Performance Analysis of Bio-Inspired Scheduling Algorithms for Cloud Environments

Ali Al Buhussain, Robson E. De Grande, Azzedine Boukerche
PARADISE Research Lab
University of Ottawa
Ottawa, ON. Canada
E-mails: aalbu052@uottawa.ca, rdgrande, boukerch@site.uottawa.ca

***Abstract*—Cloud computing environments mainly focus on the delivery of resources, platforms, and applications as services to users over the Internet. Cloud promises users access to as many resources as they need, making use of an elastic provisioning of resources. The cloud technology has gained popularity in recent years as the new paradigm in the IT industry. The number of users of Cloud services has been increasing steadily, so the need for efficient task scheduling is crucial for maintaining performance. In this particular case, a scheduler is responsible for assigning tasks to virtual machines efficiently; it is expected to adapt to changes along with defined demand. In this paper, we present a comparative performance study on bio-inspired scheduling algorithms: Ant Colony Optimization (ACO) and Honey Bee Optimization (HBO). A networking scheduling algorithm, Random Biased Sampling, is also evaluated. Those algorithms show the ability of self-managing and adapting to changes in the environment. The experimental results have shown that ACO performs better when computation power is set as the objective, and HBO shows better scheduling when the objective mainly relies on costs.**

***Keywords*-Scheduling; Bio-inspired Algorithms; Swarm Optimization.**

I. Introduction

Cloud computing services aim to provide resources with high availability. Those resources are scalable depending on the need of the users. Since flexibility and scalability are the most important properties of cloud environments, making it a highly complex and large distributed environment. These characteristics impose massive challenges on allowing centralized governance. Hence, there is an increasing need to identify distributed solutions that are able to govern the cloud environment through local knowledge. A distributed governance system is expected to be self-organized and able to manage itself [21].

Cloud Environments provides services that can be accessed through the internet [15]. Those services reduce the need for infrastructure and all the management of it by individual IT companies. Cloud also offers software as a service to users and companies. The cloud computing environment is seen as the next step in the IT industry. As a result, cloud computing has been given a lot of attention and effort by many IT initiatives, such as Amazon EC2, Microsoft Azure, and Google Cloud Platform.

In the cloud, the demands for resources change dynamically, and cloud providers are expected to be able to accommodate and react to these changes. Those changes can be set on deadlines for hard real-time applications, cost, which corresponds mostly to re-sizing resources according to dynamic application demands, the load of the cloud environment, and SLA agreements. Virtualization of physical resources provides the necessary dynamicity to manage the resources in the cloud platform [9]. Consequently, scheduling the assignment or task to virtual machines directly impacts on the performance of the Cloud and aids in balancing the distribution of load among the different physical servers. Thus, finding an effective scheduling scheme is crucial to enable Cloud services.

Several scheduling algorithms have been proposed for allocating resources in distributed systems. However, the context of resource management systems for cloud computing restricts the use of such algorithms to a narrower class of solutions. Bio-inspired algorithms, more specifically swarm or gang scheduling algorithms, have shown very suitable for Cloud environments as per results presented in some works. However, the analyses performed in previous works have limited themselves to restricted scenarios.

In this work, we conducted a series of extensive analyses in order to observe the behaviour of selected bio-inspired algorithms against extreme load and large-scale environment conditions. The results of such analyses provide conditions to better understand the scenarios that influence the performance of each algorithm, giving an indicative of better parametric configuration and allowing us to propose enhanced solutions that are able to cover such extreme cases, enabling enhanced overall scheduling efficiency.

The remaining of the paper is structured as follows. Section II provides a brief description of previous works. Section III provides details of the architecture and functioning of the HoneyBee Optimization algorithm. Section IV thoroughly describes AntColony algorithm. Section V presents Random Biased Sampling algorithm. Section VI introduces the experimental scenario and discusses the results obtained.

Finally, Section VII concludes the paper and provides directions for the future work.

II. Related Work

A cloud is seen as a dynamic and heterogeneous system. Scheduling heterogeneous resources in the cloud is considered an NP-hard problem and used to be approached simplistically. For instance, load balancing and scheduling in Amazon EC2 was achieved via replication of instances [21]. Through monitoring, if the load increases beyond a specific threshold, then new instances are instantiated. The distribution of load in Amazon EC2 was previously based on the evaluation of a rule set [21]. However, this approach is not practical for large-scale, complex systems, such as the cloud. Also, the execution of one rule might trigger a cascading process on policy execution [21]. Ultimately, the management of such a large rule set is time-consuming and not feasible, and the rules are static in nature and hence not suitable for dynamic environments.

Based on new constraints, scheduling in clouds is faced as an NP-Hard problem [7], [29], [30], [8], requiring self-regulations for balancing load on entities. This self-organization is possible with the implementation of distributed algorithms that rely on local knowledge. In recent years, nature has influenced many works on seeking out solutions to the increasing scale and complexity of the Cloud system [1]. There are two general directions that are typically followed to solve this issue. The most powerful heuristics used to solve NP-Hard problems are population-based algorithms, or evolutionary optimization, such as fuzzy and neural controllers. Due to its complexity, the scheduling in the cloud is best tackled by the previously mentioned heuristics [22], [26], including genetic algorithms (GA), Particle Swarm optimization (PSO), Ant Colony optimization (ACO), and Honey Bee optimization (HBO).

The first approach is done using GA. Those algorithms are multi-objective, and some works are focused on the cost of running tasks on the Cloud; the work described in [10], for instance, used a GA-based algorithm to classify tasks depending on time and budgetary cost constraints into priorities. Other works focus on the energy consumption, such as the work in [27], which proposes an energy effect scheduling algorithm. Other works cluster objectives, such as the one in [14], which has been developed using a multi-objective scheduling algorithm. Reputation has also been used to introduce memory in the scheduling, such as the work in [19]. However, GA scheduling algorithms are slow for Cloud due the time to converge [17]. Clustering has been used based on memory consumption and computation requirements of tasks as in [31]. A GA scheduler that scans the entire job queue for decisions was proposed in [6] to minimize the makespan of the tasks only. Thus, distributed, self-organized, and self-managed algorithms are needed to solve the cloud-scheduling problem.

