal number of solutions in the 20 experiments which also proves that it can always give the optimal solution for different problem sizes. Although there is a certain probability that the optimal solution cannot be obtained in GATA, it is acceptable compared with other algorithms. In addition, GA gets the least optimal solution compared with the other three algorithms.

C. ANALYSIS OF GATA PARAMETERS

The GATA algorithm includes iteration number (*iteration*), crossover rate (*pc*), mutation rate (*pm*), the number of chromosomes (*pop_size*), chromosome length (*n*), the number of tabu iteration (*tsGen*), the length of tabu table (*tabuLen*), and neighbor size (*neighborLen*). In addition, the number of active hosts is taken as the evaluation criteria.

For the parameters of the genetic algorithm, *n* depends on the size of the problem, that is, how many VMs need to be deployed. And *iteration* determines the final solution. In a population, *pop_size* determines the diversity of genetic algorithms. The greater the value is, the better the diversity

of the population will be. But it will increase the amount of calculation and reduce the operating efficiency. However, if the value is too small, it will be more prone to premature because of the decrease of genetic diversity. The investigation begins with the parameter *pop_size*. The value of it varies from 5 to 30 with a step length of 5. The problem scale is 10×10, 30×30 and 100×100. All the other parameters remain the same as stated above.

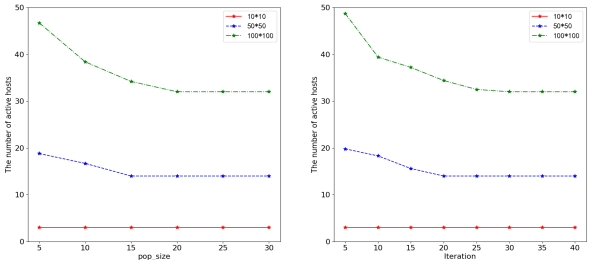

FIGURE 6. Influence of the parameter in GATA on *pop_size* and *iteration*.

In Figure 6(a), regardless of the scale of the problem, when *pop_size* increases to 8, the results are invariant. To reduce the computational burden, this paper adopts the population size of 20 in GATA.

The next parameter tested is iteration which is shown in Figure 6(b). The value of it varies from 5 to 40 with a step length of 5. All the other parameters remain the same as stated above. The tendency of the curves is similar to the parameter *pop_size*. In order to obtain the optimal solution with a greater probability in different scales, this paper adopts the iteration of 30 in GATA.

The crossover rate *pc* determines the frequency of newly generated individuals, which is one of the key parameters to ensure the diversity of the population. If *pc* is too small, the generation of new individuals will slowly affect the diversity of the population and the ability to inhibit premature will be poor. However, the *pc* should not be too large, because the excessive *pc* will make the genetic inheritance unstable and good genes are more easily to be destroyed, making the performance of the genetic algorithm similar to the performance of random search algorithms. 0.8 is commonly used in most literatures [23]. This value is also adopted in this paper.

Mutation is also a way of producing new individuals. The small probability of mutation is not conducive to the generation of new individuals and has an impact on the diversity of the population. However, too large *pm* will make the genetic inheritance of the gene unstable. In the genetic algorithm, mutation is a small probability event, and the effect of mutation operator on the population should be far less than that of crossover operator. It is generally recommended that the value of variation probability is 0.2 which is also adopted in this paper.

For the tabu search algorithm, the parameter *tabuLen* is discussed in the above. The investigation begins with the parameter *neighborLen* which represents the range of possible searches. And it is directly related to the candidate solution. If the value of it is too large, the computational memory and time will increase. If the value of it is too small, the algorithm will fall into the local optimum. The value of it varies from 5 to 25 with a step length of 1. All the other parameters remain the same as stated above.

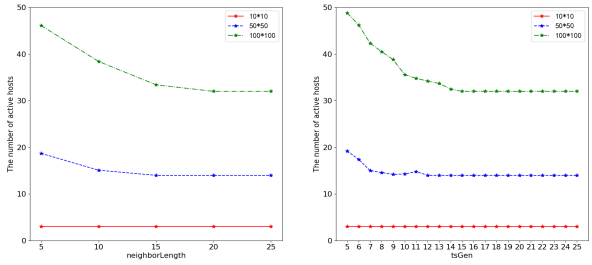

FIGURE 7. Influence of the parameter in GATA on *neighborLen* and *tsGen*.

In Figure 7(a), it can be seen that in scale 10×10, the number of active hosts is stable. In scale 50×50, after the value of 15, the number of active hosts remains unchanged. And in scale 100×100, after the value of 20, the number of active hosts remains unchanged. Because the size of the problem is increasing, too large *neighborLen* will increase the calculation time, this paper adopts the *neighborLen* of 20 in GATA in order to obtain the optimal solution with a greater probability in a short time.

Finally, parameter *tsGen* is discussed. Since tabu search is embedded in the mutation operation, the number of tabu iterations not only determines the quality of the solution after the mutation operation, but also determines the execution time of the algorithm. The value of it varies from 5 to 25 with a step length of 1. All the other parameters remain the same as stated above.

Figure 7(b) shows that the number of hosts is decreasing as the number of *tsGen* increases. In scale 10×10, the number of active hosts remains stable. In scale 50×50, the number of active hosts remains stable after the sixth generation. In scale 100×100, this value changes to 11. In order to shorten the execution time, this paper adopts the *tsGen* of 15 in GATA so as to be adaptive to larger scale.

