Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 558 – 565

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

A Novel Family Genetic Approach for Virtual Machine Allocation

Christina Terese Joseph[a,*], Chandrasekaran K[b], Robin Cyriac[a]

[a]*Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology, Kochi 682 039, Kerala, India*
[b]*Department of Computer Science and Engineering, National Institute of Technology, Karnataka, Surathkal 575 025, India*

Abstract

The concept of virtualization forms the heart of systems like the Cloud and Grid. Efficiency of systems that employ virtualization greatly depends on the efficiency of the technique used to allocate the virtual machines to suitable hosts. The literature contains many evolutionary approaches to solve the virtual machine allocation problem, a broad category of which employ Genetic Algorithm. This paper proposes a novel technique to allocate virtual machines using the Family Gene approach. Experimental analysis proves that the proposed approach reduces energy consumption and the rate of migrations, and hence offers much scope for future research.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: Cloud Computing; Genetic Algorithm; Virtual machine allocation; Family gene; Energy-efficient

1. Introduction

The term Cloud Computing is a fuzzy term for which no concrete definition exists. It can be viewed as the delivery of services over the Internet, as and when the customer demands. The transition of large organizations from the traditional CAPEX model to the OPEX model support the fact that Cloud computing is one of the most promising technologies in the current IT scenario. The increasing number of users for Cloud Computing increases the challenges faced by the Cloud service providers to provide the requested services ensuring high availability and reliability of the services. The virtualization technique proves functional in helping the Cloud service providers to meet these challenges.

In order to employ virtualization, virtual entities of the actual versions are created and deployed in the system. This technology enables the Cloud service provider to serve more number of customers than the support provided by the actual hardware resources available with the provider. Generally virtualization is applied at the computer system level. This involves the creation and deployment of virtual machines. The requests of the customers will then be processed by these Virtual Machines (VMs). Various VMs will have different processing and memory requirements. One of the major factors that needs to be considered in systems that deploy VMs is the allocation of the VMs to hosts.

* Corresponding author. Tel.: +91-944-652-1820.
E-mail address: xtina_1232@hotmail.com

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.090

An improper allocation can result to the VMs being executed on unsuitable hosts, which can then lead to unwarranted effects. This would affect the credibility of the Cloud service provider. In order to avoid this, an efficient scheme should be used to correctly allocate the VMs to the hosts that support their execution. This decision problem is called the VM Allocation Problem.

Various approaches have been applied to solve the NP-Hard problem of VM allocation. The VM allocation problem can be considered as a multi-objective constrained optimization problem. A large number of approaches applied to solve the VM Allocation problem employ evolutionary techniques including Genetic Algorithm (GA). Some of the limitations of the Genetic Algorithm approaches include the premature convergence and the high processing time involved. Due to premature convergence, many of the Genetic Algorithms converge to a sub-optimal result. These phenomena should be avoided. Another limitation is the high processing time. Most of the Genetic Algorithm approaches require a lot of time for processing the various generations before producing the optimal result. The approach proposed in this paper attempts to overcome these limitations of the Genetic Algorithm approaches to VM Allocation. In general, the paper aims to:

- Perform a literature survey on the various evolutionary approaches to resource scheduling and allocation.
- Propose a novel technique to allocate VMs which overcomes the limitations of the GA-based approaches.
- Compare the experimental results of the proposed approach with the results of the existing approaches.

The organization of the paper is as follows: Section 2 gives an outline of the various evolutionary approaches used to solve resource scheduling problems in environments that employ virtualization. Section 3 defines the problem. Section 4 gives the details of the proposed system. Section 5 presents the experimental analysis and results and the paper is concluded in Section 6.

