International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 3, June 2021, pp. 2477~2489
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i3.pp2477-2489

Hybrid load balance based on genetic algorithm in cloud environment

Walaa Saber[1], Walid Moussa[2], Atef M. Ghuniem[3], Rawya Rizk[4]
[1,4]Electrical Engineering Department, Port Said University, Port Said, Egypt
[2,3]Electrical Engineering Department, Suez Canal University, Ismailia, Egypt

Article Info

Article history:

Received Aug 9, 2020
Revised Dec 18, 2020
Accepted Dec 29, 2020

Keywords:

Cloud computing
Genetic algorithm
Load balancing
Load deviation
Metaheuristic

ABSTRACT

Load balancing is an efficient mechanism to distribute loads over cloud resources in a way that maximizes resource utilization and minimizes response time. Metaheuristic techniques are powerful techniques for solving the load balancing problems. However, these techniques suffer from efficiency degradation in large scale problems. This paper proposes three main contributions to solve this load balancing problem. First, it proposes a heterogeneous initialized load balancing (HILB) algorithm to perform a good task scheduling process that improves the makespan in the case of homogeneous or heterogeneous resources and provides a direction to reach optimal load deviation. Second, it proposes a hybrid load balance based on genetic algorithm (HLBGA) as a combination of HILB and genetic algorithm (GA). Third, a newly formulated fitness function that minimizes the load deviation is used for GA. The simulation of the proposed algorithm is implemented in the cases of homogeneous and heterogeneous resources in cloud resources. The simulation results show that the proposed hybrid algorithm outperforms other competitor algorithms in terms of makespan, resource utilization, and load deviation.

Corresponding Author:

Rawya Rizk
Electrical Engineering Department
Port Said University
Port Said, Egypt
Email: r.rizk@eng.psu.edu.eg

1. INTRODUCTION

Cloud computing technology provides a lot of services to all users over the internet using very large scalable and virtualized resources. The main objective of the cloud is to provide services all over the world with minimum cost and high performance [1, 2]. To have the ability to allow all huge number of clients all over the world to share cloud resources and provide them with high-quality service in a reasonable time, all client's requests should be handled in an efficient way that don't waste time and resources. For that reason, there is a big need for load balancing techniques which are the master key for the success of any cloud services provider. Load balancing tries to keep cloud nodes equally loaded to avoid a situation where some of the resources are overloaded while some others are under loaded which as a result reduce the response time of the assigned tasks [3-5]. Load balancing is an efficient technique used to distribute workloads over resources in a way that improve resource utilization and response time. Load balancing tries to keep cloud resources equally loaded and avoid resources becoming over-loaded or under-loaded [6].

Traditional algorithms [7-11] are used to solve this problem. However, these algorithms have limitations in the case of complex and large scale problems. Metaheuristic algorithms such as particle swarm

optimization (PSO) [12], ant colony optimization (ACO) [13], artificial bee colony (ABC) [14], and genetic algorithm (GA) [15, 16] are popular to solve non-deterministic polynomial-time (NP) complete problems. The convergence process and speed of metaheuristic algorithms with a complete random population become worse when increasing the number of jobs that make the problem more complex. Using an efficient scheduling algorithm that produces good initial solution to the initial population of metaheuristic algorithm makes use of the computational power of this metaheuristic algorithm and overcomes their drawbacks with complicated random initialized problems [17, 18].

Genetic algorithm (GA) as an evolutionary algorithm became a very popular algorithm due to its accuracy in solving complicated non-linear problems. GA has been successfully applied to many non-linear and non-smooth types of optimization challenges such as query optimization [19], medical science [20], agriculture [21], management [22], feature selection [23], power flow management [24], and sensor networks [25]. GA is basically designed for the discrete optimization problem where bits of 0's and 1's are used to encode discrete design variables. Unlike bio-inspired algorithms that are designed for continuous problems and can choose any value to encode design variables, which makes GA more suitable than other algorithms in the load balancing problem. Choosing good initial population of GA is an important step to generate new better generations with high-quality solutions within less time [26].

In this paper, a hybrid load balance based on genetic algorithm (HLBGA) is proposed to distribute the loads overall virtual machines (VMs) in an efficient way. HLBGA is implemented in two phases. In the first phase, the heterogonous initialized load balancing (HILB) algorithm is proposed. It distributes tasks overall VMs in an efficient way to avoid overloaded or under loaded VMs. In the second phase, GA is used to enhance the overall system performance. It is initialized with the output of the HILB algorithm as a good initial population for GA. This phase uses a newly formulated fitness function for GA that helps the HLBGA to reach the optimal load deviation.

The rest of this paper is organized as: Section 2 presents the related load balancing algorithms. In Section 3, the proposed load-balancing algorithm is introduced. In Section 4, the performance evaluation of the proposed algorithm is presented and compared with the existing load balancing algorithms. Section 5 presents the main conclusions and future work.

2. RELATED WORK

A large area of researches was introduced to solve the load balancing problem to get an optimal assignment solution. These researches can be categorized into three main types of algorithms: traditional, metaheuristic, and hybrid algorithms.

2.1. Traditional algorithms

Traditional algorithms are worked based on knowing information about resources and tasks to calculate their evaluation parameters. Most of them rely on execution time to assign tasks to resources in a way that minimizes makespan, load deviation, or both. Min-Min algorithm is a well-known algorithm in this category. Min-Min algorithm is the base of many scheduling algorithms [8]. In this algorithm, the completion time of all submitted tasks among all VMs is calculated. The task with minimum completion time is assigned to the corresponding VM. Then the completion time of all other tasks on that machine is updated by adding the completion time of the assigned task to their completion times. This task is removed from a list of unassigned tasks, and then this procedure is repeated until all tasks are assigned.

