Load Balancing based on Bee Colony Algorithm with Partitioning of Public Clouds

Pouneh Ehsanimoghadam
Department of Computer,
Germi Branch, Islamic Azad University
Germi, Iran

Mehdi Effatparvar*
Department of Computer
Ardabil Branch, Islamic Azad University
Ardabil, Iran

***Abstract*—Cloud computing is an emerging trend in the IT industry that provides new opportunities to control costs associated with the creation and maintenance of applications. Of prevalent issues in cloud computing, load balancing is a primary one as it has a significant impact on efficiency and plays a leading role in improved management. In this paper, by using a heuristic search technique called the bee colony algorithm, tasks are balanced on a virtual machine such that their waiting time in the queue is minimized. In the proposed model, the cloud is partitioned into several sectors with many nodes as resources of distributed computing. Furthermore, the indices of speed and cost are considered to prioritize virtual machines. The results of a simulation show that the proposed model outperforms prevalent algorithms as it balances the prioritization of tasks on the virtual machine as well as the entire cloud system and minimizes the waiting times of tasks in the queue. It also reduces the completion time of tasks in comparison with the HBB-LB, WRR, and FCFS algorithms.**

***Keywords*—Cloud computing; load balancing; bee colony algorithm; public cloud; cloud partitioning**

I. INTRODUCTION

Cloud computing provides ways of presenting IT services in a similar manner to public utility companies. In simple terms, cloud computing is a new approach to use computing resources. The cloud is a group of distributed nodes that supply resources, hardware, and software over the network based on user demand. The increasingly strong presence of such companies as Microsoft, Google, and Amazon in the arena of cloud computing indicates its rapid development and influence on IT.

New changes or concepts in the technological world can lead to problems and complications, and cloud computing is no exception. It poses many challenges to experts in the field. One of these is load balancing in the cloud. Load balancing in is crucial in computer science, and has attracted a considerable amount of research. Many techniques have been employed for load balancing, such as the genetic algorithm, bees' algorithm, neural networks, and distributed research. Load balancing algorithms make decisions about allocating resources to tasks and coordinating among them. The aim of load balancing is to share resources among tasks within the system such that for every resource, there are an equal number of tasks to be completed, and this minimizes the total time needed [3]. Algorithms for load balancing and resource management can be categorized into three groups (Wu, Wang & Xie, 2013 [7]; Yan, Wang, Chang & Lin, 2007 [9]). The first consists of algorithms for static load balancing. In such algorithms, decisions concerning load balancing are made at compile time. The advantage of static load balancing is its simple implementation and low overhead, as there is no need to permanently monitor nodes to assess the efficiency of the system. These algorithms work well when there are small changes in loads in virtual machines. Therefore, they are not appropriate for cloud and grid computing environments because the load on the network is variable at every point of time in such environments. The second category of load balancing and resource management techniques consists of algorithms of dynamic load balancing. In these algorithms, the distribution of load among nodes changes, and they use the given information to make decisions about load distribution. The third category consists of hybrid algorithms, which involve the hybrid use of static and dynamic algorithms, and switch between them when necessary.

There are many papers have been proposed based on optimization in them [10]-[19] and [24]-[26]. These papers tried to optimize their problems by presented some formal methods and fitness functions in them. In [18] authors presented new distributed method to reducing the energy for the communication between nods and coordinator. Also in [24], [25] and [26] authors used the optimization methods in Meso-Scaled material. Saffar Ardabili and Aghayi [20] evaluated efficiency score of decision making units (DMUs) by the undesirable outputs. Aghayi et al. [22] measured efficiency measure using common set of weights in present of uncertainty based on robust optimization. Aghayi [23] proposed the approach to obtain cost efficiency of DMUs by fuzzy data. Aghayi and Maleki [21] measured the efficiency of bank branches of Ardabil, Iran using robust optimization theory and undesirable outputs. Rostamy-malkhalifeh, and Aghayi [27] suggested the method for calculating overall profit efficiency using uncertainty as fuzzy in data. Aghayi and Ghelejbeigi [28] presented the improvement of cost efficiency based on resource allocation. Aghayi [29] computed revenue efficiency of DMUs with undesirable and fuzzy data. Salehpour and Aghayi [30] calculated the most revenue efficiency with price uncertainty.