2. Related Works

Barbagallo et al. propose a bio-inspired technique that aims at reducing the energy consumption in data centers through redistribution of load among the servers[1]. According to the authors, the proposed algorithm can be efficiently employed in self-organizing systems. H.Chen et al. propose a method inspired by the foraging behaviour of ants, which globally allocates resources in Cloud[2]. An improved version of the ant colony optimization algorithm that adopts the characteristics of greedy algorithms as well is presented. The proposed algorithm attains load balancing and improved scheduling time. The task scheduling in Cloud is reduced to an optimal matching problem with multiple objectives by using a bipartite graph model to represent the tasks. The resource allocation problem that considers the dependency among VMs as well as the utilization of the links of the network is considered by C. Wang et al[3]. The authors consider the scenario where the requests for resources are not independent. The proposed algorithm adopts the characteristics of PSO. A single resource request is represented using an entity called Virtual Cloud Embedding (VCE). Dong et al. propose a genetic algorithm approach that works in a distributed manner to place VMs[4]. The proposed approach may be used by IaaS Cloud providers to reduce the energy consumption and thus improve the efficiency. The optimization technique- Ant Colony Optimization (ACO) may be used to effectively consolidate VMs in a data center[5]. The performance of such algorithm can be enhanced by incorporating a distributed and parallel nature. The parallel nature also improves the scalability of the algorithm. E. Feller et al. propose a decentralized schema and propose the use of the ACO-based approach to improve the efficiency of VM consolidation by reducing the number of migrations that have to be carried out[6]. An approach to consolidate VMs using ACO to maximize resource utilization and reduce energy consumption is proposed by Ferdaus et al[7]. A Genetic algorithm approach is proposed by Paolo et al. to allocate VMs in distributed systems with more than one tiers[8]. A variation of the Genetic Algorithm called Improved Genetic Algorithm (IGA) is proposed by Zhong et al. to allocate VMs in data centers of IaaS cloud service providers[9]. The Reordering Grouping Genetic Algorithm Approach (RGGA) was proposed to solve the multidimensional binpacking problem of VM allocation by Wilcox et al.[10] Load balancing and history information was also considered in the Genetic Algorithm approach to allocate VMs by Bandi et al.[11]. The idea of Pareto dominance and simulated annealing are combined to solve the multi-objective problem of VM allocation with the objectives of load balancing and power saving in MOGA-LS[12]. The energy consumption due to communication within the data center network is one of the parameters considered in the approach using Genetic Algorithm to place

VMs by Grant et al.[13] An extension to this work uses a repairing procedure[14]. A Hybrid Genetic Algorithm approach is used to efficiently allocate VMs by Tang et al.[15].

A major class of the bio-inspired methods for VM placement employs Genetic Algorithm. The proposed approach uses a variation of the Genetic Algorithm approach, called Family Genetic Algorithm (FGA), which tries to overcome the limitations of the Genetic Algorithm approaches.

3. Problem Definition

The VM Placement problem is a multi-objective optimization problem. Our aim is to find an optimal placement, which is a mapping from VMs to hosts. Consider a system with "m" host machines and "n" VMs. Each VM is represented as v_i and each host is represented as p_j. We have a single decision variable for the problem denoted by y_{ij}. The value of this decision variable is 1 when the i^{th} VM is allocated to the j^{th} host machine and 0 otherwise. The set of all hosts and all VMs in the system are represented by P and V respectively. In our system, each host machine can be represented by the vector:

$$p_j = (id_j, cpu_j, mem_j, bw_j) \tag{1}$$

where *id* provides an identification number for the host, cpu_j gives the processing power of the host, mem_j gives the amount of memory the host has and bw_j gives the amount of the bandwidth that the host supports.

Each VM is also represented by a similar vector, given by:

$$v_i = (id_i, cpu_i, mem_i, bw_i) \tag{2}$$

where *id* gives the identification of the VM, cpu_i gives the processing power required by the VM, mem_i gives the amount of memory requested by the VM and bw_i gives the amount of bandwidth requested by the VM.

The problem can be formally defined as follows:

Find a mapping from the set of VMs, V, to the set of PMs, P, such that the physical resource utilization is maximized.

