IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 3, 2019, accepted March 17, 2019, date of current version April 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907615

Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing

MARWA GAMAL[1], RAWYA RIZK[2], HANI MAHDI[3], (Senior Member, IEEE), AND BASEM E. ELNAGHI[1]

[1]Electrical Engineering Department, Suez Canal University, Ismailia 41523, Egypt
[2]Electrical Engineering Department, Port Said University, Port Said 42526, Egypt
[3]Computers and Systems Engineering Department, Ain Shams University, Cairo 11566, Egypt

Corresponding author: Rawya Rizk (r.rizk@eng.psu.edu.eg)

ABSTRACT Cloud computing is increasing rapidly as a successful paradigm presenting on-demand infrastructure, platform, and software services to clients. Load balancing is one of the important issues in cloud computing to distribute the dynamic workload equally among all the nodes to avoid the status that some nodes are overloaded while others are underloaded. Many algorithms have been suggested to perform this task. Recently, worldview is turning into a new paradigm for optimization search by applying the osmosis theory from chemistry science to form osmotic computing. Osmotic computing is aimed to achieve balance in highly distributed environments. The main goal of this paper is to propose a hybrid metaheuristics technique which combines the osmotic behavior with bio-inspired load balancing algorithms. The osmotic behavior enables the automatic deployment of virtual machines (VMs) that are migrated through cloud infrastructures. Since the hybrid artificial bee colony and ant colony optimization proved its efficiency in the dynamic environment in cloud computing, the paper then exploits the advantages of these bio-inspired algorithms to form an osmotic hybrid artificial bee and ant colony (OH_BAC) optimization load balancing algorithm. It overcomes the drawbacks of the existing bio-inspired algorithms in achieving load balancing between physical machines. The simulation results show that OH_BAC decreases energy consumption, the number of VMs migrations and the number of shutdown hosts compared to existing algorithms. In addition, it enhances the quality of services (QoSs) which is measured by service level agreement violation (SLAV) and performance degradation due to migrations (PDMs).

INDEX TERMS Ant colony optimization, artificial bee colony, bio-inspired systems, cloud computing, load balancing, metaheuristics, osmotic computing.

I. INTRODUCTION

According to the National Institute of Standards and Technology (NIST), cloud computing is defined as ''pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction'' [1]. A developing number of organizations are transform to cloud computing to meet its requests as consumers and business can use applications without installation and access their personal files at any computer with Internet access.

Cloud computing has three services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In IaaS, fundamental resources can be accessed. PaaS provides the runtime environment for applications, development and deployment tools. SaaS allows providing software application as a service to the end users. All hardware infrastructure elements are virtualized into virtual entities. Virtualization is a technique that allows running different operating systems (OSs) together on one physical machine (PM). These OSs are separated from each other and from the underlying physical infrastructure by means of a special middleware abstraction called virtual machine (VM). The software that is responsible for managing these multiple VMs on PM is called VM kernel [2].

Cloud computing meets numerous challenges at increasing number of users because the demand of resources sharing and usage are increased rapidly. Therefore, load balancing between resources is an important challenge [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Yi Zhang.

Recently, worldview is turning into a new paradigm at area of computing called osmotic computing following the chemical osmotic behavior theory. Osmotic computing purposed to achieve balanced deployment in highly distributed environments [4], [5]. In cloud computing environment, osmosis computing can be exploited to develop balanced VMs that are migrated through cloud infrastructures.

Many bio-inspired algorithms prove their efficiency in load balancing systems such as ant colony and honey bee. However, most of them lack in achieving good results in all aspects. Therefore, hybrid algorithms are presented to exploits the advantages of each algorithm. In this paper, an Osmotic Hybrid artificial Bee and Ant Colony optimization (OH_BAC) load balancing algorithm is proposed. It will open a new trend to apply osmosis technique in load balancing.

The planning of this paper further is as follows: Section II presents a definition of osmosis technique. Section III presents an overview of related work in load balancing in cloud environments. In Section IV, the proposed OH_BAC algorithm is presented. The function of the proposed algorithm is tested with CloudSim environment in Section V. The results are presented in Section VI. The paper is concluded in Section VII.

II. OSMOSIS TECHNIQUE

In chemistry, "osmosis" represents the unrestrained net movement of molecules from a higher (low solute concentration) to a lower water concentration (high solute concentration) [4]. As shown in Figure 1(a), when pure liquid water and glucose are separated by a semi permeable membrane, water moves from high to low water activity as shown in Figure 1(b). So, osmotic pressure is important to be applied to a solution to stop the flow of water across a semi permeable membrane as shown in Figure 1(c). It is presented in Equation (1) [4].

$$\pi = ic_{solute}RT \tag{1}$$

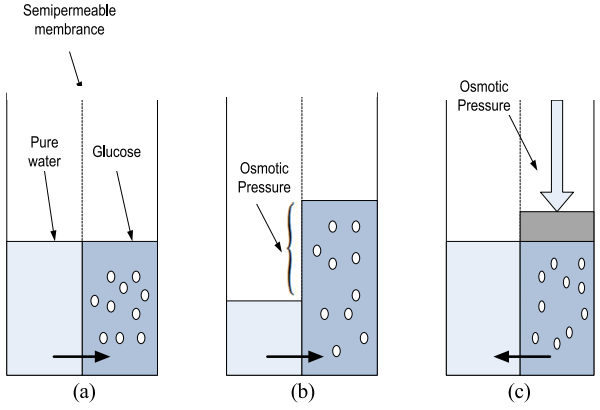


FIGURE 1. Osmosis technique and osmosis pressure. (a) Initial state (b) Equilibrium (c) External Pressure.