Secondly, particle swarm optimization (PSO) is simple and fast as it only uses two aspects: velocity and position. PSO is the algorithm with the fastest convergence when compared to GA and ACO [30]. The encoding in the work of [18] was similar to the GA in which each task is associated with an integer holding the resource id. The PSO takes into account only the cost, both data transmission and computational, as optimization objective. This work is similar to [28] in terms of the optimization objective but they differ in form [18] as the PSO is discrete not continuous [30]. The optimization factors were then further extended in [3] to include the makespan, cost, reliability. The only problem with [3] is that there is no dependency between the optimization factors. The study done by [23] used encoded the scheduling by providing a rounded integer specifying the index of the resource assign to each task. The optimization objective was to reduce cost and meet deadlines. The index does not represent enough guidance or does not show the real features of the resources chosen [30]. Hence, [12] purposed to redefine what the index of resources should mean in PSO by using a renumbering based on the price of the resource and the ability to reorder. Also [11] extended the [12] by using a include many other scheduling objectives.

Lastly, Using the behaviour of ants or bees when seeking food provides a different perspective on the scheduling problem solving. Those algorithms are distributed, and self-managed, and thus they are studied thoroughly in this paper. [30] stated that ACO is better that GA in terms of always finding a solution to the scheduling in the cloud. The HBO was also chosen because of the speed in which it converge. HBO has a similar quickness to PSO but it is understudied in the clouds and hence, it was chosen. The work conducted in [1] provided a study on several swarm or gang scheduling algorithms. This work stated that schedules are mostly bio-inspired behaviours like the ants, honeybee, and particle swarm optimization. It shows that the gang schedules provide a viable solution to schedule tasks in the cloud environment.

There exist some other schedulers that are inspired by the traditional grid schedulers, which are adapted to the cloud environment. However, the traditional algorithms have limited success in scheduling cloud tasks. An improved version of the Max-Min algorithm for task scheduling has been proposed in [4]. This mechanism depended on not only assigning the maximum execution to the least loaded computing node but also achieving lower make-span. Also, a priority based scheduling algorithm has been introduced in [25]. The proposed scheduler evaluated a priority based on the cost of the execution of the task among other aspects. Then, the tasks were divided into three priority groups; the scheduler achieved task execution with lower cost.

The current situation of schedulers used for cloud environment motivated our work on deeply analyzing bio-inspired scheduling algorithm. The next sections provide more details

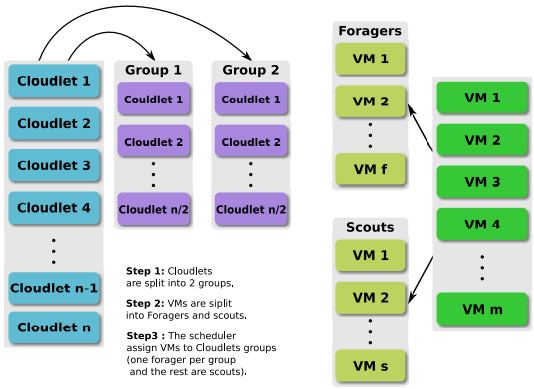


Figure 1: Honey Bee Structure.

overview of such algorithms for better understanding them in the context of our experimental analyses.

III. HoneyBee

The basic elements in HoneyBee scheduling algorithm are bees. They find the most profitable source and exploit it. Foragers are also considered by shifting in quality or profit in the nectar sources [24]. HoneyBee provides a self-managed and self-organized solution, by essence decentralized, to the scheduling problem [16]. Such characteristics make HBO along with the speed of convergence a strong candidate for cloud scheduling.

HoneyBee scheduling algorithm is basically divided into two parts, as delimited in Algorithm 1. The first part consists in the foraging behaviour of the bees as they look for food sources. The second part is the scouting where the bees start their search for the best food source brought by the foragers. The HoneyBee procedure is described according to the following elements:

1) The number of foraging VMs (n) is equal to the number of Datacenters (DCs) available. The choice of foraging VMs placement in DCs is random;
2) The DCs have its own characteristics (dch). Those characteristics (RAM, Storage, Bandwidth) help the foraging VMs to find the best food source for a cloudlet with a specific execution time;
3) The above-mentioned dch and the execution time are used to evaluate the fitness function. The outcome of the fitness value will then be the deciding factor by which we chose to run the cloudlet on which DC;
4) The DC with the highest fitness value, or the lowest cost as defined in Equation 1, receives a percentage of the tasks. The remaining subset of the tasks will be executed on the other DCs by repeating step 4 and excluding the chosen DCBest.

$$DC_{Cost}^{i,j} = (Size_i + M_i + Bw_i) \times (T_{CL_j}), i = 1...N, j = 1...M \quad (1)$$

$$Size_i = dch_{CPS} \times size_{VM_i}, i = 1...N \quad (2)$$

$$M_i = dch_{CPR} \times RAM_{VM_i}, i = 1...N \quad (3)$$

Algorithm 1 HoneyBee VM Assignment Algorithm

Require: $Cloudlet_{list}, VM_{list}, Datacenter_{list}, fac_{LB}$
1: $Groups(q) \leftarrow divide(Cloudlet_{list})$
2: **for** $i = 1$ to q **do**
3: $\quad length_i \leftarrow lenghtOfGroup_K(Groups_i)$
4: **end for**
5: **for** $k = 1$ to q **do**
6: $\quad CloudLet_L \leftarrow max(Groups_k$
7: $\quad$ **while** $Groups_k \geq \{Groups_i | i = 1..q$ and $i \neq k\}$ **do**
8: $\quad\quad$ **for** $s = 1$ to n **do**
9: $\quad\quad\quad Datacenter_s \leftarrow select(Datacenter_{list})$ // as in Eq 1
10: $\quad\quad\quad$ **if** $fac_{LB} \leq VMsAssigned(Datacenter)$ **then**
11: $\quad\quad\quad\quad assign(Cloudlet_L, Datacenter_s(VM_{leastLoad}))$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad assign(Cloudlet_L, Datacenter_{i \neq s}(VM_{leastLoad}))$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad\quad decrement(length_k)$
16: $\quad\quad$ **end for**
17: $\quad$ **end while**
18: **end for**

$$BW_i = dch_{CPB} \times Bw_{VM_i}, i = 1...N \quad (4)$$

To simulate the honey bee for a cloud environment, Figure 1 shows that Cloudlets are split into two groups forming a food source. Then, the VMs are split into foragers and scouts. Each forager is assigned to a Cloudlet group where it is responsible for recruiting scouts VMs to help execute the Tasks. Each scout chooses a task with certain probability as delimited in Equation 1. This equation is formed by a sum of the size, quantity of memory, and bandwidth, respectively represented by Equations 2, 3, and 4 where the parameters delimit them are listed in Table I.