Load balance improved Min-Min (LBIMM) algorithm improves the standard Min-Min algorithm [9]. In the first step, the Min-Min algorithm is executed to give the initial solution to start the next step. In the next step, the completion time of the smallest size task from the heaviest loaded resource is calculated on all other VMs. Makespan is calculated in case that task is removed to the VM with the minimum completion time of that task and compared with the makespan produced by Min-Min. If it is less than the task, it is reassigned to the resource that produces it, and the ready time of both resources is updated. The process repeats until no other reassignments can produce less makespan. Thus the heavy load resources are freed and the light load or idle resources are more utilized. Although the traditional algorithms are simple to implement and can improve makespan, some of them don't take the load deviation in its consideration especially in case of big difference in resource speed. Also, they can't find the optimal solution especially when the problem becomes complex or too large [25].

2.2. Metaheuristic algorithms

Metaheuristic algorithms are the most powerful techniques for the optimization of complex non-linear problems which is the case of most task scheduling and load balancing issues [26]. Metaheuristic algorithms can be classified into swarm intelligence based algorithms and evolutionary algorithms. Swarm

intelligence based algorithms such as PSO, ACO, and ABC optimize a certain problem by simulating the collective behavior of natural swarms. Evolutionary algorithms such as GA are based on the evolutionary behavior of natural systems.

PSO algorithm is one of the standard algorithms used in load balancing and also in other applications [27, 28]. It is a swarm intelligent algorithm, inspired by nature for solving nonlinear optimization problems [10]. PSO is a simulation of the advantages of bird flocks. It starts with initial individuals called particles representing initial solutions for the problem. During the search process, killing of any individual is not permitted. In PSO, all individuals remain alive and try to make themselves stronger throughout the search process. In every generation/iteration, individuals make themselves better. The identity of the individual does not change over the iterations.

GA is an evolutionary optimization algorithm based on the biological concept of population generation [13]. A new population is evolved in every generation based on predefined fitness function. GA works better for vast and complex search space problems. It works based on three main operations which are selection, crossover, and mutation. The strength of GA is in the parallel nature of its search. The genetic operators used are the main powerful reason for the success of the search. Crossover is the main genetic operator, whereas mutation is used less frequently. Crossover attempts to benefit offspring solutions and to eliminate undesirable components. By restricting the reproduction of weak offsprings, GAs eliminates not only that solution but also all of its descendants. This makes the algorithm converge towards high-quality solutions within a few generations. In order to realize powerful crossover and mutation operators, we must choose good initial population for GA [14].

However metaheuristic algorithms are powerful techniques for optimization, they are inefficient to handle the load in cloud computing in case of random initial population. Also, they suffer from increasing the computational cost in the large scale problems [29]. Therefore, hybrid algorithms are introduced to enhance the performance of both the traditional and metaheuristic algorithms in order to handle their problems.

2.3. Hybrid algorithms

Hybrid task scheduling algorithms are based on combining two scheduling algorithms to make use of the advantage of both these two algorithms. This paper presents some of the most popular hybrid algorithms to state the reason for the proposed algorithm. HGA-ACO algorithm [30] combines GA and ACO algorithms together. Randomly initialized GA is used to produce the initial pheromone for ACO. ACO starts to iterate in order to give the best solution. The best two solutions from GA and ACO are merged by crossover to give the global best solution. However, the algorithm focuses on response time, execution time and throughput, it doesn't subject to the load balancing problem. GA is not an effective algorithm to give an initial solution when it is randomly initialized.

Osmotic hybrid artificial bee and ant colony (OH_BAC) algorithm is presented in [31]. It applies the osmosis technique for providing energy efficient cloud environment. In this algorithm, ABC and ACO cooperate to select the appropriate VM to be migrated to the most suitable physical machine. In addition, it makes activation for the most suitable osmotic host among all physical machines in the system to decrease power consumption.

Moreover, integrating machine learning techniques with load balancing algorithms reinforcement the learning process and help to improve the performance and the convergence rate of the load balancing process [32]. However, the goal of most of these algorithms is to minimize the overall completion time without looking into the minimization of the overall load deviation. Most of previous algorithms choose minimizing makespan as the main goal in scheduling; however this target always chooses faster VMs to perform the assigned tasks. This results in overloaded VMs with high processing speed that yields to starvation problem of other VMs with lower processing time. In addition, the experiments of most of related work are limited as they tested their algorithms on small scale problems [33]. In this paper, a new hybrid HLBGA balancing algorithm is proposed which combines GA and a new proposed HILB scheduling algorithm which helps genetics to converge more quickly to better solution by feeding it with good initial population.

3. THE PROPOSED HLBGA

3.1. Architecture overview

In this section, the proposed HLBGA is presented. The main purpose of the proposed algorithm is to improve the assignment performance for all the submitted tasks on all VMs. It tries to assign tasks to each VM based on its computing capabilities to make use of all of them which leads at the end to balance the load among all VMs. Load balance is an optimization problem in which load deviation is the objective function needed to be minimized. GA is one of the popular algorithms that are used to solve optimization problems. The proposed algorithm uses GA with a good initial population to get the optimal solution with less time.

The proposed HLBGA is based on two main phases. The first phase is applying the proposed HILB algorithm that distributes tasks overall VMs based on each resource computing capabilities to ensure that no single VM is either overloaded or underutilized especially in case of major differences between resources computing capabilities. The second phase uses the output of the HILB algorithm as an initial population for the GA which optimizes load deviation objective function to achieve optimum load distribution.

The proposed HLBGA algorithm introduces a new objective function to improve the performance of the assignment problem even when the problem becomes complex or too large. It implemented in different environments, homogeneous, heterogeneous-low and heterogonous-high environments. HLBGA also is implemented on a different number of tasks. It improves resource utilization and it also decreases both the load deviation and the makespan.