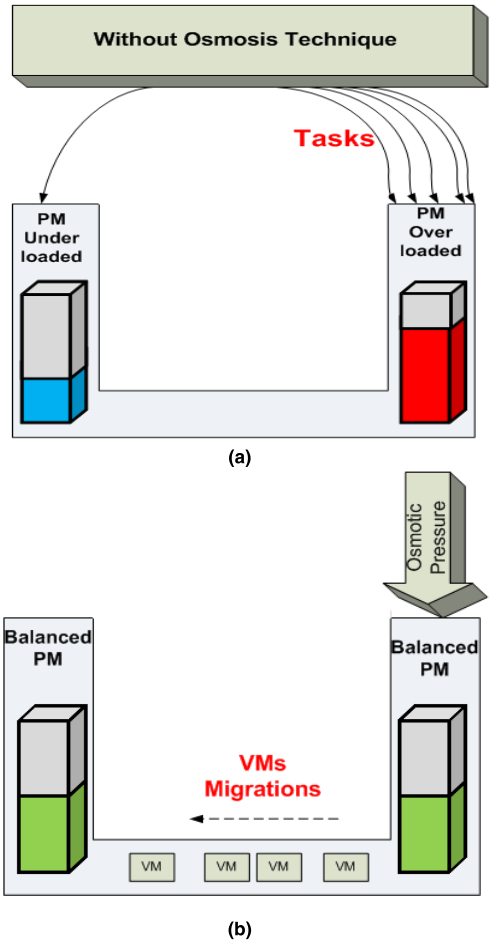


FIGURE 2. The effect of osmosis technique in Load balancing.

where π is the osmosis pressure law, i is a correction factor, c_{solute} is molar concentration of solution, R is ideal gas constant, and T is temperature in Kelvin.

In cloud environment, this process can be used to represent how VMs can be migrated across cloud computing as shown in Figure 2. Figure 2(a) shows over and under loaded PMs as a two liquids in tube, one is pure water and other is glucose. Figure 2(b) shows how osmosis technique affects on PMs to migrate VMs between them in order to achieve more balanced cloud system.

III. RELATED WORK

Load balancing algorithms are divided into two classes: Static and dynamic. In static load balancing, the balancing mechanism is done before the execution. It is done based on the probabilistic nature and no changes can be made during the execution, so time of execution-period cannot be determined exactly. In dynamic load balancing, the tasks are executed dynamically between all resources and it is necessary to monitor the current load of the system [6], [7].

A large study of different literatures was directed to different algorithms for dynamic load balancing. The most algorithms that drew attention are bio-inspired algorithms.

Many researchers looked to study the nature of bio-inspired to balance load between cloud environments. Bio-inspired load balancing algorithms are divided into evolutionary algorithms and swarm intelligence algorithms. Both are used for optimization issues such as load balancing and scheduling in cloud computing.

Evolutionary algorithms are developed mimicking the natural behavior for selection and enhancing. This class of algorithms is divided to two sub-classes genetic algorithms and genetic programming [8]. Swarm intelligence algorithms are depending on the behavior of some familiar living creatures, such as ants, bees, birds, and fishes which they have their own specific ways to discover the search space of the problems [9].

Fish Swarm Algorithm (FSA) is a swarm intelligent evolutionary technique which is inspired by the natural schooling behavior of fish [10], [11]. In FSA, there are three typical behaviors, defined as searching for food, swarming, and following to increase the chance of achieving a successful result by finding a greater amount of food. There are some advantages of FSA such as non-sensitive initial artificial fish location, flexibility and fault tolerance. Despite these advantages, FSA has also disadvantages such as causing higher time complexity, lower convergence speed, and it does not exploit its previous experiences for next moves.

Particle Swarm Optimization (PSO) algorithm is a bio-inspired algorithm which is extended from the social behavior of bird flocking searching for food [10], [9]. In PSO, each particle in the swarm acts a solution with four vectors, its present position, best position found, the best placement found by other best solution among all particles in the population, and its speed. Its position is set in the search area based on the best position reached by it and the best position reached by its neighborhood. The drawbacks of PSO algorithms are suffering from the partial optimism, which causes the less exact at the regulation of its speed and direction. PSO cannot solve the problems of the particles in the energy field.

The Firefly Algorithm (FA) produces flash light to attract the mating partner and potential prey [12]. This natural phenomenon is used to solve a great amount of complex load balance in cloud computing problem in managing the resources. However, there are some disadvantageous of FA such as its parameters are not change with the time and does not save any history of better case for each firefly.

BAT Algorithm is bio-inspired algorithm which depends on bats behavior [13]. When bats chase its prey, it flies unpredictably by changing the velocity, and positions based on the distance between the prey and itself.

Another algorithm is the Cuckoo search [14], [15] in which lay eggs in the nest of host birds. This algorithm helps in cheating the host cuckoo bird. This phenomenon can be used to solve a large amount of complex load balancing problems in managing the resources of cloud computing.

Flower pollination is another algorithm used in cloud environment [16]. On earth, nearly 80 percent of the plants makes flower pollination procedure. There are two types of the pollination process; biotic which transfers pollen grain by pollinators such as insects, birds, and bats and a-biotic pollination which does not require pollinator. This phenomenon is used to solve many complex computational and distributed problems of the cloud computing environment.

Ant Colony Optimization (ACO) is a random selecting algorithm which depends on the ant's behavior [17], [18]. It depends on searching widely for food by its pheromone trails for connection and back to their nest via shortest route by the concentration of pheromone. The concentration of the pheromone starts evaporating. The power of pheromone depends on the goodness and distance of the food [19]. ACO adapt to dynamic environments. It is extremely good in fault tolerance and scalability which improve the performance of the system. The downside of ACO is causing overhead due to utilizing more than one control parameter to find the amount of pheromone and the attractively of each movement.

Artificial Bee Colony (ABC) is another bio-inspired algorithm based on honey bee's behavior [20]. It is divided into three stages: Scout bees, employed bees, and onlooker bees. Scout bees are responsible for searching food source randomly, employed bees share information of food to the onlooker bees, and onlooker bees discover the amount of nectar and compute the probability. Finally, they come back to their hive and go to the dance zone to perform waggle dance. This dance is the best approach to share data about quality of food source. While sharing data, bees compute the quality of food and energy waste [21], [22]. After that, onlooker bees pick the best one and then scout honey bees back to the chosen food source to get nectar and come back to the hive. ABC performs well as system diversity growing. However, various downsides of ABC incorporate lack of use of secondary information, the possibility of losing important data, slow down when used in sequential processing, and the increasing number of solutions raises the computational cost [23].

At last years, many researches find new mechanism for load balancing by combining various algorithmic ideas to realize higher performance. Such approaches are commonly referred to as hybrid metaheuristics [24]. One of the examples of metaheuristics is joining ACO and ABC together. In [25], hybrid algorithm is proposed to combine ACO and ABC together. However, in the design of this algorithm, load balancing parameters are not stated and inherit the waggle dance behavior of ABC only.