In this study, a method for cloud partitioning is proposed that is also used to investigate the load on systems in heterogeneous environments, with the aim of reducing the time needed for scheduling and other tasks. The proposed method is dynamic. In this method, the algorithms of load balancing can

represent each resource according to its capabilities and accessibility to tasks services, which enhances the efficiency of the cloud system. To balance the load, load balancing as used in cloud computing environments is used, inspired by the HBB-LB algorithm (Babu & Krishna, 2013 [1]).

The rest of this paper is organized as follows: in Section 2, prevalent scheduling algorithms are introduced. Section 3 describes the proposed scheduling algorithm, Section 4 details the simulation to test it and the results and Section 5 contains the conclusions of this study and recommendations for future work in the area.

II. REVIEW OF LOAD BALANCING ALGORITHMS

One method for load balancing involves the efficient use of virtual machines. It was proposed by Domanal and Reddy (2013) [2], and allows for load distribution on accessible virtual machines to guarantee that stable use of resources or virtual machines in the cloud system. This is in contrast to active load balancing, which involves the proper distribution of load within the system to solve the problem of inefficient use of virtual machines in other algorithms. However, the service time following load balancing has not been yet studied. In 2013, Babu and Krishna [1] proposed a load balancing algorithm inspired by the food-finding behavior of honey bees, and is used on the Web. The aim is to reach a balanced load within virtual machines by maximizing capability. Moreover, it balances the prioritization of tasks in virtual machines such that the waiting times of tasks in the queue are minimized. However, this algorithm is impractical for dependent tasks. Ren, Lan, and Yin (2012) [5] proposed a dynamic load balancing algorithm according to the migration of virtual machines within the cloud computing environment. This algorithm contains a unit to monitor excessive loads, one for diagnosis, and a unit for load scheduling. The unit of load monitoring is used to collect the load information pertaining to a group of virtual machines and the resources' server (calculating the load and updating it). The database information for this algorithm is collected according to the trigger strategy based on fractal methods. It determines the time of migration from an overloaded virtual machine in the system. In this method, operating capability is maximized by using unemployed nodes in the system. Moreover, the overload in load balancing systems is minimum. However, in this method, only the load is studied. TeraScaler ELB was proposed by Wu et al. (2013) [7] based on the prediction of elastic load balancing for resource management in cloud computing. In this algorithm, virtual machines are added or removed according to the analysis and prediction of the given load and its history. The algorithm of ELB resource management is regularly implemented through two events:

- The load balancer regularly collects resource information from the back-end server.
- The load balancer determines whether there is a request to remove the back-end virtual machine according to the collected resource information from the back-end server.

In 2012, Nishant and et al. [4] proposed an algorithm for the load distribution of workloads among the nodes of a cloud using ant colony optimization. According to this study, ants can move in two directions: forward and backward. In the forward direction, if an ant faces overloaded nodes, they will move forward. In the backward direction, if an ant faces an overloaded node, which has faced an under-loaded node, it will move backward. The main duty of ants is to redistribute tasks. In this approach, the ants repeatedly update their pheromones during all their moves. They also identify the tasks of nodes and find their way among different types of nodes. In 2013, Xu, Pang, and Fu [8] proposed a load balancing model based on cloud partitioning for public clouds in different geographical locations. This method renders load balancing easier in extremely large and complicated environments. Clouds have a main controller to choose the proper sectors of tasks, and the balancer chooses the best strategy for load balancing per sector of the cloud. Sectors can be unemployed, normal, or overloaded. The sector of load balancing decides how to assign tasks to nodes of normal or unemployed sectors. In this algorithm, the features of virtual machines are not considered. Soni and Kalra (2014) [6] proposed a central load balancer to balance loads among virtual machines in a cloud data center. This algorithm distributes the load among heterogeneous virtual machines based on hardware configuration and their states in the cloud data center. This method can balance the load quickly and reliably in cloud computing environments by using all virtual machines based on their calculation capacities. The central load balancer communicates with all users and virtual machines, which are presented in cloud data center through a data center controller, which also analyzes the values' table containing the identifies, states, and priorities of virtual machines. It searches for the virtual machine with the highest priority to allocate user requests. The data center controller allocates the requests to the identifier of the virtual machine as presented by the central load balancer.