3.2. Load balancing problem analysis

Although cloud computing is dynamic, at any particular instance the load balancing problem can be formulated as assigning a set of *n* tasks on a set of *m* VMs. Assume that the cloud task scheduler receives *n* independent tasks $t_1\ t_2\ t_3\ \dots\dots t_n$ with different lengths, which are expressed in million instructions (MI) as (1):

$$T = [t_1 t_2 t_3\ \dots t_i \dots . t_n]^T \text{ where } t_i \text{ is the length of task } i \text{ and } i = \{1.2. \dots . n\} \quad (1)$$

Also, assume that the cloud task scheduler contains information about the m VMs; $v_1\ v_2\ v_3 \dots\dots . v_m$ with different processing speeds, which are expressed in million instructions per second (MIPS) as:

$$V = \left[v_1 v_2 v_3 \dots v_j \dots . v_m\right]^{\mathrm{T}} \quad (2)$$

where v_j is the processor speed of VM j and $j = \{1.2. \dots . m\}$

The assignment matrix θ of tasks over VMs can be represented as:

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{1j} & \theta_{1m} \\ \vdots & \dots & \vdots \\ \theta_{i1} & \theta_{ij} & \theta_{im} \\ \vdots & \vdots & \vdots \\ \theta_{n1} & \theta_{nj} & \theta_{nm} \end{bmatrix} \quad (3)$$

where $\theta_{ij} = 1$ if task t_i is assigned to VM v_j , otherwise $\theta_{ij} = 0$

Assume also that at any time there will be load matrix X contains information about the current load of the m VMs $x_1\ x_2\ x_3 \dots\dots . x_m$. The VMs loads are defined in the load matrix as:

$$X = \left[x_1 x_2 x_3 \dots x_j \dots . x_m\right]^T \quad (4)$$

$$x_j = \sum_{i=1}^{n} \frac{\theta_{ij} t_i}{v_j} \quad \text{where } x_j \text{ is the current load of VM} j \text{ and } j = \{1.2. \dots . m\} \quad (5)$$

The performance of the assignment solution can be measured using makespan, load deviation (σ), and resource utilization (U). They can be calculated as [23]:

$$MakeSpan = \max(x_j) \quad \forall j \text{ where } x_j \text{ is the completion time of VM}_j. \quad (6)$$

$$\sigma = \sqrt{\frac{\sum_{j=1}^{m} (x_j - \mu)^2}{m}} \quad \text{where } \mu = \frac{\sum_{j=1}^{m} x_j}{m} \quad (7)$$

$$U = \frac{\mu}{MakeSpan} \times 100 \quad (8)$$

3.3. The problem formulation of HLBGA

The goal of the proposed HLBGA algorithm is to optimally assign a set of tasks on a set of VMs in a way that minimize the load deviation of all VMs. Minimizing load deviation yields to minimize makespan and maximize resource utilization since it assigns the tasks to all VMs with different

computing capabilities. It ensures that all VMs are not overloaded or under loaded. Then it prevents starvation problem of VMs with low processing speed. Table 1 shows the parameters' notations that are used in the proposed model.

Table 1. Parameters' notations used in the proposed model

Parameter	Meaning
n	The number of tasks
m	The number of VMs
T_{nx1}	The task length matrix where t_i is the length of i^{th} task in MI
V_{mx1}	The processor speed matrix where v_i is processor speed of j^{th} VM in MIPS
X_{mx1}	The load Matrix for all VM where x_i is load of j^{th} VM
σ^2	Load variance
σ	Load deviation
μ	Load Mean
$\theta_{n\times m}$	Assignment matrix where θ_{ij} is a binary bit equals to 1 or 0, which represents assignment state of task i on $VM\ j$

The proposed model formulates the objective in terms of the assignment matrix. It tries to get the assignment matrix that provides the solution with minimum load deviation. The load variance can be obtained as:

$$\sigma^2 = \frac{\sum_{j=1}^{m}(x_j-\mu)^2}{m} \tag{9}$$

assume

$$\dot{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \mu = X - 1\,\mu \tag{10}$$

then

$$\sum_{j=1}^{m}(x_j-\mu)^2 = \dot{X}^T\dot{X} \tag{11}$$

because

$$\sum_{j=1}^{m} x_j = 1^T\,X \tag{12}$$

then

$$\mu = \frac{1^T X}{m} \tag{13}$$

Substitute (13) in (10)

$$\dot{X} = (I - \frac{1\,1^T}{m})\,X \tag{14}$$

where I is an identity matrix,

$$\text{let} Q = \left(I - \frac{1\,1^T}{m}\right) \text{ then } \dot{X} = QX \tag{15}$$

All the diagonal elements of the Q matrix are $\frac{m-1}{m}$ and its off-diagonal elements are $\frac{-1}{m}$, so Q is an idempotent matrix [34]. The matrix Q is useful in computing sums of squared deviations.

$$Q = \frac{1}{m}\begin{bmatrix} m-1 & -1 & \dots & -1 \\ -1 & m-1 & -1 & \vdots \\ \vdots & -1 & \ddots & -1 \\ -1 & \dots & -1 & m-1 \end{bmatrix} \tag{16}$$