In the environment considered, the objective of maximizing physical resource utilization can be decomposed into three objectives:

$$Maximize\left(\frac{v_i^{cpu}}{p_j^{cpu}}, \frac{v_i^{mem}}{p_j^{mem}}, \frac{v_i^{bw}}{p_j^{bw}}\right) \tag{3}$$

subject to the constraints

$$\begin{aligned} &\forall i \sum_{j=1}^{m} y_{ij} = 1 \\ &\forall j \sum_{i=1}^{n} y_{ij} \cdot v_i^{cpu} \leq p_j^{cpu} \\ &\forall j \sum_{i=1}^{n} y_{ij} \cdot v_i^{mem} \leq p_j^{mem} \\ &\forall j \sum_{i=1}^{n} y_{ij} \cdot v_i^{bw} \leq p_j^{bw} \end{aligned} \tag{4}$$

The constraints ensure that each VM is allocated to only one host, though one host may be mapped to more than one VMs. They also ensure that the load on each host machine is not greater than its capacity. We can also define upper and lower thresholds for the utilization on a host. The utilization of j^{th} host machine by i^{th} VM can be given as:

$$p_u^{j(i)} = \begin{cases} \left(\frac{v_i^{cpu}}{p_j^{cpu}} \cdot \frac{v_i^{mem}}{p_j^{mem}} \cdot \frac{v_i^{bw}}{p_j^{bw}}\right) * 100, & \text{if } y_{ij} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The total utilization of host can then be calculated as

$$p_u^j = \textstyle\sum_{i=1}^{n} p_u^{j(i)} \tag{6}$$

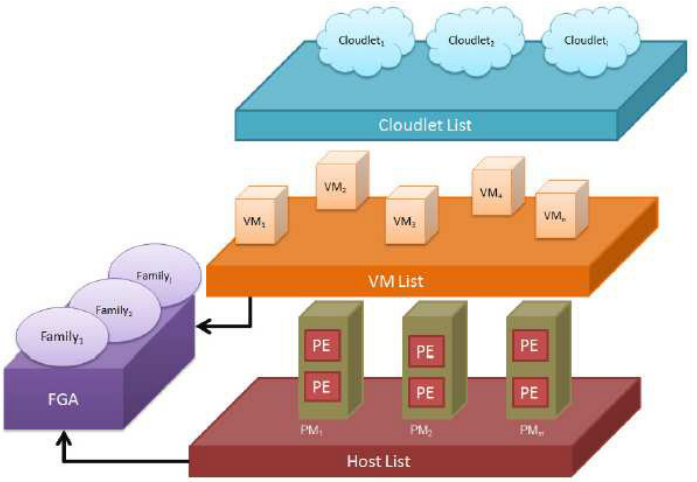


Fig. 1. Architecture of the proposed system integrated into CloudSim.

4. Proposed System

The architecture of the proposed system is as shown in Fig. 1. The Family Genetic Algorithm (FGA) module is integrated into CloudSim. In CloudSim, we have Data centers that comprise of hosts. Each of the hosts has one or more Processing Elements (PE). On these hosts, we have various VMs running. These VMs have one or more cloudlets running on them. In CloudSim, user jobs are directly represented as Cloudlets. The cloudlets have various requirements. The processing power requirement of each Cloudlet is represented using Million Instructions Per Second (MIPS).

In the proposed architecture, the FGA module takes the host list and the VM list and produces an optimal mapping. The FGA module divides the entire processing among the various families that run in parallel in the module.

The Family Genetic Algorithm (FGA) attempts to overcome the limitations of the Genetic Algorithm approaches. According to Jian et al.[16], the major contributing factor towards premature convergence is the mutation operator. The authors attempt to reduce the chances of premature convergence by using a self-adjusting mutation operator. Generally, in GA approaches, the mutation rate, that is, the probability of mutation is static. The value of this parameter of GA is defined at the beginning of the GA and remains constant throughout. As a variation to this traditional GA, Jian et al. vary the rate of mutation. Thus here, the mutation probability is dynamic. It is defined to be dependent on a parameter called population differentia.