Table I: HBO Parameters: Algorithm and Equations

Parameter	Values
T_{CL_j}	The cLength of the $Cloudlet_j$
$Size_i$	The cost of storage used by Vm_i
dch_{CPS}	The Cost of storage of $Datacenter_i$
$size_{VM_i}$	The storage required by VM_i
M_i	The cost of RAM to execute $Cloudlet_j$ by VM_i
dch_{CPR}	Cost of RAM for executing $Cloudlet_j$ by VM_i
RAM_{VM_i}	The RAM required by VM_i
BW_i	Cost of Bandwidth for executing $Cloudlet_j$ by VM_i
dch_{CPB}	$Datacnter_i$ cost per bandwidth.
Bw_{VM_i}	The needed bandwidth consumed by VM_i

IV. Ant Colony

The Ant Colony Optimization (ACO) uses the behaviour of real ants in foraging for food to implement a solution for the cloud task scheduling. The ants leave the nest to search for food sources (VMs) in random. Then they evaluate the quality of the food source and carry it back to the nest. The ants leave a chemical trail on the ground. The strength of that chemical trail depends on the quality of the food source found [5]. Researchers used ACO to solve NP-hard problems such as travelling salesman problem, graph colouring problem, vehicle routing problem, and scheduling problem. In the context of cloud computing, ACO is used to find the optimal way to schedule tasks to VMs [5], [13].

In ACO when ants search for food first, ants search randomly for food sources and once one is found, an ant

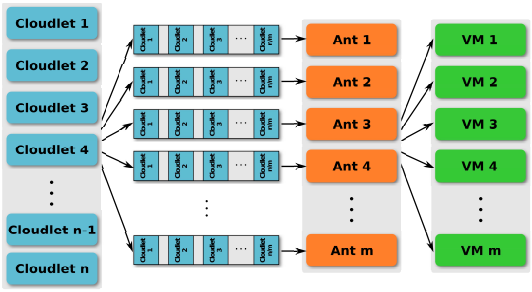


Figure 2: Ant Colony Architecture.

leaves a chemical trail called pheromone leading to that food source. Then, other ants are attracted to that specific food source by following the pheromone trail. This process continues until ants find the shortest path leading to a specific food source by accumulating huge amounts of pheromone on the shortest path leading to the food source [5]. The ACO can be applied to solve complex combination problems if the following elements are properly delimited for the algorithm:

- Problem statement: In this algorithm, ants find the optimal solution to the cloud-scheduling problem by moving from town to town (VM to VM) to choose the best VM to the Cloudlets. Ants move from At the start of the simulations, ants are placed randomly at different VMs with initial pheromone trail $\tau(0)$ as in Equation 5.
- Heuristic desirability η: the inverse of the expected execution time is used.
- Constraint satisfaction method: each ant is only allowed to visit a VM once to minimize scheduling time
- Pheromone-updating rule: each ant deposit a pheromone the depends on the quality of the solution. The pheromone is updated according to Equations 6-11.
- Probabilistic transition rule: the ant related to $tabu_k$ is updated by adding the visited VM in the initial step. Afterwards, Ant_k selects the next VM_j to execute $Cloudlet_i$ based on the probability defined in Equation 5.

$$\rho_{i,j}^{k} = \begin{cases} \left(\frac{[\tau_{i,j}(t)^{\alpha}] \times [\eta_{i,j}]^{\beta}}{\sum_s Allowed_k [\tau_{i,s}](t)^{\alpha} \times [\eta_{i,s}]^{\beta}} \right) & , j \epsilon Allowed_k \\ 0 & , \text{Otherwise} \end{cases} \quad (5)$$

- $\tau_{i,j}$(t) : is the pheromone concentration between task i and VM j.
- $allowed_k$: keeps track of the VMs that Ant_k can use at all times.
- $\eta_{i,j}$: is the heuristic value calculated in Eq(3) where $\eta_{i,j} = \frac{1}{d_{i,j}}$.

$$d_{i,j} = \left(\frac{TL\_Task_j}{Pe\_num_j \times Pe\_mips_j} + \frac{InFileSize}{VM\_bw_j} \right) \quad (6)$$

- α and β : to choose the relative weight of between the pheromone concentration and the heuristic value.

Equations 7, 8, 9, 10, and 11 are used to update the pheromone concentration. Initially each ant is placed on a VM randomly and pheromone value of $\tau(0)$ is given to all edges, as described in Algorithm 2.

Table II: ACO Parameters

ACO Parameter	Values
$Ants$	50
α	0.01
β	0.99
ρ	0.4
Q	100

Algorithm 2 AntColony VM Assignment Algorithm

Require: $\alpha, \beta, max_{iterations}, Cloudlet_{list}, VM_{list}$
for i in $Cloudlet_{list}$ **and** k in VM_{list} **do**
  $pair_{Cloudleti,VM_k} \leftarrow \tau_{i,j}(0) = C$ // pheromone(C)
end for
$VM_k \leftarrow Ant_j \leftarrow randomPick(Ant_{pool})$
$Ant_j^{tabu} \leftarrow add(VM_k)$
while NOT done **do**
  for $k = 1$ to m **do**
    $VM_s \leftarrow select(Ant_k, VM_{list}, Cloudlet_{list})$ // as in Eq 5
    $Ant_j^{tabu} \leftarrow add(VM_s)$
  end for
  for $k = 1$ to m **do**
    $L_k \leftarrow calculate()$ // as in Eq 8
  end for
  $\tau_{i,j} \leftarrow update()$ // as in Eq 9
  $pheromone_{global} \leftarrow update()$ // as in Eq 11
  $increment(iterations)$
end while

$$\Delta\tau_{i,j}^{k}(t) = \begin{cases} \left(\frac{Q}{L_k(t)} \right) & , i, j \epsilon T_k(t) \\ 0 & , Otherwise \end{cases} \quad (7)$$

$$L_k(t) = argmax_{j \epsilon J} sum_{j \epsilon IJ}(d_{ij}) \quad (8)$$

$$\tau_{ij}^{k}(t) = (1 - \rho)\tau_{ij}^{k}(t) + \Delta\tau_{ij}^{k}(t) \quad (9)$$

$$\Delta\tau_{ij}^{k}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (10)$$

$$\tau_{ij}^{k}(t) = \tau_{ij}^{k}(t) + \left(\frac{Q}{L_k(t)} \right) if(i, j) \epsilon T_k(t) \quad (11)$$

$L_k(t)$ is the length of the current best tour done by the ants in the current iteration as in Equation 8. The local Pheromone value is updated by Equation 9 where ρ is the decay factor of the pheromone deposited before. Finally the global pheromone value is updated by Equation 11. Multiple values were tested, and the best parameters were chosen, as listed in Table II.