By substituting (11) in (9)

$$\sigma^2 = \frac{\dot{X}^T \dot{X}}{m} = \frac{X^T Q^T Q X}{m} \quad \text{where } Q^T Q = Q, \text{ then} \tag{17}$$

$$\sigma^2 = \frac{X^T\ Q\ X}{m} \tag{18}$$

$$X^T\ Q\ X = \sum_{k=1}^{m} x_k^2 q_{kk} + \sum_{\substack{z=1 \\ z \neq j}}^{m} \sum_{j=1}^{m} x_z x_j q_{zj} \tag{19}$$

where $q_{kk} = \frac{m-1}{m}, k = 1, 2, \ldots, m$ and $q_{\substack{zj \\ z \neq j}} = -\frac{1}{m}, z, j = 1,2, \ldots, m$

$$\sigma^2 = \frac{1}{m^2}\left[(m-1)\sum_{k=1}^{m} x_k^2 - \sum_{Z=1}^{m} \sum_{\substack{j=1 \\ Z \neq j}}^{m} x_z x_j\right] \tag{20}$$

where

$$x_z = \sum_{i=1}^{n} \frac{\theta_{iz} t_i}{v_z} \tag{21}$$

$$x_j = \sum_{l=1}^{n} \frac{\theta_{lj} t_l}{v_j} \text{ and} \tag{22}$$

$$x_k^2 = \sum_{i=1}^{n} \sum_{l=1}^{n} \frac{\theta_{ik} \theta_{lk}\, t_i\, t_l}{v_k^2} \tag{23}$$

The objective function is concluded by substituting (21), (22), and (23) in (20) that yields (24). As shown in (25) is the nonlinear objective function of HLBGA where t, *v*, *m*, and *n* are constants for each problem which represent tasks length, VMs processor speed, number of VMs, and number of tasks need to be assigned, respectively. While θ contains the assignment variables need to be solved for the optimum solution.

This objective function is subject to three constrains which are formulated in (26-28). As shown in (26) means that each task should be assigned to only one VM. θ in (27) is a binary variable which can be 1 or 0, i.e., assigned or not assigned. As shown in (28) states that, the completion time for any VM for optimum solution should be less than or equal to the makespan of the initial assignment matrix ($\text{Makespan}_{\text{initial}}$).

$$\sigma^2 = \frac{1}{m^2}\left[(m-1)\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{n}\frac{\theta_{ik}\,\theta_{lk}}{v_k^2}\ t_i\ t_l - \sum_{\substack{z=1 \\ z\ \neq\ j}}^{m}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{n}\frac{\theta_{iz}\,\theta_{lj}}{v_z\,v_j}\ t_i\ t_l\right] \tag{24}$$

$$\sigma = \sqrt{\frac{1}{m^2}\left[(m-1)\sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{n}\frac{\theta_{ik}\,\theta_{lk}}{v_k^2}\ t_i\ t_l\ -\ \sum_{\substack{z=1 \\ z\ \neq\ j}}^{m}\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{n}\frac{\theta_{iz}\,\theta_{lj}}{v_z\,v_j}\ t_i\ t_l\right]} \tag{25}$$

subject to:

$$\sum_{j=1}^{m} \theta_{ij} = 1\ \forall\ i \tag{26}$$

$$\theta_{ij} \epsilon\ \{0,1\}\ \forall i\ \forall j, \quad i = 1,2, \ldots . n \quad j = 1,2, \ldots \ldots m \tag{27}$$

$$\sum_{i=1}^{n} \frac{\theta_{ij} t_i}{v_j} \leq Makespan_{initial}\ \forall j \quad j = 1,2, \ldots, m \tag{28}$$

3.4. The HLBGA phases

The proposed HLBGA algorithm has two phases. First, HILB algorithm is proposed as a new traditional algorithm in order to distribute tasks overall VMs in an efficient way to avoid overloaded or under loaded VMs. The second phase uses the output as an initial population for GA. Figure 1 shows the main steps of the two phases of the proposed algorithm. These two phases are implemented as:

3.4.1. Phase I: Initial population phase

In this phase, the HILB algorithm is proposed in order to balance the load and minimize makespan. Algorithm strategy is based on moving tasks from heavy loaded machines to least loaded ones as:

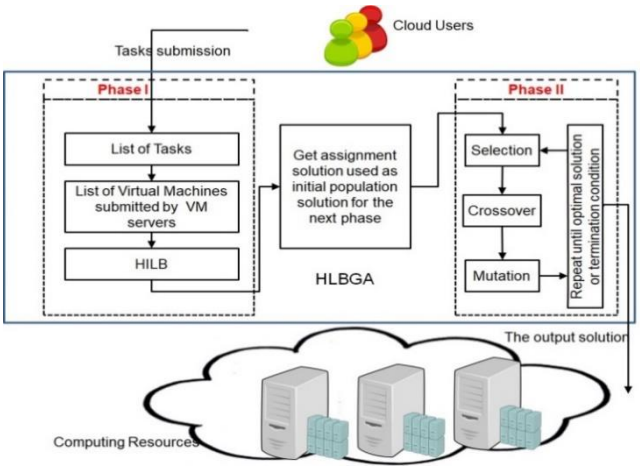

Figure 1. Flow structure of the HLBGA algorithm

a. HILB gets an initial assignment solution for all the submitted tasks over all the available resources by assigning the task with min. completion time to the corresponding machine. Then, it calculates makespan and load deviation for this initial solution as the current makespan, and load deviation, respectively.
b. HILB calculates the completion time of all the available VMs. It tries to move the shortest task in the heaviest loaded resource to the least loaded resource. HILB considers two conditions for accepting any new task movement from one VM to another. It guarantees that each new task movement is a forward step in enhancing makespan and load deviation. The two conditions are: (1) New Makespan <= Current Makespan, and (2) New load deviation <= Current load deviation.

HILB makes all the available task movements for the current heaviest loaded VM to any one of the remaining VMs. HILB repeats these previous operations on all the available resources. It balances the load overall resources even very slow ones in a way that achieves high load balancing and optimum completion time. This algorithm avoids starvation problem between VMs.

3.4.2. Phase II: GA phase

HLBGA algorithm relies on GA as a powerful solution for nonlinear programming optimization NP-complete problems. Genetics in this algorithm relies on three main operations; elite, crossover, and mutation. In Elite operation, the algorithm chooses the assignment matrices that give the best fitness functions to pass to the next generation. In crossover and mutation operations, the algorithm reassigns tasks to different VMs to form new solutions in different ways. Crossover recombines each two assignment matrices to form two new ones which practically mean reassignment of tasks to form two new solutions. The recombination must be done on a complete row basis i.e., complete rows are swapped between matrices. While in mutation, random changes done to a single assignment matrix. Algorithm 1 shows the main processes of the proposed HLBGA.