Population differentia is a ratio that is used to indicate the rate at which the different individuals differ from each other. This parameter guides the probability of mutation. The use of this self-adjusting probability of mutation ensures that no premature convergence takes place. In their approach the degree at which 2 individuals, say "A" and "B" differ from each other is given by:

$$d(A,B) = \sum_{j=0}^{l-1} A_j \oplus B_j \tag{7}$$

where l gives the length of the chromosome.

Thus, the total rate at which each of the individuals differ from the rest of the population can be defined as:

$$Population\ differentia = \sum_{i=0}^{N}\sum_{j=0}^{N} \frac{d(A_i, B_k)}{(N-1)\cdot(N-1)\cdot l} * 100 \tag{8}$$

where N is the population size.

The outline of the Family Gene Algorithm is described in Algorithm 1. The basic idea in FGA is that we divide the entire population into families. In traditional GA, we take an entire population. The various operators of GA, selection,

crossover and mutation are applied at once to the entire population across all the generations. Researchers have proved that these steps are the most time-consuming steps in GA. In FGA, by dividing the population into families and then processing each of these families in parallel, we attempt to enhance the speed of GA. When employed in a distributed parallel system, the processing of each family may be carried out in parallel, thus greatly reducing the total runtime. This approach was first proposed by Jianhua et al[17]. In their approach the families were constructed considering the neighbouring solutions. Our problem of VM allocation does not define such neighbours. So here, in order to construct the families, we perform simple mutations. The resulting chromosomes which vary, though only slightly, from each other, are placed in the same family. The processing time is further reduced by destroying the families which do not offer any hope of obtaining better individuals. Each family is processed "k" times. If no better individual has been encountered till then, we destroy the family and take the next family. If atleast one better individual has been generated from the processing of the current family, then we continue processing the family for "W" iterations. The values of "k" and "W" are determined through experimental evaluations.

For each individual in the population, we assess the quality of the individual by calculating the fitness value associated with it. In GA, the fitness value is generally a function of the objectives that we take into consideration. In the proposed approach, the objective that we take into consideration is the physical resource utilization. The algorithm that we used to calculate the fitness value of each individual is outlined in Algorithm 2.

An additional precaution has to be taken while employing family gene approach to the VM allocation problem. It should be ensured that all the chromosomes satisfy the constraints. To ensure this, we implement a separate function where each individual is checked for feasibility. In case the individual is found to be infeasible, an attempt is made to transform the infeasible solution into a feasible one. Algorithm 3 takes as input an individual and returns a chromosome representing a feasible assignment.

Algorithm 1 Outline of the Family Gene algorithm

Input: Lists of hosts and VMs
Initialize the list of hosts and Vms.
Initialize the values of parameters of GA and the number of families to be constructed.
Randomly initialize the population.
Compute the fitness values of each chromosome in the population.
Calculate the population differentia.
Perform crossover and mutation.
Select the family heads as the best individuals from the current population.
for all *Family* ∈ *Population* **do**
  repeat
    Perform mutation on the family head and insert mutated chromosome into family.
    Compute the fitness value of the chromosome obtained after mutation.
    if fitness of the mutated chromosome is greater **then**
      add the mutated chromosome to the population.
      Set flag as **true**
    end if
  until family size
  repeat
    Perform crossover and mutation on the current family to get the next generation of the current family.
  until 'k' times if flag=**true** else 'W' times if flag=**false**
  Select the fittest individual from the population to get the best solution.
end for

5. Experimental Analysis and Results

The experimental analysis was done using the CloudSim toolkit, which was developed by Rajkumar Buyya et al.[18] This is an open source tool used by majority of the researchers to simulate the Cloud environment. The toolkit