In [26], Hybrid artificial Bee and Ant Colony optimization (H_BAC) algorithm is proposed to perform load balancing in VMs placement among hosts. It shows the strong points of both ACO and ABC. The pheromone behavior of ACO performs well in finding out solutions rapidly at variety systems, so it is adaptive to dynamic environments. In the other hand, ABC attains global load balancing its behavior of sharing data by waggle dancing. H_BAC improves the results in [25] by taking into consideration the parameter of monitoring the load of VM and the decision of load balancing before scheduling tasks in VMs.

Recently, the scientific researches tend to apply the osmosis theory from the chemistry science to form osmotic computing. The goal of this paper is to propose the osmotic behavior in order to realize the load balancing. In addition, it integrates both ACO and ABC with the osmosis technique in order to exploits their behavior while overcoming their drawbacks.

IV. THE PROPOSED OH_BAC ALGORITHM

The overview of the proposed OH_BAC algorithm is described in Figure 3. Figure 3(a) shows the unbalanced system which contains active and not active hosts. Active hosts in the cloud environment are divided into under loaded, over loaded and balanced hosts. As shown in Figure 3(a), there are no any balanced hosts in the system. In Figure 3(b) OH_BAC is applied for monitoring the state of system balance. Then, VMs are migrated from over to under loaded hosts to achieve load balanced system as shown in Figure 3(c).

OH_BAC inherits the main behaviors of ACO in discovering solutions rapidly at diversity systems and ABC in its behavior of sharing information by waggle dancing. Waggle dance is represented in OH_BAC as a knowledge base. OH_BAC applies knowledge base with osmosis technique to sort PMs according to energy consumption instead of selecting PMs randomly by ACO. OH_BAC takes into consideration the dynamic value of threshold according to state of cloud system.

Figure 4 shows the flow chart of the proposed OH_BAC algorithm. In the following, the OH_BAC algorithm is explained:

- In cloud computing environment each PM has a different number of VMs. The set of all PMs in datacenter is $P = \{P_1, P_2, \ldots\ldots, P_n\}$ where n is the number of PMs and the set of all VMs in datacenter is $V = \{v_1, v_2, \ldots\ldots, v_m\}$ where m is the number of VMs.
- In OH_BAC, Scout bee is responsible for calculating standard deviation (σ) for each PM to find both under and over utilized hosts. This need to find the load of each PM which depends on the load of VMs deployed into it. The average load of j^{th} VM in i^{th} PM ($\bar{V}_{ij}$) is calculating as following:

$$\bar{V}_{ij} = U_{Cpu_j} + U_{Mem_j} + U_{Bw_j} \tag{2}$$

where $U_{Cpu_j}, U_{Mem_j}, U_{Bw_j}$ is the CPU utilization, memory utilization, and bandwidth utilization of the j^{th} virtual machine VM_j, respectively.

The average load of PM_i ($\bar{P}_i$) and standard deviation for PM_i (σ_i) can be calculated as following:

$$\bar{P}_i = \frac{\sum_{j=1}^{m} \bar{V}_j}{m} \quad \forall V_{1,2,\ldots..m} \in P_i \tag{3}$$

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\bar{PT} - \overline{P_i})^2} \tag{4}$$

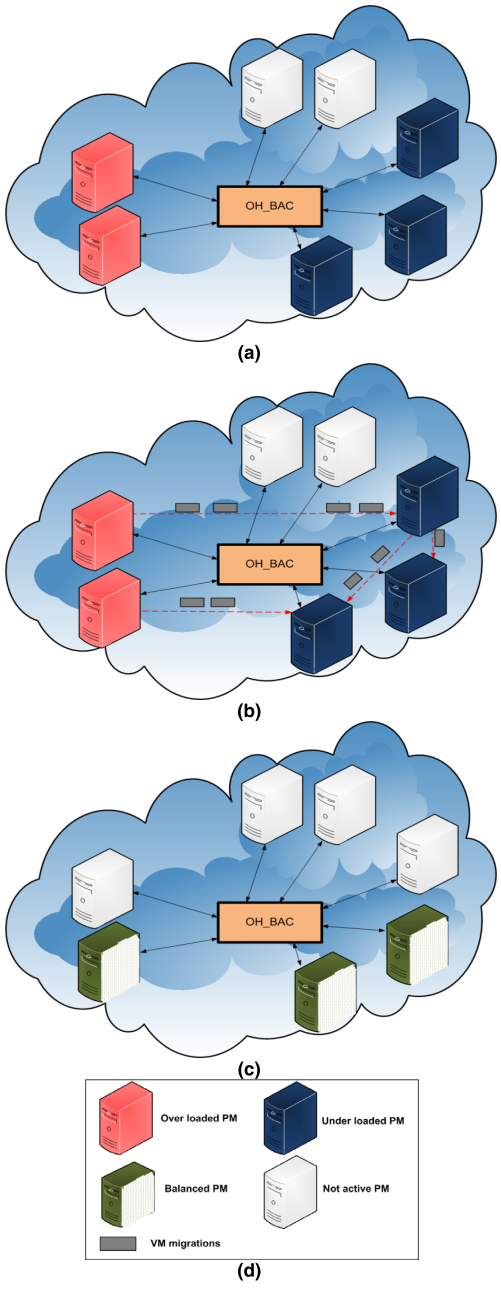


FIGURE 3. System overview (a) Before load balance (b) Applying Load balance with the proposed OH_BAC technique (c) Load balanced system (d) The legend of the system.

where n is the number of all PMs and $\overline{PT}$ is the average load of all PMs can be calculated as following:

$$\bar{PT} = \frac{1}{n}\sum_{i=1}^{n} \bar{P}_i \tag{5}$$

- If σ_i is less than lower threshold, then PM_i is under-utilized host. If σ_i exceed upper threshold, then PM_i is over utilized host. Threshold is computed as following: Lower threshold is the minimum $\bar{P}_i$ among all PMs and the upper threshold is equal $\bar{PT}$.