Algorithm 1: The proposed HLBGA

```
Begin
// start Phase I: HILB Algorithm
1.  For    any  submitted  task  Ti  calculate  completion  time  Ctij  for  Resourcej  Rj
Ctij=Etij+rtj;
2. while the non-submitted task list is not empty
3.    Find task I with minimum completion time and assign to corresponding Resource
4.    Remove the task from non-submitted task list  and update resource ready time rtj
5. End
6. Calculate current Makespan Mc and load deviation Lc
7. Add all VMs to Resources list   R
8. while list R not empty
9.   Find Heaviest loaded VM RH in Resource List
10.   Add other Resources to load list L and find least load Resource RL
11.   move the shortest task in the heaviest loaded resource to  RL
12.   a. IF New Makespan Mn  <= Mc  And New  Load Deviation Ln<= Lc
      b. Then Mc= Mn and Lc=Ln  And Goto step  9
      c. Else if L is not empty
      d. Then undo step 11  And remove RL from List L  And go to step 11
      e. Else remove RH  from list R   And go to step 8
13. End
```

```
    // start  Phase II: Applying GA
14. Initialize population by adding the result of phase 1 to random initial population
15. Set initial parameters
      E Elite count fraction, P population size, C Crossover fraction G number of
      generations
16. Calculate number of variables V= n×m
17. Set mutation fraction U= 1- ( E + C )
18. while termination condition not satisfied
19.    Evaluate  each chromosome using fitness function
20.     Choose (E × P) chromosomes with the best fitness function as elite for the next
         generation
21.    Select (C × P) chromosomes for  crossover operation
22.    For k=1 to ( C × P)
23.        Select two random chromosomes as input for crossover operation
24.        Perform crossover operation on selected chromosomes
25.        Select  the two output chromosomes to the next generation
26.   End For
27.    Select ( U × P )  chromosomes for  mutation operation
28.      For k=1 to ( U × P)
29.     Select one random chromosome as input for mutation operation
30.       Perform Mutation process on  the selected chromosome
31.      Select  the  output chromosome  to the next generation
32.   End For
33.    Replace the current population by new generation
34. End
```

3.5. Complexity of HLBGA

The HLBGA is based on two main phases. In the first phase, it runs the HILB. The time complexity of this phase is based on the number of the movements that performed to reach the initial population. It can be computed as: $O(n_1)$. In the second phase, the HLBGA runs the GA. The complexity in this phase can be computed as O(G×N) [35]. Comparing the time complexity of the first phase to the second phase, it was found that $n_1 << G \times N$, so it can be neglected. Therefore, the total complexity of the HLBGA algorithm is: O(G×N). The initial population that is used in the proposed algorithm helps the genetics to reach a better solution with less population size and number of generations which decreases the complexity of the algorithm. Table 2 shows the time complexity of the HLBGA and a description of the complexity parameters.

Table 2. Time complexity of the HLBGA

Algorithm	Time complexity	Description
Phase I: HILB	$O(n_1)$	n_1: Number of moves to reach the initial population
Phase II: GA	O(G×N)	G: Number of generations N: n × m × P (time overhead of all chromosomes) where n × m: Number of variables that represent the number of genes in each chromosome (time overhead of one chromosome) P: Population size (number of chromosomes in each generation)
HLBGA	O(G×N)	

4. PERFORMANCE EVALUATIONS

In this section, the performance of the proposed HLBGA algorithm is evaluated in different environments and conditions. The proposed algorithm is compared against variant techniques; Min-Min [8] and LBIMM [9] as traditional algorithms, PSO [10] with two different objective functions as metaheuristic techniques; PSO1 is the basic PSO algorithm where the objective function is to minimize the makespan while PSO2 is an updated version of the basic PSO algorithm where the objective function is to minimize the load deviation, and GA [13] as an evolutionary algorithm which is the original of the proposed algorithm. In addition, the comparison includes the proposed HILB that represents the initial population of HLBGA. The evaluation is based on the results of simulation done using CloudSim [35].

4.1. Simulation overview

CloudSim is a simulation tool that simulates the behavior of load balancing algorithms when run on real data centers. It was used to test the performance of the proposed algorithm and compare the results with the other algorithms in terms of makespan, resource utilization, and load standard deviation [25]. Table 2 shows the CloudSim configuration for the four simulations used to test the behavior of the proposed algorithm in different running conditions. Each simulation was run 10^5 times and the average was considered in the results. The parameters of GA and PSO are shown in Table 3.

Table 3. CloudSim configurations

		Simulation_1	Simulation_2
Number of Datacenters		1	1
Number of Hosts		1	1
Number of VMs		4	5
Number of Tasks		10: 150	15
Task length (MI)		200: 3000	150:300
VM Scheduler policy		Time shared	
Cloudlet Scheduler policy		Space shared	
GA algorithm parameter setting	Parameter	Value	
	Crossover	0.8	
	Elite	0.05	
	Max. number of generations	200	
	Population size	min.(10×number of genes, 250)	
PSO algorithm parameter setting	Parameter	Value	
	Maximum iterations	200	
	C_1, C_2	1.49445	
	K	5	
	ω_{min}, ω_{max}	0.1, 0.9	
	Population size	20	

4.2. Impact of increasing the workloads with fixed resources

In this case, the number of tasks is increased while the number of VMs is fixed to check the algorithm's behavior in different workloads on the same resources. The simulation parameters of Simulation 1 are shown in Table 3. The number of tasks is varying from 10 to 150. The tasks have different lengths as happen in real-world workloads. They were generated randomly at the range from 200 to 3000 (MI). Four VMs were considered for the simulation. The evaluation metrics are makespan, resource utilization, and load deviation.