By studying research on the load balancing of tasks and resources, it can be concluded that more research has been conducted on load balancing in heterogeneous environments. Existing algorithms have some drawbacks in the cloud sample, and this study attempts to alleviate some of them with solutions for responding to requests quickly and managing virtual machines properly.

III. PROPOSED METHOD

In this method, load balancing in cloud computing environments, inspired by the algorithm that mirrors the food-finding behavior of honey bees (HBB-LB), is used. This algorithm not only balances load, but also considers the priority of tasks removed from virtual machines due to overload. The techniques of load balancing are effective for reducing the time needed to answer and service requests. The load balancing of non-exclusive independent tasks on virtual machines is an important aspect of task scheduling in cloud computing. The load on virtual machines must be distributed on balance, so that the machine is used efficiently.

In the ABC algorithm, several species of bees act in a research atmosphere. The bee that is randomly chosen to act to search is called the scout bee. It determines the location of sources of food and nectar.

A. Scheduling System Model

As in the cloud computing environment, we encounter a large space with several users and service providers, tasks are not predictable, and the capacity of each virtual machine is different. In the proposed method, the cloud is partitioned into several sectors. When the environment is very large, balancing the load within the entire system is difficult. In this method, to balance the load in smaller sectors, the load balancing algorithm inspired by the behavior of honey bees is used.

The aim of load balancing algorithms is to balance the load among virtual machines to maximize operating capability. The proposed algorithm balances tasks on virtual machines as well as the entire cloud system, so that the waiting time of tasks in the queue is minimized.

Fig. 1 depicts the proposed method. The user presents tasks to the cloud system, which contains a data center for independent tasks. The tasks are presented to the system and, prior to execution, the processing time of each task is calculated (or the processing times of tasks are estimated through mathematical models) and the characteristics of all tasks are identified. The service provider has a controller and a state table that records the features of all virtual machines. Moreover, in the controller sector of the load of each virtual machine, the system is divided into three general sectors (overloaded, under-loaded, and balanced sectors). If a virtual machine loses its load while performing tasks, it can be moved from one sector to another. Tasks are assigned based on the magnitude of loads of the virtual machines and processing times. Moreover, the indices of speed and cost are considered for the prioritization of the virtual machines. In the following, all stages of this process are discussed.

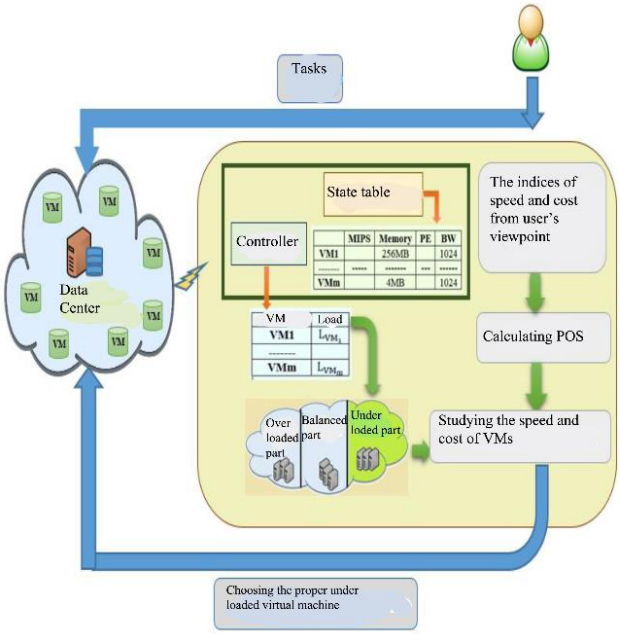

Fig. 1. The architecture of tasks.