Algorithm 2 Calculation of the fitness value of each chromosome

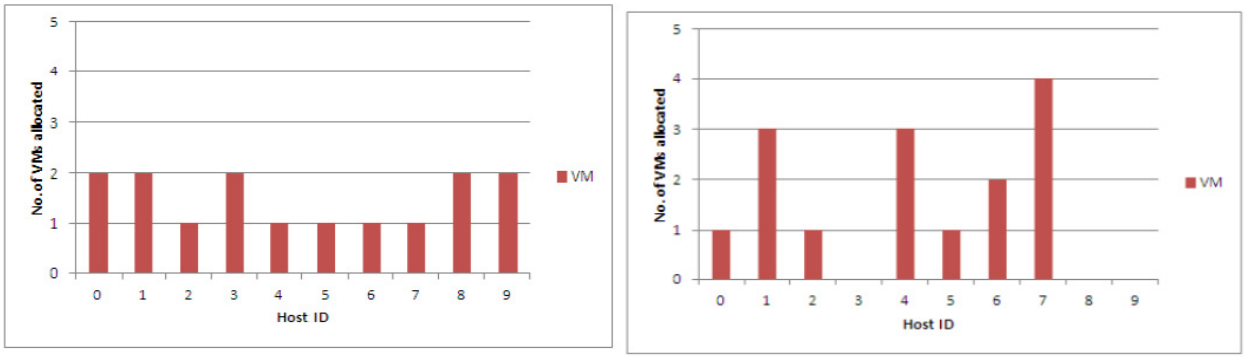
Input: Chromosome
Output: Fitness of the chromosome
  for all $p_j \in P$ **do**
    Initialize utilization values
    for all $v_i \in V$ **do**
      if VM is assigned to current host **then**
        Update the utilization values
      end if
    end for
  end for
  return $\frac{v_i^{cpu}}{p_j^{cpu}} * \frac{v_i^{mem}}{p_j^{mem}} * \frac{v_i^{bw}}{p_j^{bw}}$

Algorithm 3 Check the feasibility of an individual

Input: *Chromosome*
Output: *FeasibleSolution*
  Initialize list of free hosts to contain all hosts
  Initialize the capacities and the number of pes
  for all $v_i \epsilon V$ **do**
    Remove from free host the host assigned in chromosome.
  end for
  for all $p_j \epsilon P$ **do**
    for all $v_i \epsilon V$ **do**
      if v_i *is assigned to* p_j **then**
        Calculate the utilization.
        Update the remaining capacity of the host
      end if
      if the host is overutilized **then**
        Assign any non-free host that can accept the VM.
        if no allocated hosts can accept the VM **then**
          Assign a suitable host from the free hosts.
          Update the capacities of the assigned host.
        end if
      end if
    end for
  end for
  Update the chromosome with the new assignments.
  return Updated chromosome