Figure 2 shows the makespan comparison of the proposed HLBGA with the intended algorithms. It is shown that HLBGA minimizes the makespan comparing with the other algorithms. The makespan improvement of HLBGA over HILB and GA is up to 15.7% and 71%; respectively. Figure 3 shows the load deviation comparison for Simulation 1. It can be seen that the load deviation of the proposed HLBGA is minimized when compared with the other algorithms. The load deviation improvement of HLBGA over HILB and GA is 28.5% and 96.1%, respectively in the case of 150 tasks. Figure 4 shows a resource utilization comparison for Simulation 1. It is shown that the resource utilization of the proposed HLBGA is maximized when compared with other algorithms. The increase in the utilization of the proposed HLBGA over HILB and GA is 1.8% and 67.4%, respectively in the case of 150 tasks.

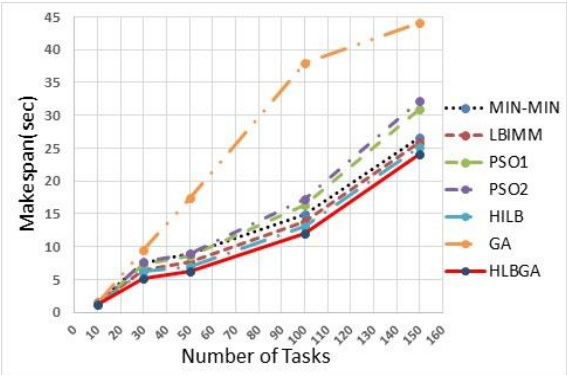

Figure 2. Makespan versus number of tasks

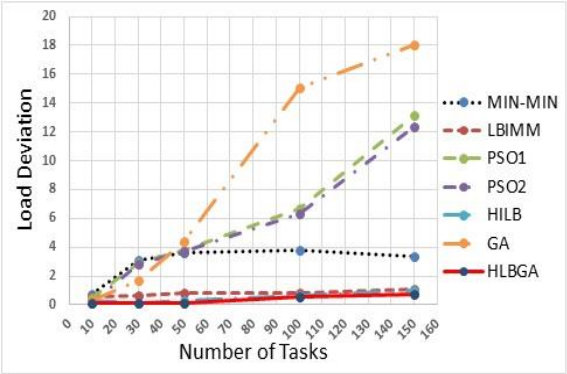

Figure 3. Load deviation versus number of tasks

The results show that the performance of the metaheuristic algorithms such as PSO1, PSO2, and GA is much lower than the performance of the traditional algorithms at a large number of tasks. With increasing in the number of tasks, HILB introduces a good performance than the other traditional algorithms so it can be used to produce an initial population for GA to form the proposed HLBGA. The proposed HLBGA algorithm as a hybrid technique between HILB and GA outperforms the other algorithms. The makespan, load deviation, and utilization improvement of HLBGA over HILB and GA are 8% and 48.3%, 34.3% and 85%, and 3.4% and 40%, respectively.

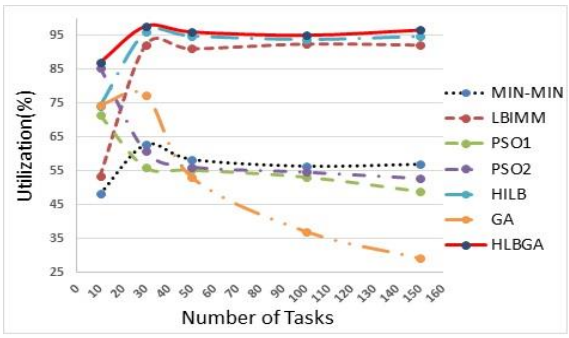

Figure 4. Resource utilization versus number of tasks

4.3. Implementations in homogeneous and heterogeneous environments

In this case, the simulation is implemented on a fixed number of Cloudlets and VMs but the speed of VMs are changed to test the performance of the algorithms in Homogeneous (Homog), Heterogonous-high (Het-high) and Heterogeneous-low (Het-low) processors. The simulation parameters of Simulation 2 are shown in Table 3. Three simulations were run with different VM speed environments. In Homogenous, all the VMs have the same speed. In Heterogeneous-low, the speed variation among VMs is low with ratio 1:2.5 between lowest and highest speed VM while in Heterogeneous-high, simulation a high-speed variation among all VMs with ratio 1:7 is considered.

The target of this experiment is to test the proposed algorithm behavior in the case of workloads with different lengths in varying environments. Figure 5 shows a makespan comparison of the proposed HLBGA algorithm with the LBIMM, HILB, standard GA and PSO algorithms while the simulation environment varies from homogeneous to heterogeneous. It is shown that the makespan improvement of HLBGA over HILB and GA is up to 2.6% and 42.5%, respectively. Figure 6 shows a load deviation comparison of Simulation 3. It can be seen that the load deviation of the proposed algorithm is minimized when compared with the other algorithms. Figure 7 shows the utilization comparison of Simulation 3. It is clear that the utilization of the proposed algorithm is maximized when compared with the other algorithms.