The diagram represented in Figure 2 illustrates in general terms the whole process of combining Cloudlets to VMs, which is described in Algorithm 2. The scheduler firstly creates the ants and then distribute the Cloudlets to each ant. Then, the ants evaluate the VMs based on the previously mentioned equations to choose the best VM. Finally, the tabu of each ant is updated with the chosen VM and the algorithm is repeated in the following iteration.

Algorithm 3 Random Biased Sampling VM Assignment Algorithm

Require: $Cloudlet_{list} and VM_{list}$
1: **for** $k = 1$ to n **do**
2: $Groups(q) \leftarrow divide(VM_{list}, groupSize(number(r)))$
3: **end for**
4: **for** $k = 1$ to q **do**
5: $Group_k \leftarrow ascending(WIL)$
6: **end for**
7: **for all** $Cloudlet_i$ in $Cloudlet_{list}$ **do**
8: $Cloudlet_i \leftarrow random(WIL)$
9: **end for**
10: **for** $k = 1$ to m **and** i to q **do**
11: **if** $Cloudlet_k.WIL \geq Group_i.WIL$ **then**
12: $Group_i \leftarrow Cloudlet_k$
13: **else**
14: $increment(Clouldet_k.WIL, 1)$
15: $Group_{i+1} \leftarrow Cloudlet_k$
16: **end if**
17: **end for**

V. Random Biased Sampling (RBS)

The Random Biased Sampling (RBS) algorithm provides a way to construct a network of resources (VMs). Those networked resources are then divided into groups. The groups are given a degree. This degree on an average of all resource groups must be the same achieving balanced scheduling. The created network of resources is self-organized and able to distribute the tasks based on only local knowledge. Thus, the Random Sampling algorithm represents a viable solution to the cloud task-scheduling problem [20].

RBS constructs a graph of resource each with a node in degree (NID) and a walk length threshold (υ). The tasks coming into the servers have an associated walk in length (ω) that is used to schedule those tasks to the appropriate resources. The NID value varies depending on the number of free resources on a given node. When a $task_i$ comes into the servers, $node_k$ will only executes a task if the execution test is fulfilled ($\omega \geq \upsilon$). If the previous condition is not satisfied, ω is incremented by one and sent to the other nodes and the execution test is applied again. Algorithm 3 represent the general procedure of RBS in a cloud environment. The following steps shortly describe the functioning of the algorithm:

- Step1: The VMs are split into n groups. Each group contains an equal number of VMs;
- Step2: Each Group is assigned walk length threshold (υ) and an NID. The NID is equal to the number of free VMs in that group;
- Step3: Each Cloudlet is given a random ω;
- Step4: The execution test is performed to assign $Cloudlet_i$ to $group_j$;
- Step5: The NID of groupj is reduced by one;
- Step6: The assignment inside the VMs groups is done in a cyclic way;
- Step7: Repeat starting from step 3.

Figure 3 depicts the match-making of Cloudlets to VMS of the algorithm on a simple diagram. In the figure, the RBS scheduler divides the VMs available into two groups. Then, the created VM groups are assigned a walk in length (WIL)

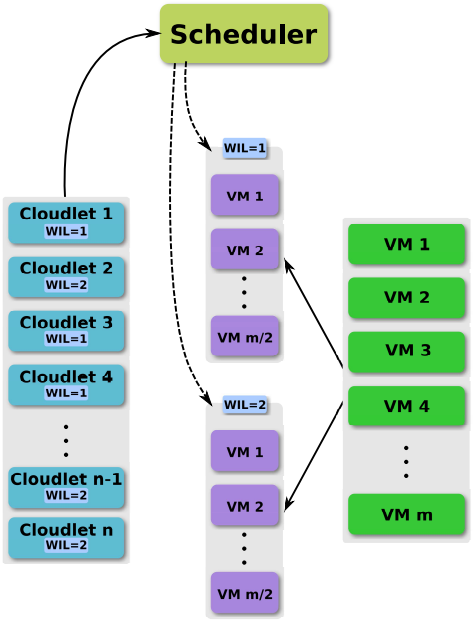


Figure 3: Random Biased Sampling Match-Making.

values (WIL = 1 .. n). Following this step, the Cloudlets are given a random WIL value between the available WIL values assigned to the VM groups. The RBS assigns a Cloudlet to a VM group if the Cloudlet WIL is equal or greater to the WIL of the VM group. If this condition is not applied, then RBS scheduler will increase the Cloudlet WIL in incremented by one and the algorithm is run again.

VI. Experimental Analysis

A series of experiments have been conducted in order to evaluate the performance of AntColony, HoneyBees, and Random Biased scheduling algorithms on a set of varied setup scenarios. The scenarios were configured with parameters that stretched over extreme situations in order to stress the algorithms and identify their limits on providing the proper allocation of virtual resources. Simulations have been conducted in order for this analysis to be conducted on extremely large scenarios; particularly, CloudSim was used as the simulator and worked as the experimental testbed.

CloudSim emerged from the need to have a cloud simulation environment that would extend the traditional distributed systems simulators (Grid and Network) [2]. Moreover, testing on real cloud environment is costly and imposes several challenges, which considerably increase the complexity in conducting large-scale experiments; several factors drive such challenges, such as (i) variation in the clouds demand, supply pattern, system size, and resources; (ii) the heterogeneous characteristics of user and QoS requirements in these dynamic environments; and (iii) variations in applications performance, dynamic, and scaling requirements. Moreover, a simulator is crucial in providing controlled scenarios in which results are reproducible over the most diverse combination of setup parameters.