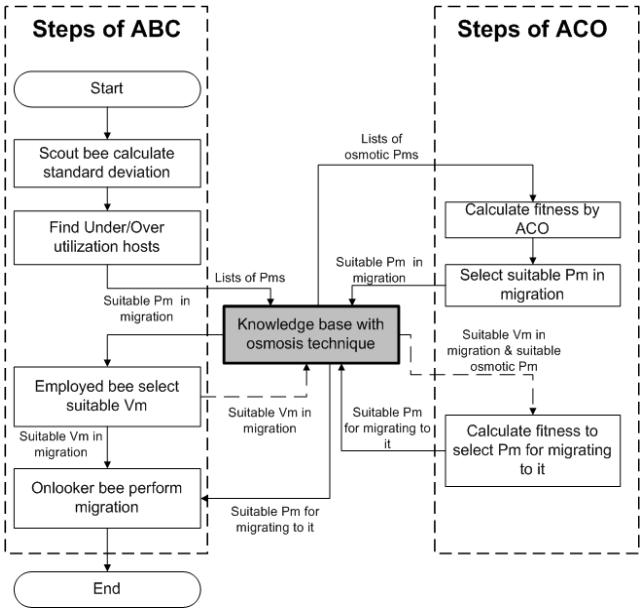


FIGURE 4. OH_BAC flowchart.

- After finding under/over utilized hosts, Scout bee makes its waggle dance in knowledge base to inform all swarms about its results.
- In this stage at OH_BAC algorithm, knowledge base performs osmosis technique to arrange hosts by (6) and takes into considerations power consumptions for each PM as shown in Figure 4.

$$\pi_{PM\,n} = iPC_{PM\,n} N_{PMs} L_{PM\,n} \tag{6}$$

where $\pi_{PM\,n}$ is the osmotic law, i is a correction factor, $PC_{PM\,n}$ is a power consumption of PM_n, N_{PMs} is a constant represented the number of PMs in cloud system, and $L_{PM\,n}$ is the load of PM_n.

- Then send the new hosts' list to ACO. After that ACO starts its trip to find the suitable PM among all osmotic hosts to perform virtual machine migration from it.
- ACO calculates its fitness (F_i) according to PM's classification (over/under utilized hosts) refer to (7) and (8), respectively.

$$F_i = \frac{(\tau_i)^{\alpha} * (\eta_i)^{\beta}}{\sum_{i=1}^{m} (\tau_i)^{\alpha} * (\eta_i)^{\beta}} \tag{7}$$

$$F_i = \frac{(1/\tau_i)^{\alpha} * (\eta_i)^{\beta}}{\sum_{i=1}^{m} (1/\tau_i)^{\alpha} * (\eta_i)^{\beta}} \tag{8}$$

where α and β give relative importance between pheromone τ_i, and edge weight η_i. The pheromone parameter τ_i is represented by the load of PM $_{\mathrm{i}}$ and ηi is the bandwidth.

- Hence, select the most suitable PM_i, put it in migration state and inform all swarm by knowledge base.
- Then, The pheromone is updated as following:

$$\tau_i = (1 - \rho)\,\tau_i + \Delta\tau_i \tag{9}$$

In the Equation (9) ($0 < \rho < 1$) is the evaporation rate and $\Delta\tau_i$ is defined as follows:

$$\Delta\tau_i = \frac{1}{(\eta_i)^{\gamma}} + P_i \tag{10}$$

where P_i is the load of PM and γ is a parameter to define the proportional importance of the heuristic value with relation to the load condition.

- After that, Employed bee performs its calculations to select suitable VM to be migrated to another host by the Minimal Migration Time (MMT) policy [20] which is the amount of RAM utilized by the VM divided by the spare network bandwidth available for the host$_i$ as following:

$$\frac{RAM(a)}{net_i} \leq \frac{RAM(u)}{net_i} \quad \forall a,\ u \in V_j \tag{11}$$

where V_j is set of VMs currently allocated to host$_{\mathrm{i}}$ and net_i is the spare network bandwidth available for the host$_{\mathrm{i}}$. The parameters $RAM\ (a)\ and RAM\ (u)$ are the amount of RAM currently utilized by the VM_a, VM_u, respectively.

- After this stage, ACO performs fitness function to find best mapping relationship between selected VMs to be matched to the most suitable PM which compatible with osmotic list of hosts from Knowledge base 19] as (12).

$$Fitness\left(VM_j, Pm_i\right) = \frac{Pm_{CPU_i} - VM_{CPU_j}}{VM_{CPU_j}} . \frac{Pm_{mem_i} - VM_{mem_j}}{VM_{mem_j}} . \frac{Pm_{net_i} - VM_{net_j}}{VM_{net_j}} . \frac{Pm_{storage_i} - VM_{storage_j}}{VM_{storage_j}} \tag{12}$$

where VM_{CPU_j}, VM_{mem_j}, VM_{net_j}, and $VM_{storage_j}$ represent the VM's parameters (CPU utilization, memory, bandwidth, and the storage size, respectively) which VM needs, and Pm_{CPU_i}, Pm_{mem_i}, Pm_{net_i}, and $Pm_{storage_i}$ represent the PM's parameters (CPU utilization, memory, bandwidth, and the storage size, respectively) which PM has.

- Finally, onlooker bees take its information about VM which will be migrated from employee bees and suitable PM to migrate VM to it by knowledge base. Onlooker bees perform migration by moving the VM to the suitable PM.

V. SIMULATION ENVIRONMENT

The proposed OH_BAC algorithm has been implemented using CloudSim API 3.0.3. Table 1 shows the simulation environment, where there are 50 PMs. There are two types of PMs, 50% of hosts are HP ProLiant ML110 G4 (Intel Xeon 3040, 1860 MHz, 2 cores, 4GB) and the other are HP ProLiant ML110 G5 (Intel Xeon 3075, 2 cores × 2660 MHz, 4GB). Each host has 1GB/s network bandwidth. The MIPS rating is 1860 MIPS for each core of HP ProLiant ML110 G4 hosts, and 2660 MIPS for each core of HP ProLiant ML110 G5 hosts.

TABLE 1. Parameters setting of cloud simulator.