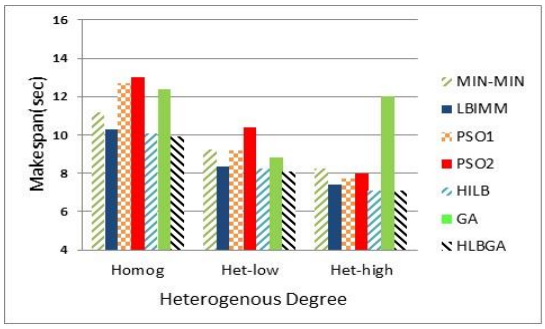

Figure 5. Makespan in different environments

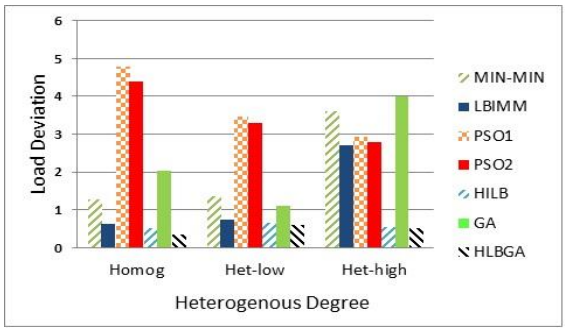

Figure 6. Load deviation in different environments

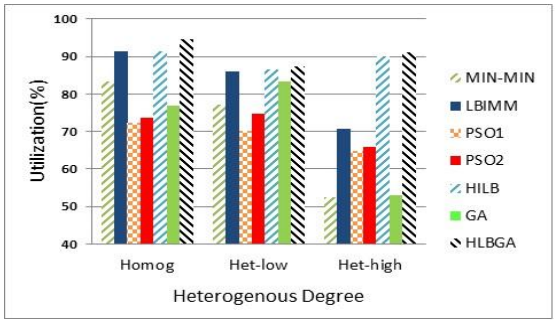

Figure 7. Utilization in different environments

The results show that GA works better than the other metaheuristic algorithms, and also HILB is more powerful in load balancing than the other traditional algorithms. The proposed HLBGA algorithm performs better than the other algorithms in all cases especially in Heterogeneous-high which gives the best results compared to the other algorithms.

5. CONCLUSION AND FUTURE WORK

In this paper, HLBGA algorithm is proposed. It is implemented in two phases. In the first phase, HILB scheduling algorithm is proposed to perform a good task scheduling process in order to improve the makespan and produce a good initial population to the second phase. In the second phase, GA as an evolutionary-based algorithm is used with a newly formulated fitness function in the way of reaching the optimal load deviation. The proposed algorithm is tested on two simulations. The first simulation tests the effect of increasing the workloads on the same number of VMs. The simulation results show that the proposed HLBGA outperforms the other standard and metaheuristic algorithms; Min-Min, LBIMM, GA and PSO. The second simulation tests the algorithm behavior in the case of distributing tasks of different lengths on resources that have one of three cases: the same speed (Homogeneous), a slight difference in the speeds (Heterogeneous-low), and a large variation in the speeds (Heterogeneous-high). The simulation results show that the proposed HLBGA outperforms all the other algorithms especially in Heterogeneous-high case.

This study focuses on the processor speed of VMs since it is the most effective factor, while other factors such as memory size and bandwidth of VMs are constants. In future work, the performance of the proposed algorithm with more other conditions will be investigated. Also, integrating a machine learning technique with the proposed algorithm adds a new value and can be tested.

REFERENCES

[1] J. Shen, T. Zhou, D. He, Y. Zhang, et al., "Block design-based key agreement for group data sharing in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 5971, no. c, pp. 1-15, 2017.

[2] H. Nashaat, N. Ashry, and R. Rizk, "Smart elastic scheduling algorithm for virtual machine migration in cloud computing," *The Journal of Supercomputing, Springer*, vol. 75, no. 7, pp. 3842-3865, 2019.

[3] E. Ghomi, A. Rahmani, and N. Qader, "Load-balancing algorithms in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 50-71, 2017.

[4] M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, "Bio-inspired load balancing algorithm in cloud computing," in *Proc. The International Conference on Advanced Intelligent Systems and Informatics (AISI)*, Cairo, Egypt, Chapter 54, 2017, pp. 579-589.

[5] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 11, no. 12, pp. 5694-5711, 2017.

[6] M. Ala'Anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: a meta-study," *IEEE Access*, vol. 7, pp. 141868-141887, 2019.

[7] W. E. Saber, R. Y. Rizk, W. M. Moussa, and A. M. Ghonem, "LBSR: Load balance over slow resources," in *Proc. IEEE 1st International Conference on Computer Applications & Information Security (ICCAIS*), Riyadh, Saudi Arabia, 2018, pp. 1-7.

[8] Y. Shi and K. Qian, "LBMM: A load balancing based task scheduling algorithm for cloud," *in Proc. Information and Communication Conference*, Springer, Cham, 2019, pp. 706-712.

[9] S. Abdolhosseini and M. T. Kheirabadi, "Scheduling independent parallel jobs in cloud computing: A Survey," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 11, no. 3, pp. 11-21, 2019.

[10] H. Saleh, H. Nashaat, W. Saber, and H. Harb, "IPSO Task scheduling algorithm for large scale data in cloud computing environment," *IEEE Access*, vol. 7, pp. 5412-5420, 2018.

[11] X. Liu, Z. Zhan, J. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Transactions on Evolutionary Computation*, vol. 22, pp. 113-128, 2016.

[12] H. Xing, F. Song, L. Yan, and W. Pan, "A modified artificial bee colony algorithm for load balancing in network-coding-based multicast," *Soft Computing*, vol. 23, no. 15, pp. 6287-6305, 2019.

[13] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (GA) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340-347, 2013.

[14] H. Xue, K. T. Kim, and H. Y. Youn, "Dynamic load balancing of software-defined networking based on genetic-ant colony optimization," *Sensors*, vol. 19, no. 2, p. 311, 2019.

[15] M. Kalra, and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal Cairo University*, vol. 16, no. 3, pp. 275-295, 2015.

[16] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," *Journal of Network and Computer Applications*, vol. 128, pp. 64-77, 2019.

[17] M. Sharma, G. Singh, and R. Singh, "Clinical decision support system query optimizer using hybrid Firefly and controlled Genetic Algorithm," *Journal of King Saud University-Computer and Information Sciences*, 2018.