A. Baseline Test

In the experiment performed in this work, the default scheduler in CloudlSim is used as the base test to compare with the studied algorithms. The algorithm of this Base Test is a simple scheduler that assigns Cloudlets to VMs in a cyclic matter. For instance, in an attempt to allocate a set of virtual machines (vm1, vm2) for a set of Cloudlets (c1,c2,...,cn), the Base Test assigns vm1 to c1, vm2 to c2, vm1 to c3 and so forth until all Cloudlets are assigned, showing an equally distributed of load of the virtual resources. This scheduler represents the best solution in a homogeneous setup and, for this reason, is chosen to compare with the other solutions.

B. Simulation Scenarios

Two scenarios have been used to analyze the bio-inspired schedulers fully and thoroughly. The first scenario comprises a homogeneous setup of physical resources that receives a homogeneous load, Cloudlets that impose the same amount of workload. The use of this scenario aims at testing the suggested algorithms against a base test, which is expected to be the optimum solution in this type of homogeneous setup. On the other hand, the second scenario represents a more realistic cloud environment, in which heterogeneous resources and load are most likely to exist.

The homogeneous environment scenario was conducted to show that even in the worst case conditions in which no scheduler is needed, the bio-inspired algorithms are expected to converge to the optimal scheduling performance, and the only difference in this particular case resides on the time each scheduler takes to produce the assignment of load, which shows that the bio-inspired scheduling requires significantly more time that the base test.

Tables III and IV show the experimental setup used in the homogeneous scenario respectively for the environment, VMs, and the load, Cloudlets. The experiment was conducted with the number of VMs ranging from 1000 to 100000 and 1000000 Cloudlets. There is a focus on computational performance on this analysis, so negligible is observed on the execution of the Cloudlets during the simulation; thus, the default network topology provided by CloudSim is used when running the experiments. Several different aspects have been considered in the analysis in order to determine effectiveness and efficiency of the algorithms. The first measurement observed in the experiments consists in the scheduling time of the selected scheduling algorithm. The second measurement criteria involve the execution time that the Cloudlets generated in order to complete their work. The third measurement considers the execution time imbalance which gives a sense of the balancing ratio of the load during its execution in the simulator.

Since all elements in this scenario present the same characteristics and capabilities, it represents the worst case for the scheduling algorithms, and the base test is necessarily the optimal solution for any time to scheduling setup due to equally assign Cloudlets to the virtual resources in a cyclic fashion.

Table III: VM Characteristics in the Homogeneous Setup

VM characteristics	Values
vmMips	1000
vmSize	5000
vmRam	512
vmBw	500
vmPesNumber	1

Where:

- vmMips: million instructions per second.
- vmSize: the size of the virtual machine in MB.
- vmRam: the RAM of the virtual machine.
- vmBw: the bandwidth of the virtual machine.
- vmPesNumber: the number of processing elements of the virtual machine.

Table IV: Cloudlet Parameters in the Homogeneous Setup

Cloudlet characteristics	Values
cLength	250
cFileSize	300
cOutputSize	300
cPesNumber	1

Where:

- cLength: The required MIPS for the cloudlet.
- cFileSize: The required memory for the cloudlet.
- cOutputSize: The size of the output file of the cloudlet.
- cPesNumber: The required processing elements number for the cloudlet.

The heterogeneous environment scenario was implemented to mimic real cloud environment where task and virtual machines are not similar. In other words, different workloads were submitted to virtual machines that also have different capabilities. For the setup on this scenario, a different range of parameters was used, restricting to smaller dimensions; however, such restrictions did not impact on the results obtained. Thus, the number of virtual machines was reduced to 50 and the number of Cloudlets was reduced to 5000. Tables V, VII, and VI list the characteristics of the VMs, Datacenters, and the tasks, respectively.

Table V: VM Characteristics in the Heterogeneous Setup

Heterogeneous VM characteristics	Values
vmMips	500-4000
vmSize	5000
vmRam	512
vmBw	500
vmPesNumber	1

C. Metrics Used in the Performance Analysis

Three metrics were used to provide a overview on the performance of the algorithms on the experiments. Each of the metric is detailed in the following subsections.

Table VI: Cloudlet Parameters in the Heterogeneous Setup

Heterogeneous Cloudlet characteristics	Values
cLength	1000-20000
cFileSize	300
cOutputSize	300
cPesNumber	1

Table VII: Datacenter Values in the Heterogeneous Setup

Datacenter characteristics	Values
CostPerMemeory	0.05-0.01
CostPerStorage	0.004-0.001
CostPerBandwidth	0.05-0.01
CostPerPrcessing	3

1) Scheduling Time: The scheduling time of the scheduling algorithms is obtained by retrieving the start-up time in which the algorithm has been triggered and the final time when the algorithm returned with the solution of a specific combination setup. Due to the large scale of the setup environments used in our experiments, the unit of the scheduling time is hour even though the precision of the retrieved metrics are on the scale of milliseconds. The times for the ACO, HoneyBee, and RBS are roughly the same. They are larger that the base test as they require more computation to schedule the Cloudlets to the VMs.

2) Simulation Time: The simulation time corresponds to the maximum overall time that the tasks took to complete the execution. As described in Equation 12, this metric consists in the difference between earliest start execution time of a Cloudlet and finish time of the latest Cloudlet to end its execution. Please note that this metric unit corresponds to milliseconds of simulation wall clock time.

$$T_{sim} = T_{maxFinishTime} - T_{minStartTime} \tag{12}$$

Where:

- T_{sim} : simulation time of the Cloudlets.
- $T_{maxFinishTime}$: maximum finish time of the Cloudlets.
- $T_{minStartTime}$: minimum start time of the Cloudlets.

Through this simulation time metric, we can determine the effectiveness of the scheduling algorithms.

3) Time Imbalance: The time imbalance provides an overview on the fairness of the Cloudlets execution on VMs. In summary, this metric represent the variation in the execution time of the tasks. Equation 13 details how time imbalance values are obtained based on the execution times of the Cloudlets.

$$T_{im} = \frac{[T_{max}] - [T_{min}]}{T_{avg}} \tag{13}$$

Where:

- T_{im} : time imbalance.
- T_{max} : maximum execution time of Cloudlets.
- T_{min} : minimum execution time of Cloudlets.
- T_{avg} : average execution time of Cloudlets.