VI. CONCLUSION

With the development of virtualization technology, how to deploy VMs in the cloud computing environment based on energy-aware has become a hot research topic. In this paper, the VMP problem is regarded as a two-dimensional bin-packing problem. And an energy-aware algorithm called GATA is proposed which combines the genetic algorithm and the tabu search algorithm. The objective is to decrease the energy consumption of the data center while maximizing the load balance of the data center. And the algorithm is compared with some existed VMP algorithms. The experimental results show that the GATA algorithm can achieve a more energy-efficient and a more reasonable placement scheme when compared with two heuristic algorithms and a newly proposed ant colony algorithm in an acceptable time.

Finally, the number of optimal solution of these algorithms are discussed.

REFERENCES

[1] M. Armbrust, "Above the clouds: A berkeley view of cloud computing," *Science*, vol. 53, no. 4, pp. 50–58, 2009.
[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. Grid Comput. Environ. Workshop*, Nov. 2009, pp. 1–10.
[3] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *Eprint Arxiv*, vol. 12, no. 4, pp. 6–17, 2010.
[4] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, and H. Tenhunen, "Energy-aware VM consolidation in cloud data centers using utilization prediction model," *IEEE Trans. Cloud Comput.*, to be published.
[5] A. Ishida, "3. amazon ec2(cloud computing)," *Ipsj Mag.*, vol. 50, pp. 1068–1073, Jul. 2009.
[6] P. Kurp, "Green computing," *Commun. ACM*, vol. 51, no. 10, pp. 11–13, 2008.
[7] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, *A Taxonomy Survey Energy-Efficient Data Centers Cloud Computing System*. Amsterdam, The Netherlands: Elsevier, 2011.
[8] V. Petrucci and O. Loques, "A framework for dynamic adaptation of power-aware server clusters," in *Proc. ACM Symp. Appl. Comput.*, 2009.
[9] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of HPC applications," in *Proc. Int. Conf. Supercomputing*, Jun. 2008, pp. 175–184.
[10] L. Liu *et al.*, "GreenCloud: A new architecture for green data center," in *Proc. 6th Int. Conf. Ind. Session Autonomic Comput. Commun. Ind. Session*, Jun. 2009, pp. 29–38.
[11] J. L. Berral *et al.*, "Towards energy-aware scheduling in data centers using machine learning," in *Proc. Int. Conf. Energy-Efficient Comput. Netw.*, Apr. 2010, pp. 215–224.
[12] J. Xu and J. A. B. Fortes, *Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments*. Washington, DC, USA: IEEE Computer Society, 2010.
[13] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, Dec. 2013.
[14] J. Wang, C. Huang, H. Kai, X. Wang, C. Xi, and K. Qin, "An energy-aware resource allocation heuristics for VM scheduling in cloud," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun. IEEE Int. Conf. Embedded Ubiquitous Comput.*, 2014.
[15] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proc. Conf. Usenix Tech. Conf.*, Jun. 2009, p. 28.
[16] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," in *Proc. ACM/IFIP/USENIX Int. Conf. Distrib. Syst. Platforms Open Distrib. Process.*, 2008, pp. 243–264.
[17] H. Ibrahim, R. O. Aburukba, and K. El-Fakih, "An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers," *Comput. Electr. Eng.*, vol. 67, pp. 551–565, Apr. 2018.
[18] R. Raju, J. Amudhavel, N. Kannan, and M. Monisha, "A bio inspired energy-aware multi objective chiropteran algorithm (EAMOCA) for hybrid cloud computing environment," in *Proc. Int. Conf. Green Comput. Commun. Electr. Eng.*, Mar. 2014, pp. 1–5.
[19] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers," in *Proc. IEEE Int. Conf. Services Comput.*, Jul. 2010, pp. 514–521.
[20] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proc. Conf. Power Aware Comput. Syst.*, 2008, p. 10.
[21] Y. Wu, M. Tang, and W. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2012, pp. 1245–1250.
[22] X. F. Liu, Z. H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.
[23] Y. Xin, Y. Yin, and L. Lan, "Energy-efficient many-objective virtual machine placement optimization in a cloud computing environment," *IEEE Access*, vol. 5, pp. 16006–16020, 2017.
[24] L. Hong and Y. Ge, "GACA-VMP: Virtual machine placement scheduling in cloud computing based on genetic ant colony algorithm approach," in *Proc. 12th Int. Conf. Ubiquitous Intell. Comput.*, Aug. 2016, pp. 1–9.
[25] X. Fan, W. D. Weber, and L. A. Barroso, *Power provisioning for a warehouse-sized Computer*. New York, NY, USA: ACM, 2007.
[26] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proc. IEEE/ACM Int. Conf. Grid Comput.*, Sep. 2011, pp. 26–33.
[27] S. Lee *et al.*, "Validating heuristics for virtual machines consolidation," *Microsoft Res.*, vol. 9, pp. 1–4, Jan. 2011.
[28] Z. Liu and S. Cho, "Characterizing machines and workloads on a google cluster," in *Proc. Int. Conf. Parallel Process. Workshops*, Aug. 2012, pp. 1–9.

DA-MING ZHAO received the B.S. degree in software engineering from the Harbin University of Science and Technology, Harbin, China, in 2016, and the M.S. degree in software engineering from Inner Mongolia University, Hohhot, China, in 2018, where he is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science and Technology. His research interests include energy-aware for cloud computing, virtualization, and algorithm design.

JIAN-TAO ZHOU received the Ph.D. degree from Tsinghua University, in 2005. Since 1999, she has been on the faculty with Inner Mongolia University, China, where she is currently a Professor. Her research interests include formal methods, cloud computing, and software engineering.

KEQIN LI is a Suny Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor with Hunan University, China. He has published over 630 journal articles, book chapters, and refereed conference papers. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyberphysical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing. He is an IEEE Fellow. He has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.