B. Mechanism of Partitioning the Cloud System

In the cloud model, an infrastructure is considered an IaaS service that provides users with virtual resources. The cloud is partitioned into several sectors. A cloud may contain a large number of nodes in different geographical locations. Partitioning the cloud leads to better use. Fig. 2 lists details of a cloud system that has been partitioned into several separate sectors. It is worth mentioning that each area can be partitioned into several sectors. Heterogeneous virtual machines have been used in various areas, and are managed by a central controller. The methods of management may vary by cloud service provider.

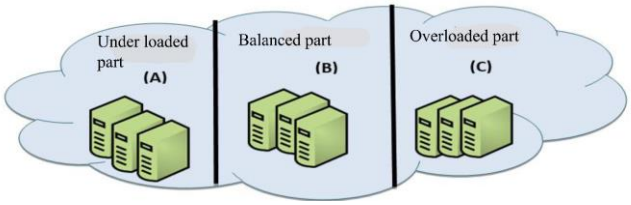

Fig. 2. Partitioning the cloud into three sectors.

When the environment is large, partitioning based on load balancing limits the search environment for assigning tasks. The cloud has a main controller that selects appropriate sectors for input tasks. The balancer per sector of the cloud selects the best strategy for load balancing (Xu et al., 2013) [8]. The load states of all virtual machines per data center are stored in the central controller, which controller deals with information for each sector, collects the information of each node, and selects the best strategy for partitioning virtual machines and assigning tasks. This information is updated repeatedly. When a task is entered into the cloud, the first stage involves selecting the proper sector. To partition the data center, it is necessary to calculate the load of each virtual machine. The controller searches sectors of the cloud and investigates the overloaded, underloaded and balanced states. Therefore, the cloud environment is partitioned into overloaded, underloaded, and balanced sectors. The method of calculating load per virtual machine is explained in Table I. Following partitioning, the tasks are presented to the underloaded sectors. To prioritize virtual machines, speed and cost are used. It is worth mentioning that after each stage of assigning tasks, the load on the system is updated. In this cloud system, for each sector, a processor is used to monitor and investigate the load on the virtual machines. If the magnitude of load on a virtual machine changes, it is moved to another sector. The values of $L_{VMk,t}$ and $L_{VMk,t+1}$ indicate the loads on virtual machines at times t and t + 1, respectively. Load differences in a time interval may show a change in the load on the virtual machine in overloaded or balanced sectors. When diffL is zero, there are no load changes in virtual machines and no changes in the sectors. However, if it is lower than zero, virtual machines are moved from one sector to another.

TABLE I. PSEUDO CODE USED TO DETERMINE CHANGES IN LOAD

The pseudo code used to determine changes in load
1) diffL = $L_{VMk,t+1} - L_{VMk,t}$
2) If (diffL = 0) no change in the magnitude of load
3) Else if (diffL < 0) the virtual machine can be moved from one sector to another.
4) End

In Table II, the parameters used in this study are introduced.