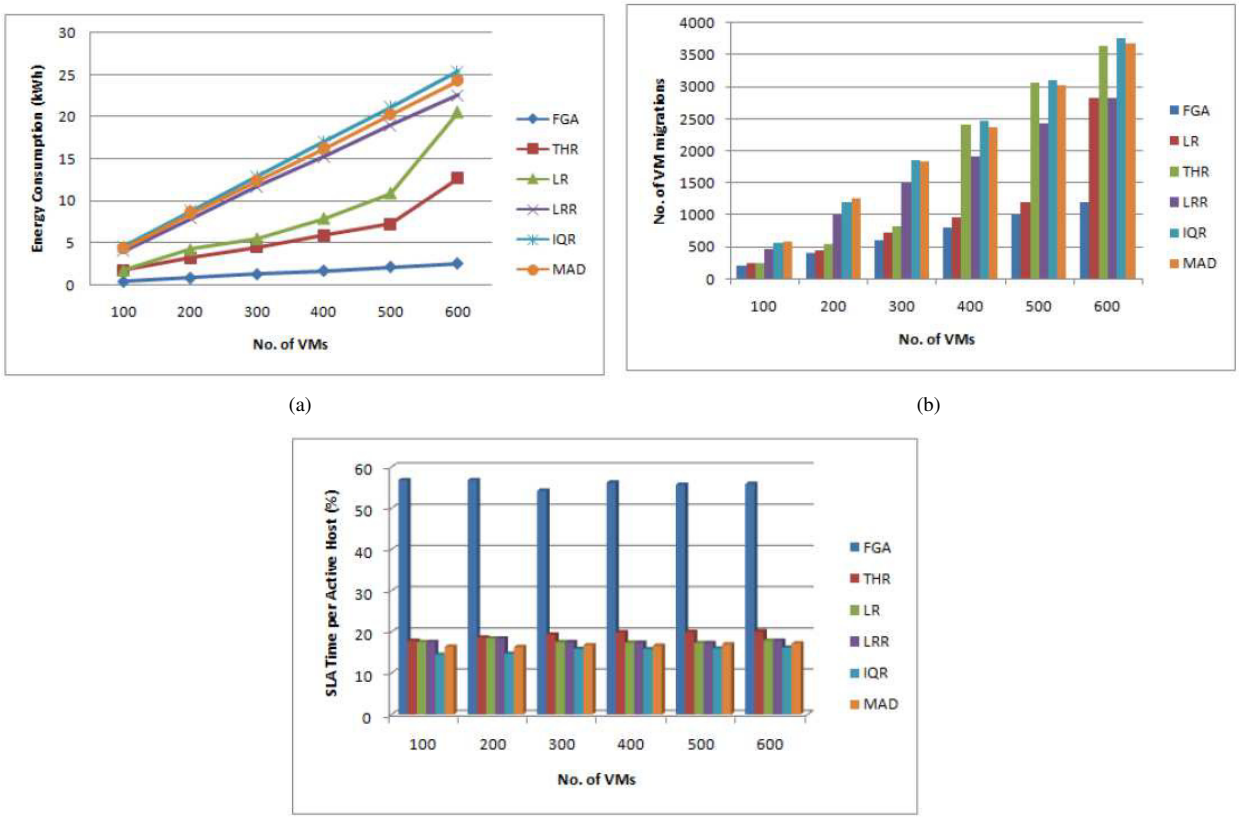
provides simple allocation policies and 6 power-aware allocation policies. The proposed allocation policy using FGA was implemented, run and compared with the existing policies.

Fig. 2(a). shows the placement of hosts by the default allocation policy in CloudSim. Fig. 2(b). gives the placement that results from using the proposed approach. While allocating VMs using the proposed approach the number of hosts in use is reduced. The proposed approach performs the allocation using just the hosts that are required to satisfy the VM requirements. The remaining hosts which are not mapped to any VM may be switched off to further reduce energy consumption. An important parameter that characterizes the performance of an allocation policy is the energy consumption. There is a growing concern nowadays for the increasing power consumption of data centers. Nevertheless, an allocation policy that reduces energy consumption is much more favourable. For our analysis, we allocate varying number of VMs using the proposed approach and the existing approaches. On analysis, it is found



(a) (b)

Fig. 2. (a) the no. of VMs on each host in the initial placement; (b) the number of VMs on each host in the placement by the proposed approach.



(a) (b)

Fig. 3. (a) the energy consumption; (b) the number of VM migrations; (c) the SLA time per active host for the existing and proposed approaches.

that the energy consumption is greatly reduced by allocating VMs using the proposed approach. Fig. 3(a). supports this observation.

For any allocation policy, if the resulting allocation is unable to meet any of the resource requirements, the VMs have to be migrated from the allocated host to some other suitable host. The migration of VMs incurs an overhead on the system. So, an allocation policy that keeps the number of VM migrations at a minimum is preferred.

Fig. 3(b). compares the number of migrations for various approaches for varying number of VMs. It can be observed that the number of migrations is lesser for the proposed approach.

When users submit jobs to be executed in the Cloud environment, they specify certain conditions that should be met by the Cloud service provider. This set of user requirements is called the Service Level Agreement (SLA). The major objective of the Cloud service provider should be to attain a higher level of SLA. A parameter related to SLA that depends on the allocation policy used is the SLA time per host. This parameter gives the time in percentage where each host follows the SLA. Fig. 3(c). shows that the SLA time per active host is greater for the proposed approach, ensuring a higher SLA level in the proposed approach.

In summary, it can be observed that the proposed approach reduces energy consumption and the number of VM migrations and increases the SLA level, while keeping the number of active hosts at a minimal level.