4) Processing Cost: The processing cost of the Cloudlets is defined in terms of the assigned virtual machine Vm_i uses of the resources provided by the datacenter on which it executes. Those resources are defined as the following characteristics:

- Cost of Memory : the memory used by Vm_i in its datacenter;
- Cost of Bandwidth : the amount of bandwidth consumed by Vm_i;
- Cost of MIPS : the cost of computation (MIPS) to execute the task.

D. Experimental Results

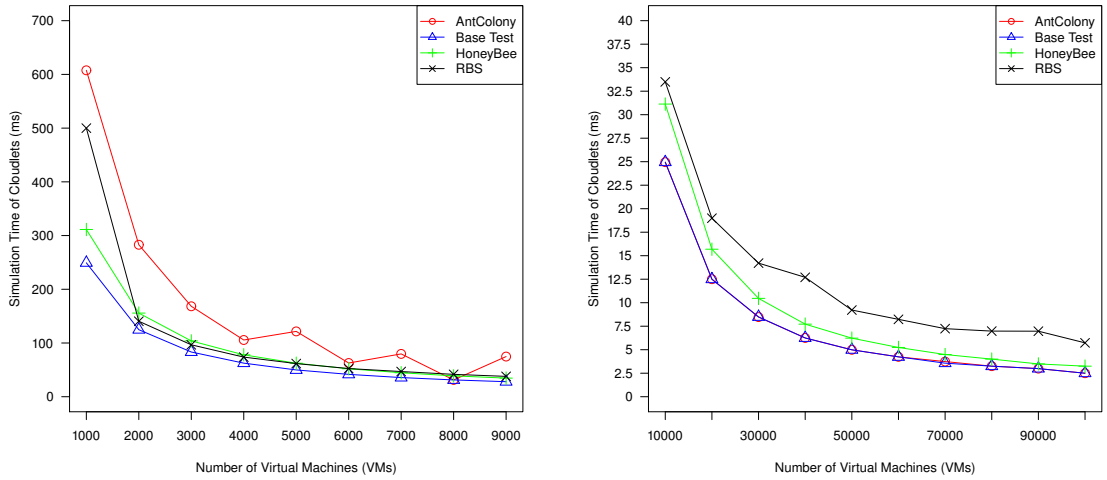
The results obtained in the experiments are presented and discussed separately in two sections: homogeneous scenario and heterogeneous scenario.

1) Homogeneous Scenario: In Figure 4, the simulation shows that the performance in terms of Cloudlet simulation time is similar to the optimum scheduler (base test). As the number of virtual machines increases the simulation time of the Cloudlets decreases. The significance of this figure is that it shows that even in the worst case scenario, the algorithms behave closely to the Base test. The ACO started the worst performance but as the number of virtual machines grows it caught up to the optimal solution. The HBO began and ended the closest of the algorithm to the base test while the RBS started close to the best solution and ended the worst due to the randomness in assigning tasks a WIL value.

In Figure 5, the scheduling time for the base test is significantly less that the other algorithms. The reason behind the base test having the shortest scheduling time is that it does not have any computation to make the scheduling decision; it simply assigns the Cloudlet to the next virtual machine cyclically. The rest of the schedulers (ACO, HBO, And RBS) require computations to assign the Cloudlets so take a longer time to make a scheduling decision. Also, the size of the test makes the scheduling time difference significant (1000000 Cloudlets and up to 100000 VM to choose from).

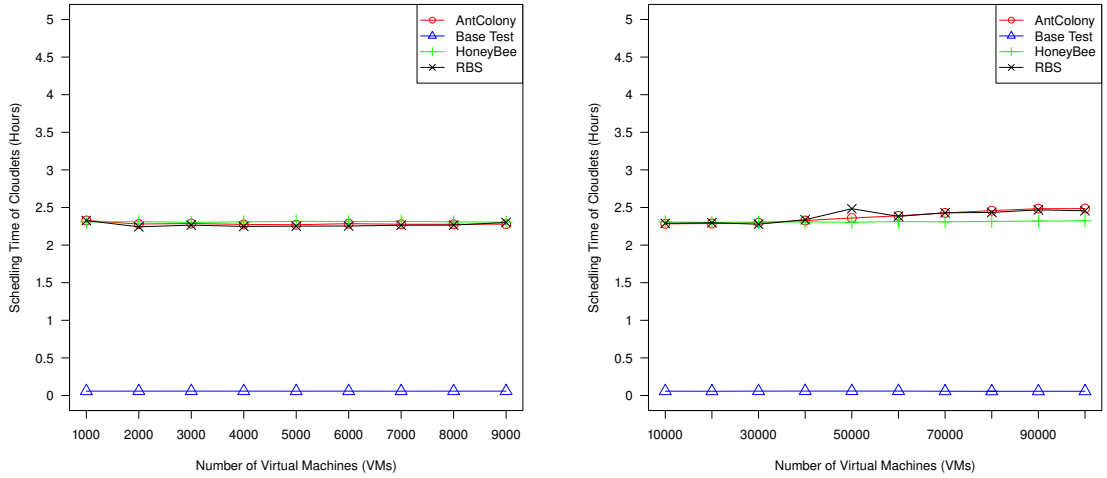
2) Heterogeneous Scenario: In this scenario, the number of virtual machines is reduced significantly from the homogeneous scenario; the test used 50 virtual machines and up to 1000 Cloudlets. The scenario was used to evaluated the algorithms according to the same metrics described previously.

In Figure 6a, the obtained simulation times are shown for an increasing number of virtual machines. In this experiment, it can be noticed that ACO presents the best performance as the Cloudlets finished the fastest. The ACO variables were chosen to best meet the computation speed. The base test performance is steady as it does not consider any characteristics of the resources or tasks in its selections. The HBO main purpose is to save money, and this is the main reason it does not present a performance as good as the ACO; however, it still shows slightly better performance than the base test. The RBS performance is almost the same as



(a) (b)

Figure 4: Simulation Time of the Homogeneous Scenarios.