[18] I. K. Gupta, V. Yadav, and S. Kumar, "Medical data clustering based on particle swarm optimisation and genetic algorithm," *International Journal of Advanced Intelligence Paradigms*, vol. 14 no. 3-4, pp. 345-358, 2019.

[19] S. K. Roy and D. De, "Genetic algorithm based internet of precision agricultural things (IopaT) for agriculture 4.0," *Internet of Things*, pp. 1-19, 2020.
[20] P. Kaur and M. Sharma, "Diagnosis of human psychological disorders using supervised learning and nature-inspired computing techniques: A meta-analysis," *Journal of medical systems*, vol. 43, no. 7, pp. 1-30, 2019.
[21] S. Sayed, M. Nassef, A. Badr, and I. Farag, "A nested genetic algorithm for feature selection in high-dimensional cancer microarray datasets," *Expert Systems with Applications*, vol. 121, pp. 233-243, 2019.
[22] K. Sureshkumar and V. Ponnusamy, "Power flow management in micro grid through renewable energy sources using a hybrid modified dragonfly algorithm with bat search algorithm," *Energy*, vol. 181, pp. 1166-1178, 2019.
[23] N. T. Hanh, H. T. T. Binh, N. X. Hoai, and M. S. Palaniswami, "An efficient genetic algorithm for maximizing area coverage in wireless sensor networks," *Information Sciences*, vol. 488, pp. 58-75, 2019.
[24] A. M. S. Kumar and M. Venkatesan, "Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment," *Wireless Personal Communications*, vol. 107, no. 4, pp. 1835-1848, 2019.
[25] C. V. Raja and D. L. Jayasimman, "A Cost Effective Scalable Scheme for Dynamic Data Service in Heterogeneous Cloud Environment," *International Journal of Advanced Science and Technology*, vol. 28, no. 20, pp. 764-776, 2019.
[26] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp. 149-158, 2020.
[27] X. Cheng, D. Ciuonzo, and P. S. Rossi, "Multibit decentralized detection through fusing smart and dumb sensors based on RAO test," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 2, pp. 1391-1405, 2019.
[28] J. G. Jamnani and M. Pandya, "Coordination of SVC and TCSC for management of power flow by particle swarm optimization," *Energy Procedia*, vol. 156, pp. 321-326, 2019.
[29] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment," *Journal of King Saud University-Computer and Information Sciences*, 2019.
[30] W. Bei and L. Jun, "Load balancing task scheduling based on multi-population genetic algorithm in cloud computing," in *Proc. the 35th Chinese Control Conference*, Chengdu, China, 2016, pp. 27-29.
[31] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," *IEEE Access*, vol. 7, pp. 42735-42744, 2019.
[32] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University-Computer and Information Sciences*, 2020.
[33] F. Yao et al., "An intelligent scheduling algorithm for complex manufacturing system simulation with frequent synchronizations in a cloud environment," *Memetic Computing*, vol. 11, no. 4, pp. 357-370, 2019.
[34] J. E. Gentle, "Matrix algebra, Theory, Computations, and Applications in Statistics," *Springer-Verlag*, New York, 2007.
[35] J. D. Pagare and N. A. Koli, "Design and simulate cloud computing environment using cloudsim," *International Journal of Computer Technology & Applications*, vol. 6, no. 1, pp. 35-42, 2015.

BIOGRAPHIES OF AUTHORS

Walaa Saber is an assistant professor in the Electrical Engineering Department, Port Said University, Egypt. She received her B.Sc. and M.Sc. degrees in Computer and Control Engineering, Suez Canal University in 2001 and 2008, respectively. Her Ph.D. degree in computer and control engineering is taken from Port Said University in 2014. Her research interests are in the area of computer networking, including cloud computing, clustering, and Internet of Things.

Walid Moussa received the B.Sc. and M.Sc. degrees in Computer Engineering from Suez Canal University, Ismailia, Egypt, in 2001, and 2019, respectively. From 2001 to 2009, he was a system engineer in Suez Canal Authority, Ismailia, Egypt. His work includes the administration of vessel traffic management system. Since 2009, he has been project manager for navigational control projects with the transit department, Suez Canal Authority, Ismailia, Egypt.

Atef M. Ghuniem is a Professor emirate at Electrical Engineering Department Department, Suez Canal University, Egypt, He received the B.Sc. and M.Sc. in E. E. from Military Technical College (MTC), Cairo, Egypt, in 1971 and 1979, respectively. He received the Ph.D. in E.E. (major area: Electronics and Waves) from George Washington University, Washington, D.C., USA, in 1985. He joined the staff of the E.E. department in MTC from 1975 till 1996. He was the head of department during the period 1995-1996. Since 1998 till 2008 he is an associate professor at the E.E. department, Faculty of engineering, Sues Canal University and from 2008 till now be a professor emirate at the same department. His research interest is in wireless communications including antennas and wave propagation, passive and active microwave devices, and microwave communications. Recently, he is interested in information technology.

Rawya Rizk is a Professor of Computers and Control in Electrical Engineering Department, Port Said University, Egypt. She is the Head of Electrical Engineering Department, Port Said University, 2017 till now. She is the Chief Information Officer (CIO) of Port Said University (PSU), 2014 till now. She received her BSc, MSc and PhD in Computers and Control Engineering from Suez Canal University in 1991, 1996 and 2001, respectively. Her research interest is in computer networking, including mobile networking, wireless, ATM, Sensor Networks, Ad Hoc Networks, QoS, traffic and congestion control, handoffs and cloud computing. She is a reviewer in many of international communication and computer journals such IEEE Access, IET communications, IET sensors, IET Networks, Journal of Supercomputing, Journal of Network and Computer Applications, Computers & Electrical Engineering, Telecommunication Systems (TELS), and Mathematical Problems in Engineering.