6. Conclusion

The problem of VM Allocation is one of the most important decision problems present in all systems that involve virtualization, such as, Clouds and Grids. The paper proposes an approach to enhance the efficiency of the traditional GA approaches to VM allocation. It has been seen that the energy consumption has been greatly reduced. The number of VM migrations is also reduced, while at the same time increasing the SLA time per host.

The promising results obtained from the proposed approach show that the family genetic algorithm may be employed efficiently in real data centers. As the energy consumption is reduced, it can also be used in green data centers. Though the proposed approach has been tested in the Cloud simulation environment, this approach may be extended to any of the other systems that involve virtualization.

References

1. Barbagallo, D., Di Nitto, E., Dubois, D.J., Mirandola, R.. A bio-inspired algorithm for energy optimization in a self-organizing data center. In: *Self-Organizing Architectures*. Springer; 2010, p. 127–151.
2. Hongwei Chen Lei Xiong, C.W.. Cloud task scheduling simulation via improved ant colony optimization algorithm. *Journal of Convergence Information Technology(JCIT)* 2013;**8**.
3. Wang, C., Wu, Q., Tan, Y., Guo, D., Wu, Q.. Vce-pso: Virtual cloud embedding through a meta-heuristic approach. In: *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*. IEEE; 2013, p. 1908–1915.
4. Dong, Y.S., Xu, G.C., Fu, X.D.. A distributed parallel genetic algorithm of placement strategy for virtual machines deployment on cloud platform. *The Scientific World Journal* 2014;**2**.
5. Esnault, A., Feller, E., Morin, C.. Energy-aware distributed ant colony based virtual machine consolidation in iaas clouds bibliographic study. *Informatics Mathematics (INRIA)* 2012;:1–13.
6. Feller, E., Morin, C., Esnault, A., et al. A case for fully decentralized dynamic vm consolidation in clouds 2012;:26–33.
7. Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R.. Virtual machine consolidation in cloud data centers using aco metaheuristic. In: *Euro-Par 2014 Parallel Processing*. Springer; 2014, p. 306–317.
8. Campegiani, P.. A genetic algorithm to solve the virtual machines resources allocation problem in multi-tier distributed systems. In: *Second International Workshop on Virtualization Performance: Analysis, Characterization, and Tools (VPACT 2009), Boston, Massachusett*. 2009, .
9. Zhong, H., Tao, K., Zhang, X.. An approach to optimized resource scheduling algorithm for open-source cloud systems. In: *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*. IEEE; 2010, p. 124–129.
10. Wilcox, D., McNabb, A., Seppi, K.. Solving virtual machine packing with a reordering grouping genetic algorithm. In: *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE; 2011, p. 362–369.
11. Madhusudhan, B., Sekaran, K.C.. A genetic algorithm approach for virtual machine placement in cloud 2013;.
12. Zhao, J., Ding, Y., Xu, G., Hu, L., Dong, Y., Fu, X.. A location selection policy of live virtual machine migration for power saving and load balancing. *The Scientific World Journal* 2013;.
13. Wu, G., Tang, M., Tian, Y.C., Li, W.. Energy-efficient virtual machine placement in data centers by genetic algorithm. In: *Neural Information Processing*. Springer; 2012, p. 315–323.
14. Quang-Hung, N., Nien, P.D., Nam, N.H., Tuong, N.H., Thoai, N.. A genetic algorithm for power-aware virtual machine allocation in private cloud. In: *Information and Communication Technology*. Springer; 2013, p. 183–191.
15. Tang, M., Pan, S.. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Processing Letters* 2014;:1–11.
16. Jian, Z., Wang, S.. Study on self-adjusting of gene migration genetic algorithm. *Journal of Xi'an Jiaotong University* 2002;**11**.
17. Jianhua, L., Xiangqian, D., Sun'an, W., Qing, Y.. Family genetic algorithms based on gene exchange and its application. *Systems Engineering and Electronics, Journal of* 2006;**17**(4):864–869.
18. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011;**41**(1):23–50.