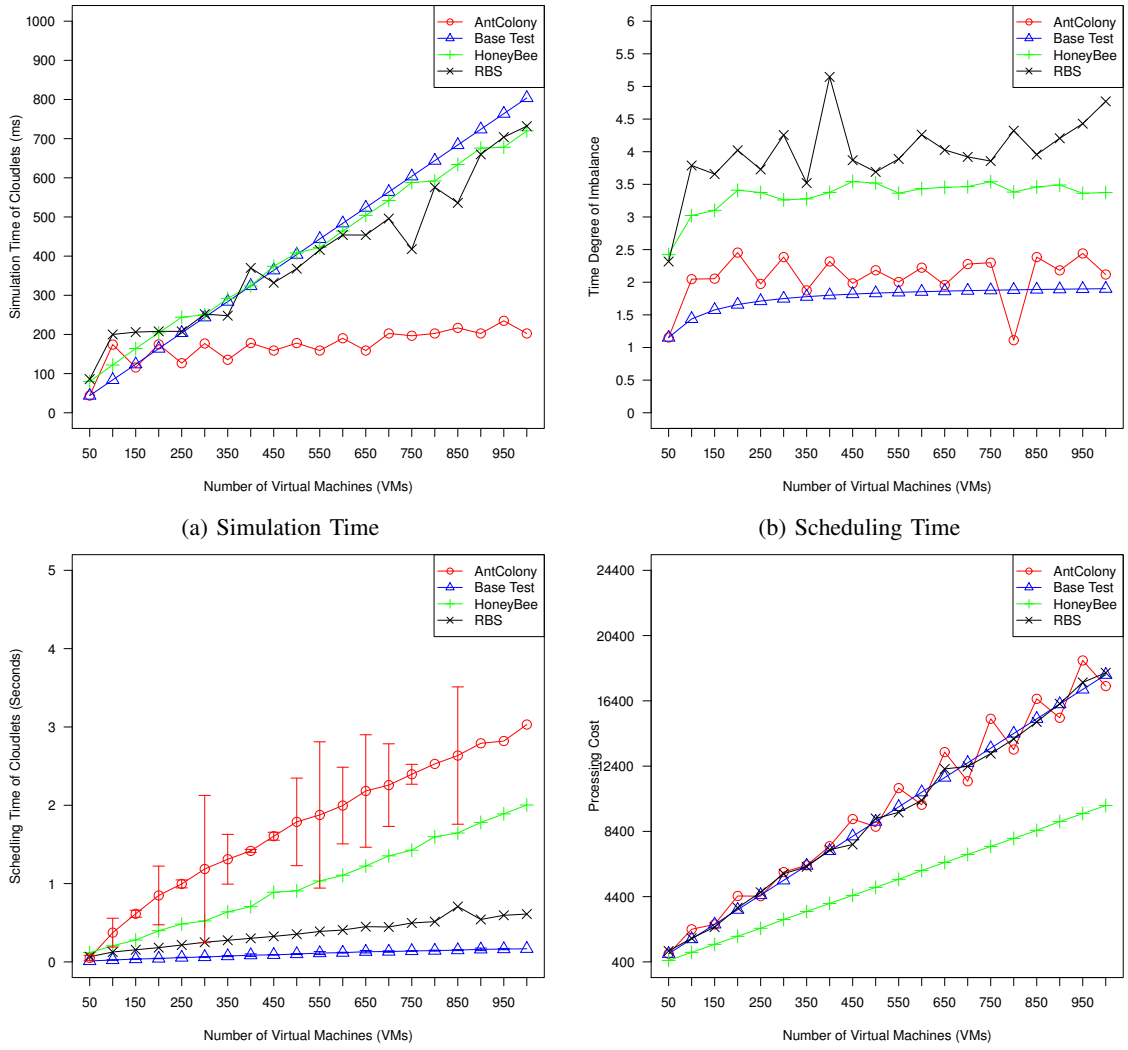
(a) (b)

Figure 5: Scheduling Time for the Homogeneous Scenarios.

the base test and HBO. The fluctuation in RBS performance are due to the randomness in assigning WIL to Cloudlets, and that caused only some of the virtual machines to be available and not all of them.

Figure 6b shows the time each scheduler takes to finish assigning virtual machines to Cloudlets. The time was recorded in seconds. The figure describes that the base test presented the shortest scheduling time because of its simplicity and no computation needed to make the scheduling choices. The second best scheduling times is the RBS due to its simple assignment decisions as it only checks for the WIL. The third best scheduling is the HBO as it only identifies the least cost of the virtual machines to run the Cloudlets. ACO is the one that takes the longest to provide a combination solution as it needs to perform more complex, iterative computations and multiple searches to schedule the Cloudlets.

As shown in Figure 6c, in measuring the time degree imbalance to find the best load distribution of Cloudlets among virtual machines, the base test has the edge in load distribution due to its nature of scheduling as it gives each of the virtual machines an equal number of Cloudlets; hence, the execution times of those Cloudlets are close to each other. However, it does not mean that the Cloudlets finish fast as proven in Figure 6a. RBS is the second best in load distribution as it is used as a load balancer in networking. HBO is better that ACO because of the load balancing factor it used to the allocating resources. ACO needs to choose the



(a) Simulation Time

(b) Scheduling Time

(c) Degree of Time Imbalance

(d) Processing Costs

Figure 6: Performance Analyses for the Heterogeneous Scenarios.

best virtual machine to execute the Cloudlet, and it does not consider load when making that selection; this leads it to show the highest imbalance degree.

To calculate a cost of Cloudlet processing, we took into account the bandwidth, memory, and MIPS needed. As depicted in Figure 6d, HBO presents the best price value as it does consider the lowest processing cost when scheduling even though the load balancing factor shows an effect on the decision making but it is minimal. The rest of the algorithms show close values as none of the them contain a factor cost when scheduling decisions are made.

VII. Conclusion

In this paper, we conducted an extensive set of experimental analyzes on bio-inspired scheduling algorithms, HoneyBee and Ant Colony, as well as Random Biased Sampling algorithm. The purpose of the analysis was to stress the algorithms at its extremes to determine the conditions in which they would perform best in a Cloud environment. Our experiments were divided into two different scenarios: homogeneous and heterogeneous. In the homogeneous scenario, the base test performed better than the rest of the schedulers since no advanced decision-making was necessary. In that scenario, the simulation environment was large and time-consuming. In the second scenario, the heterogeneous, the schedulers were tested in heterogeneous setups with different tasks sizes. In this test the ACO showed the best simulation time and HBO had the minimal processing cost.