Type	Parameter	Value
Host	Number of Hosts	50
	Types of Hosts	HP ProLiant ML110 G4
		HP ProLiant ML110 G5
HP ProLiant ML110 G4	Number of PEs per Host	2
	Bandwidth (GB)	1
	Host Memory (GB)	4
	MIPS of PE	1860
HP ProLiant ML110 G5	Number of PEs per Host	2
	Bandwidth (GB)	1
	Host Memory (GB)	4
	MIPS of PE	2660
VM	Total number of VMs	50
	Types of VMs	High-CPU Medium Instance
		Extra Large Instance
		Small Instance
		Micro Instance
High-CPU Medium Instance	MIPS of PE	2500
	Number of PEs per VM	1
	VM Memory (GB)	0.85
	Bandwidth (MB)	100
Extra Large Instance	MIPS of PE	2000
	Number of PEs per VM	1
	VM Memory (GB)	3.75
	Bandwidth (MB)	100
Small Instance	MIPS of PE	1000
	Number of PEs per VM	1
	VM Memory (GB)	1.7
	Bandwidth (MB)	100
Micro Instance	MIPS of PE	500
	Number of PEs per VM	1
	VM Memory (MB)	613
	Bandwidth (MB)	100
Cloudlets	Total number of Tasks	50-250
	Length of Task (MI)	2500*simulation limit
	Number of PEs per requirement	1
ACO algorithm parameter setting	Number of Iterations	100
	Number of Ants	5
	α	0.8
	γ	0.8
	β	0.32
	ρ	0.1
ABC algorithm parameter setting	Number of Iterations	100
	Number of Honeybees	15
	Number of Scout bees	7
	Number of employed bees	7
	Number of Onlooker bees	1
OH_BAC and H_BAC algorithms parameter setting	Number of Iterations	100
	Number of Ants	5
	Number of Honeybees	15
	α	0.8
	γ	0.8
	β	0.32
	ρ	0.1
	i	0.8

The type of all VMs which are deployed on the PMs have a single core, RAM is divided based on number of cores for each VMs types: High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB). We considered 50 heterogeneous VMs in a data center. The other parameters of OH_BAC are shown in Table 1.

TABLE 2. Comparison between host overloading detection algorithms and bio-inspired algorithms.

Algorithm	Energy consumption (Kwh)	SLAV $\times 10^{-4}$	SLATAH (%)	PDM (%)	No. of hosts' shut-downs	VM migr-ations
Lr	35.4	19	14	0.13	806	2872
Mad	46	19	9	0.23	1528	5265
Iqr	48	17	8	0.23	1549	5502
ABC	28.56	33.5	32	0.11	449	1404
ACO	34.4	46	27	0.17	781	2355
H_BAC	40	28	26	0.11	525	1532
OH_BAC	20	2	55	0.01	46	81

VI. SIMULATION RESULTS

In this section, OH_BAC algorithm is compared with existing schemes in two experiments. First, OH_BAC algorithm is compared with fixed parameters against ACO [18], ABC [20], H_BAC [26] and host overloading detection algorithms provided by [27] which are Inter Quartile Range (Iqr), Median Absolute Deviation (Mad), and Local Regression (Lr) algorithms with Minimal Migration Time (MMT) policy. Second, OH_BAC algorithm is compared with variable loads of tasks against bio-inspired algorithms (ACO [18], ABC [20], and H_BAC [26]). In addition, the time complexity of OH_BAC algorithm is measured and compared with bio-inspired algorithms. The fixed parameters of ACO, ABC, and H_BAC are shown in Table 1.

A. COMPARISON WITH HOST OVERLOADING DETECTION ALGORITHM

In the first experiment, the comparison between the proposed OH_BAC algorithm with H_BAC [26], ABC [20], ACO [18], and other algorithms provided by [27] in terms of energy consumption, the SLA Violation (SLAV), SLA Violations Time per Active Host (SLATAH), Performance Degradation due to Migrations (PDM), number of hosts shutdowns, and number of VMs migrations. This experiment is applied with fixed parameters. The number of tasks is equal to 50 tasks and the number of both hosts and VMs are equal to 50 which are stated in Table 1.

Table 2 shows the simulation results of the comparison between the proposed OH_BAC algorithm with host overloading detection algorithms and bio-inspired algorithms. It is shown that, H_BAC consumes energy more than ACO and ABC but OH_BAC decrease energy consumption with compared by other algorithms. OH_BAC improves H_BAC by applying osmotic technique as it takes into consideration the power consumption of each PM to select the lowest power consumption.

QoS is a necessary matter in cloud computing systems. QoS is formed in the form of SLAs, which are service-level

requirements for data center. SLA is considered when making a migration decision. When overloading and under loading cases occur, the PMs are ranked according to their loads. So that the suitable destination PM for VM migrations is detected when its capacity is suitable according to SLA [28]. SLAV is a useful metric used to evaluate the SLA delivered by a VM in an IaaS cloud by the following equation.

$$\text{SLAV} = \text{SLATAH} \times \text{PDM} \tag{13}$$

where SLATAH is service level agreement violations time per active host which formal by (14). SLATAH is a percentage of time that active host experienced the usage of CPU 100% and PDM is performance degradation due to migrations as (15).

$$\text{SLATAH} = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{14}$$

where N is the number of PMs, T_{s_i} is the whole time during i-th PM is utilized 100%; T_{a_i} is the total number of i-th PM that is in the active state.

$$\text{PDM} = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{15}$$

where M represents the number of VMs; C_{d_j} is the evaluation of the performance degradation of the j^{th} VM caused by migrations; C_{r_j} is the overall CPU capacity demanded by the j^{th} VM.

In Table 2, SLA Violation and its metrics are presented. As shown in Table 2, ACO and ABC have more SLAV with compared to other algorithms while OH_BAC performs high improvement with compared by all other algorithms. In addition, OH_BAC enhances the PDM because in OH_BAC algorithm, ACO and ABC cooperate to select the best VM to migrate to the most suitable PM. However, OH_BAC has more SLATAH with compared to other algorithms. As from the result of (14) which dependent on the number of active hosts, OH_BAC minimizes the number of active hosts with compared to other algorithms which determines the hosts that was active and then shutdown.

It is clear from the results that OH_BAC achieves better results than H_BAC and other algorithms. This is due to OH_BAC actives the most suitable osmotic host among all PMs in the system to decrease power consumption. OH_BAC takes into consideration the power consumption of each PM between active hosts with refer to (6) to select the lowest power consumption.

The experiments considered number of VM migrations as a metric to compare the performance of other algorithms. It is shown at results in Table 2 that OH_BAC and H_BAC have the least number of VM migrations among other algorithms.

Although, ABC gets better than ACO and host overloading detection algorithms provided by [27], OH_BAC gets more enhancement than H_BAC and ABC. This is due to ACO and ABC together in OH_BAC select the most suitable VM to migrate from the most suitable overloaded host.

Finally, OH_BAC improves energy consumption, SLAV, number of VM migrations, and number of hosts' shutdowns with compared to other algorithms. However, it has more SLATAH with compared to other algorithm but it is not effected in the performance of the cloud system.

B. COMPARISON WITH BIO-INSPIRED ALGORITHMS

In the second experiment, the loads of tasks are variable as the number of tasks is gradually increased from 50 to 250 tasks. All the previous parameters which were measured in the first experiment were also measured in the second experiment as shown in Figures 5-10.

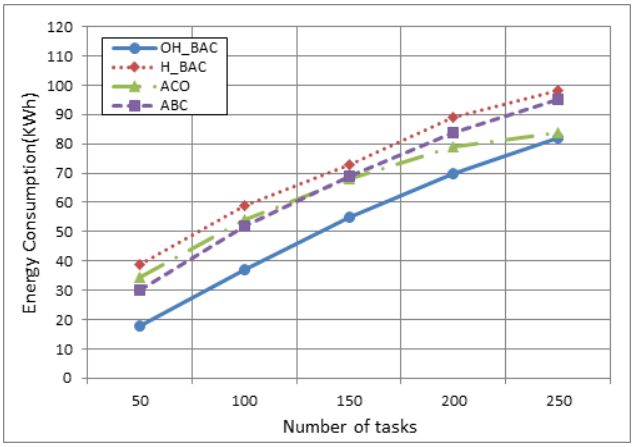


FIGURE 5. The comparison of energy consumption in bio-inspired algorithms.

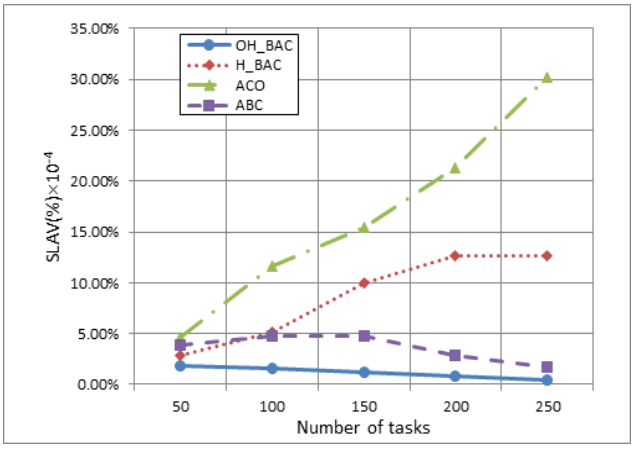


FIGURE 6. The comparison of SLA Violation in bio-inspired algorithms.

Figure 5 shows the energy consumption of OH_BAC and the other bio-inspired algorithms. It is shown that H_BAC consumes more energy than ACO and ABC. However OH_BAC achieves improvements by about 27% compared with H_BAC, 21% compared with ABC algorithm, and 18% compared with ACO algorithm.

From Figure 6 to Figure 8, SLAV and its metrics; PDM and ALATAH, respectively are presented. ACO and ABC have

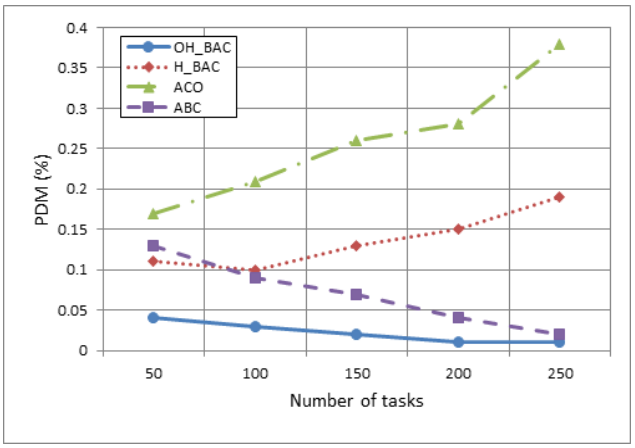


FIGURE 7. The comparison of SLA Violation metrics (PDM) in bio-inspired algorithms.

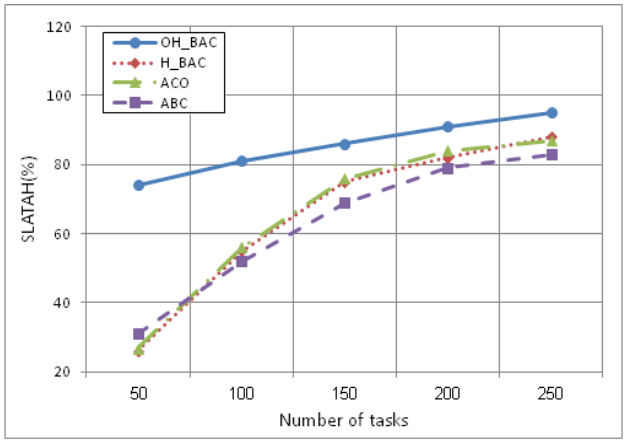


FIGURE 8. The comparison of SLA Violation metrics (SLATAH) in bio-inspired algorithms.

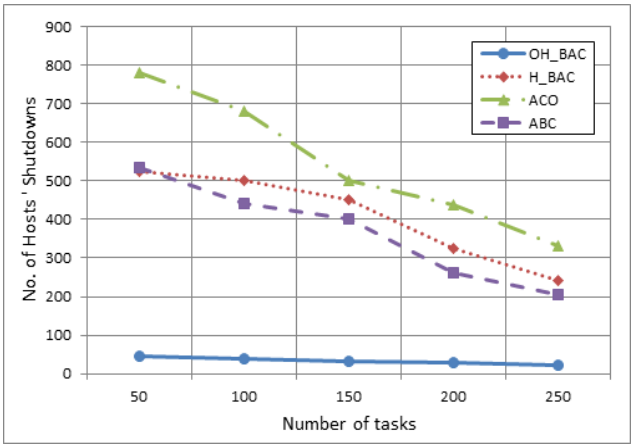


FIGURE 9. The comparison of number of hosts' shutdowns.

more SLAV compared with OH_BAC and H_BAC. However, OH_BAC is better than H_BAC by 98%, 99% compared with ACO, and 94% compared with ABC. OH_BAC enhances the PDM by about 84% compared with H_BAC, 69% compared

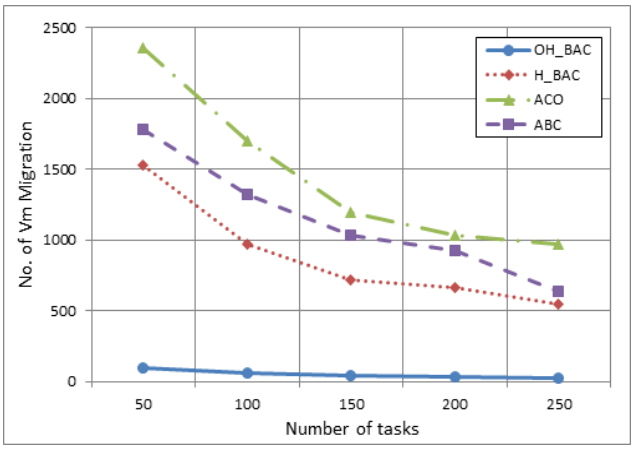


FIGURE 10. The comparison of number of VMs migrations in bio-inspired algorithms.

with ABC algorithm, 92% compared with ACO algorithm. Figure 7 shows that H_BAC starts with PDM less than ACO and ABC. However with increasing loads in cloud environments, H_BAC increases PDM as ACO algorithm due to random selection feature. However in OH_BAC, ABC and ACO cooperate to select the best VM to be migrated.