As future work, we intend to explore the drawbacks that have been shown in the results. Based on such issues, we will propose a hybrid scheduling algorithm in which the conditions of the system and environment against pre-selected requirements function as key elements to select a specific behavior of the scheduling algorithm. In order to to obtain such approach, a modular solution will be designed.

REFERENCES

[1] S. Bilgaiyan, S. Sagnika, and M. Das. An analysis of task scheduling in cloud computing using evolutionary and swarm-based algorithms. *Int. Journal of Computer Applications*, 89(2), 2014.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[3] W.-N. Chen and J. Zhang. A set-based discrete pso for cloud workflow scheduling with user-defined qos constraints. In *Proc. of the IEEE Int. Conference on Systems, Man, and Cybernetics*, pages 773–778, 2012.

[4] S. Devipriya and C. Ramesh. Improved max-min heuristic model for task scheduling in cloud. In *Proc. of the Int. Conference on Green Computing, Communication and Conservation of Energy*, pages 883–888, 2013.

[5] M. Dorigoa and C. Blumb. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278, 2005.

[6] Y. Ge and G. Wei. Ga-based task scheduler for the cloud computing systems. In *Proc. of the Int. Conference on Web Information Systems and Mining*, pages 181–186, 2010.

[7] T. A. Genez, L. F. Bittencourt, and E. R. Madeira. Workflow scheduling for saas/paas cloud providers considering two sla levels. In *Proc. of the IEEE Network Operations and Management Symposium*, pages 906–912, 2012.

[8] M. Guzek, P. Bouvry, and E.-G. Talbi. A survey of evolutionary computation for resource management of processing in cloud computing [review article]. *Computational Intelligence Magazine, IEEE*, 10(2):53–67, 2015.

[9] J. Hu, J. Gu, G. Sun, and T. Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Proc. of the Int. Symposium on Parallel Architectures, Algorithms and Programming*, pages 89–96, 2010.

[10] S. H. Jang, T. Y. Kim, J. K. Kim, and J. S. Lee. The study of genetic algorithm-based task scheduling for cloud computing. *Int. Journal of Control and Automation*, 5(4):157–162, 2012.

[11] H.-H. Li, Z.-G. Chen, Z.-H. Zhan, K.-J. Du, and J. Zhang. Renumber coevolutionary multiswarm particle swarm optimization for multi-objective workflow scheduling on cloud computing environment. In *Proc. of the Companion Publication on Genetic and Evolutionary Computation Conference*, pages 1419–1420, 2015.

[12] H.-H. Li, Y.-W. Fu, Z.-H. Zhan, and J.-J. Li. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling. In *Proc. of the IEEE Congress on Evolutionary Computation*, pages 870–876, 2015.

[13] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang. Cloud task scheduling based on load balancing ant colony optimization. In *Proc. of the Annual Chinagrid Conf.*, pages 3–9, 2011.

[14] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, and B.-N. Li. Job scheduling model for cloud computing based on multi-objective genetic algorithm. *Int. Journal of Computer Science Issues*, 10(3):134–139, 2013.

[15] P. Mell and T. Grance. The nist definition of cloud computing. Technical Report Special Publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce, 2011.

[16] S. Nakrani and C. Tovey. On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior*, 12(3-4):223–240, 2004.

[17] S. Pandey, L. Wu, S. M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Proc. of the 24th IEEE Int. Conference on dvanced Information Networking and Applications*, pages 400–407, 2010.

[18] S. Pandey, L. Wu, S. M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Proc. of the 24th IEEE Int. Conference on Advanced Information Networking and Applications*, pages 400–407, 2010.

[19] F. Pop, V. Cristea, N. Bessis, and S. Sotiriadis. Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments. In *Proc. of the 27th Int. Conference on Advanced Information Networking and Applications Workshops*, pages 772–776, 2013.

[20] O. A. Rahmeh, P. Johnson, and A. Taleb-Bendiab. A dynamic biased random sampling scheme for scalable and reliable grid networks. *INFOCOMP Journal of Computer Science*, 7(4):1–10, 2008.

[21] M. Randles, D. Lamb, and A. Taleb-Bendiab. A comparative study into distributed load balancing algorithms for cloud computing. In *Proc. of the IEEE 24th Int. Conference on Advanced Information Networking and Applications Workshops*, pages 551–556, 2010.

[22] V. Roberge, M. Tarbouchi, and G. Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. *IEEE Transactions on Industrial Informatics*, 9(1):132–141, 2013.

[23] M. A. Rodriguez and R. Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2):222–235, 2014.

[24] T. D. Seeley, S. Camazine, and J. Sneyd. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4):277–290, 1991.

[25] S. Selvarani and G. Sadhasivam. Improved cost-based algorithm for task scheduling in cloud computing. In *Proc. of the IEEE Int. Conference on Computational Intelligence and Computing Research*, pages 1–5, 2010.

[26] M. Shen, Z.-H. Zhan, W.-N. Chen, Y.-J. Gong, J. Zhang, and Y. Li. Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks. *IEEE Transactions on Industrial Electronics*, 61(12):7141–7151, 2014.

[27] X. Wang and Y. Wang. An energy and data locality aware bi-level multiobjective task scheduling model based on mapreduce for cloud computing. In *Proc. of the IEEE/WIC/ACM Int. Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 648–655, 2012.

[28] Z. Wu, Z. Ni, L. Gu, and X. Liu. A revised discrete particle swarm optimization for cloud workflow scheduling. In *Proc. of the Int. Conference on Computational Intelligence and Security*, pages 184–188, 2010.

[29] J. Xu, J. Tang, K. Kwiat, W. Zhang, and G. Xue. Enhancing survivability in virtualized data centers: a service-aware approach. *IEEE Journal on Selected Areas in Communications*, 31(12):2610–2619, 2013.

[30] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.*, 47(4):63:1–63:33, July 2015.

[31] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu. Independent tasks scheduling based on genetic algorithm in cloud computing. In *Proc. of the Int. Conf. on Wireless Communications, Networking and Mobile Computing*, pages 1–4, 2009.