OH_BAC has more SLATAH compared with H_BAC, ABC, and ACO algorithms by 24%, 34%, and 30%, respectively as shown in Figure 8. However, it is not affected in the performance of the cloud system. As from the result of Equation (14) which dependent on the number of active hosts, OH_BAC minimizes the number of active hosts compared with H_BAC, ABC, and ACO algorithms by 93%, 91%, 94%, respectively as shown in Figure 9 which determine the hosts that was active and then shutdown. The number of shutdown hosts is not related to the number of hosts in the cloud environment. The host is shutdown if this host has not any tasks and this process may be done after VM migrations. If all VMs in one host are migrated, then the host will be off to decrease the power consumption. However, the host may be return active if there is one VM is migrated to it again. In OH_BAC algorithm, if there is any migration, OH_BAC checks first the active hosts. After that, if all active hosts cannot accept this VM, then it checks the not active hosts and open one.

Live virtual machine migration is a costly process that contains some amount of CPU processing on the source PM. Also, it depends on the bandwidth between the source and destination PMs. As well as, migrating VM process affects time as there are the down time of the services on the migrating VM and the total migration time. So, our goal is to minimize the number of VM migrations. Number of VM migrations doesn't depend on number of tasks. It is possible that one task is migrated more than 1000 time to achieve the balanced system. Then, the algorithm becomes efficient if it achieves balanced system with minimum number of migrations.

Hence, number of VM migrations is considered as a metric to compare the performance of the proposed algorithm with other algorithms. It is shown in Figure 10 that, OH_BAC and H_BAC have the least number of VM migrations among other algorithms. Although, ABC gets better than ACO and the algorithms provided by [27], OH_BAC is better than H_BAC, ABC and ACO by about 94%, 95%, and 96%, respectively. This is due to ACO and ABC perform better together in OH_BAC in order to select the most suitable VM to migrate from the most suitable overloaded host.

C. TIME COMPLEXITY

Table 3 shows the time complexity of OH_BAC compared with other bio-inspired algorithms. It is shown that OH_BAC has a minimum value of processing time. The time complexity of OH_BAC is calculated as follows: $O(2n^2 + 17n)$ due to the consumed time by ABC and ACO in OH_BAC is O(n) as each of these algorithms has one inner loop to search suitable food source among whole cloud system. As well as, the consumed time by knowledge base is $O(n^2)$ as knowledge base has two inner loops to sort under loaded PMs and over loaded PMs. Hence, time complexity of OH_BAC can be summarized as $O(n^2)$. Note that the time complexity shown in the table before final abbreviation. Precisely, that leads to $O(n^2)$ which is the result of time complexity of OH_BAC, H_BAC and ACO, as along with $O(n^3)$ of ABC.

TABLE 3. Time complexity of bio-inspired algorithms.

Algorithm	Time complexity
ABC	$O(n^3+ n^2 +4n+ \log n) = O(n^3)$
ACO	$O(5n^2+8n + \log n) = O(n^2)$
H_BAC	$O(3n^2+18n + \log n) = O(n^2)$
OH_BAC	$O(2n^2+17n) = O(n^2)$

VII. CONCLUSION

In this paper, the osmosis theory from the chemistry science is proposed to form osmotic computing and find load balancing for VM placement by OH_BAC algorithm. OH_BAC applies osmosis technique to provide energy efficient cloud computing environment. In OH_BAC algorithm, ACO and ABC cooperate to select the best VM to migrate to the most suitable PM. In addition, OH_BAC makes activation to the most suitable osmotic host among all PMs in the system to decrease power consumption. The proposed algorithm has been simulated to calculate the performance at various metrics in two experiments with fixed and variable loads. OH_BAC algorithm is compared with ACO, ABC, H_BAC and host overloading detection algorithms. The simulation results showed that OH_BAC improves energy consumption, SLAV, number of virtual machine migration, and the number of hosts' shutdowns when compared with other algorithms in fixed and variable loads. However, OH_BAC has more SLATAH when compared with other algorithms but it is not affect the performance of the considered cloud system.

REFERENCES

[1] X. Xu, H. Hu, N. Hu, and W. Ying, "Cloud task and virtual machine allocation strategy in cloud computing environment," in *Proc. NCIS*, Berlin, Germany, 2012, pp. 113–120.

[2] H. Nashaat, N. Ashry, and R. Rizk, "Smart elastic scheduling algorithm for virtual machine migration in cloud computing," *J. Supercomput.*, pp. 1–24, Jan. 2019. doi: 10.1007/s11227-019-02748-2.

[3] A. Shawish and M. Salama, "Cloud computing: Paradigms and technologies," in *Inter-cooperative Collective Intelligence: Techniques and Applications*, vol. 495. Berlin, Germany: Springer, 2014, pp. 39–67.

[4] M. Villari, A. Celesti, and M. Fazio, "Towards osmotic computing: Looking at basic principles and technologies," in *Proc. CISIS*, 2017, pp. 906–915.

[5] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 76–83, Nov./Dec. 2016.

[6] S. Aslam and M. A. Shah, "Load balancing algorithms in cloud computing: A survey of modern techniques," in *Proc. NSEC*, Rawalpindi, Pakistan, Dec. 2015, pp. 30–35.

[7] W. E. Saber, R. Y. Rizk, W. M. Moussa, and A. M. Ghuniem, "LBSR: Load balance over slow resources," in *Proc. 1st Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Riyadh, Saudi Arabia, Apr. 2018, pp. 1–7.

[8] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1–17, Apr. 2017.

[9] B. Balusamy, J. Sridhar, D. Dhamodaran, and P. V. Krishna, "Bio-inspired algorithms for cloud computing: A review," *Int. J. Innov. Comput. Appl.*, vol. 6, nos. 3–4, pp. 182–202, 2015.

[10] S. Binitha and S. S. Sathya, "A survey of bio inspired optimization algorithms," *Int. J. Soft Comput. Eng.*, vol. 2, pp. 137–151, May 2012.

[11] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. N. Toosi, "Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications," *Artif. Intell. Rev.*, vol. 42, no. 4, pp. 965–997, 2014.

[12] N. J. Kansal and I. Chana, "Energy-aware virtual machine migration for cloud computing—A firefly optimization approach," *J. Grid Comput.*, vol. 14, no. 2, pp. 327–345, 2016.

[13] S. Sharma, A. K. Luhach, and S. A. Sinha, "An optimal load balancing technique for cloud computing environment using bat algorithm," *Indian J. Sci. Technol.*, vol. 9, no. 28, pp. 1–4, 2016.

[14] R. Singh, "Cuckoo genetic optimization algorithm for efficient job scheduling with load balance in grid computing," *Int. J. Comput. Netw. Inf. Secur.*, vol. 8, no. 8, pp. 59–66, 2016.

[15] A. A. Alexander and D. L. Joseph, "An efficient resource management for prioritized users in cloud environment using cuckoo search algorithm," *Procedia Technol.*, vol. 25, pp. 341–348, 2016. doi: 10.1016/j.protcy.2016.08.116.

[16] G. Singh and A. Kaur, "Bio inspired algorithms: An efficient approach for resource scheduling in cloud computing," *Int. J. Comput. Appl.*, vol. 116, no. 10, pp. 16–21, 2015.

[17] K. Nishant *et al.*, "Load balancing of nodes in cloud using ant colony optimization," in *Proc. UKSim-ICCS*, Cambridge, U.K., Mar. 2012, pp. 3–8.

[18] W.-T. Wen, C.-D. Wang, D.-S. Wu, and Y.-Y. Xie, "An ACO-based scheduling strategy on load balancing in cloud computing environment," in *Proc. IEEE-FCST*, Dalian, China, Aug. 2015, pp. 364–369.

[19] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "An ant colony based load balancing strategy in cloud computing," in *Proc. ICACNI*, 2014, pp. 403–413.

[20] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-MMT: A load balancing method for power consumption management in cloud computing," in *Proc. IC*, Aug. 2013, pp. 76–80.

[21] L. D. D. Babu and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292–2303, May 2013.

[22] Y. S. Sheeja and S. Jayalekshmi, "Cost effective load balancing based on honey bee behaviour in cloud environment," in *Proc. ICCSC*, Dec. 2014, pp. 214–219.

[23] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing," *Trans. Internet Inf. Syst.*, vol. 11, no. 12, pp. 5694–5710, Dec. 2017.

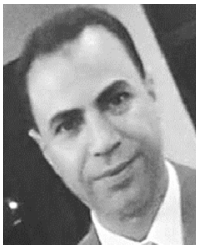
[24] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *J. Appl. Soft Comput.*, vol. 11, no. 6, pp. 4135–4151, 2011.

[25] R. Madivi and S. S. Kamath, "An hybrid bio-inspired task scheduling algorithm in cloud environment," in *Proc. ICCCNT*, Hefei, China, Jul. 2014, pp. 1–7.
[26] M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, "Bio-inspired load balancing algorithm in cloud computing," in *Proc. AISI*, 2017, pp. 579–589.
[27] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various VM placement strategies in CloudSim," *J. Cloud Comput.*, vol. 4, no. 1, pp. 1–21, Dec. 2015.
[28] T. T. Zin, J. C. W. Lin, J. S. Pan, P. Tin, and M. Yokota, "Genetic and evolutionary computing," in *Proc. 9th Int. Conf. Genetic Evol. Comput. (ICGEC)*, vol. 1. Yangon, Myanmar: Springer, 2015.

MARWA GAMAL received the B.Sc. and M.Sc. degrees in computer and control engineering from Suez Canal University, Egypt, in 2007 and 2012, respectively, where she is currently an Assistant Lecturer with the Electrical Engineering Department. Her research interest includes the area of cloud computing.

RAWYA RIZK received the B.Sc., M.Sc., and Ph.D. degrees in computer and control engineering from Suez Canal University, in 1991, 1996, and 2001, respectively. Since 2014, she has been the Chief Information Officer (CIO) of Port Said University (PSU), Egypt. Since 2017, she has been the Head of the Electrical Engineering Department, PSU, where she is currently a Professor of computers and control with the Electrical Engineering Department. Her research interests include computer networking, including mobile networking, wireless, ATM, sensor networks, ad hoc networks, quality of service (QoS), traffic and congestion control, handoffs, and cloud computing. She is a Reviewer for many of international communication and computer journals such as the IEEE Access, *IET Communications*, *IET Sensors*, *IET Networks*, the *Journal of Supercomputing*, the *Journal of Network and Computer Applications*, *Computers & Electrical Engineering*, *Mathematical Problems in Engineering*, and IJACSA.

HANI MAHDI received the Doctorate degree from Technische Universität, Braunschweig, Germany, in 1984. He was a Postdoctoral Research Fellow with the Electrical and Computer Engineering Department, The Pennsylvania State University, PA, from 1988 to 1989. He was a Visiting Professor with the Computer Vision and Image Processing (CVIP) Lab, Electrical Engineering Department, University of Louisville, KY, USA, from 2001 to 2002. He was on leave from Al-Isra University, Amman, Jordan; El-Emarat University, El Ain, United Arab Emirates; and Technology College, Hufuf, Saudi Arabia. He is currently a Professor of computer systems with the Faculty of Engineering, Ain Shams University, Cairo, Egypt. He was a recipient of the Distinguished University Award of Ain Shams University, in 2015. He received the Best (First) Graduated Students from the Electrical Engineering Department, Faculty of Engineering, Communication and Electronics Section, Ain Shams University, in 1971.



BASEM E. ELNAGHI received the B.Sc. and M.Sc. degrees in electrical engineering from Suez Canal University, Port Said, Egypt, in 2003 and 2009, respectively, and the Ph.D. degree in electrical engineering from Port Said University, Port Said, in 2015.

He is currently an Assistant Professor with the Electrical Engineering Department, Suez Canal University, Ismailia, Egypt.