TABLE II. DEFINITIONS OF THE PARAMETERS IN THE EQUATIONS

Variables	Definitions of the Variables
CT_{max}	Maximum completion time of task
CT_{ij}	Completion time of task i on machine j
P_{ij}	Processing time of task T_i by virtual machine VM_j
Length T_i	Length of task i
Pe_{numj}	Number of processors in VM_j
Pe_{mipsj}	Million instructions per second of all processors in VM_j
P_j	Processing time of all tasks in VM_j
C_j , C	Processing capacity of VM_j and optimal processing capacity
VM_{bwj}	Communication bandwidth capability of VM_j
$L_{VMj,t}$	Load of VM_j at time t
N(T,t)	Number of requests per period
$S(VM_j,t)$	Speed of service
L , L_{VMj}	Load on all virtual machines in a data center, load on VM_j
PT_j , PT	Processing time of VM_j and processing times of all virtual machines
δ	Standard deviation
Pos_j	Priority per node
$Speed_j$	Speed of VM
$Cost_j$	Cost of VM
α	Importance of speed index
β	Importance of cost index

C. Calculating Processing Time

In HBB-LB, it is hypothesized that VM = {VM_1, VM_2, VM_3… VM_m} is a collection of m virtual machines with no links and in parallel, where they must process n tasks. Tasks are shown as a collection T = {T_1, T_2, T_3, … , T_n}. Independent tasks are non-exclusive, and are scheduled on virtual machines. A collection of virtual machines for processing tasks are an underloaded collection of virtual machines in the data center. Makespan is the time taken for task completion, and is shown in (1) (Babu & Krishna, 2013):

1. MakeSpan = max{CTij|i ϵ T , I = 1,2,…,n and j ϵ VM , j = 1,2,…,m }

The processing time of task i on virtual machine j is P_{ij}, and is calculated through (2) (Li, Xu, Zhao, Dong & Wang, 2011). The processing time of all tasks on virtual machine j is P_i, and is obtained through (3):

2. $P_{ij} = \frac{\text{length } T_i}{Pe_{numj} \times Pe_{mipsj}}$ I = 1, 2, … , n j = 1, 2, … , m
3. $P_j = \Sigma^{n}_{i=1} P_{ij}$ I = 1, 2, … , n j = 1, 2, … , m

The processing times of all tasks on a virtual machine must be smaller than or equal to their completion times. Therefore, by minimizing CT_{max}, (4) is obtained:

4. $\Sigma^{n}_{i=1} P_{ij} \le CT_{max}$ j = 1, 2, … , m

According to (3) and (4), (5) is obtained as

5. 3 and 4$\Rightarrow P_j \le CT_{max}$ j = 1, 2, … , m
6. $CT_{max} = \{\max^{n}_{i=1} CT_i, \max^{m}_{j=1} \Sigma^{n}_{i=1} P_{ij}\}$

D. Calculating Capacity of Virtual Machines

The capacity of a given virtual machine as well as all virtual machines is calculated through (7) and (8) in conjunction with the HBB-LB algorithm. The total capacity of all virtual machines is equal to the capacity of the data center. C_j is the capacity of virtual machine j and C the capacity of all virtual machines. Moreover, Pe_{numj}, Pe_{mipsj}, and VM_{bwj} respectively indicate the number of processors in virtual machine, millions of instructions for all VM_j and the bandwidth of VM_j.

7. $C_j = Pe_{numj} \times Pe_{mipsj} \times VM_{bwj}$
8. $C = \Sigma^{m}_{j=1} C_j$

E. Calculating Load on Virtual Machines

Load includes all tasks assigned to a virtual machine. A problem in cloud systems that has negative effects on them is unbalanced loads. Heterogeneous and unequal distributions of loads among virtual machines of cloud systems can create this problem. As some processors may be overloaded and others unemployed, load balancing increases the efficiency of distributed systems. This happens when nodes with heavy loads are moved to other nodes for processing. The proposed algorithms for load balancing are inventive and varied.

The load on a virtual machine can be calculated as the number of tasks at time t in a queue in virtual machine j divided by the servicing speed of virtual machine j at time t. The magnitude of the load on all virtual machines in a data center is calculated through (10) (Babu & Krishna, 2013) [1]:

9. $L_{VMj,t} = \frac{N(T,t)}{S(VM_j,t)}$
10. $L = \Sigma^{m}_{j=1} L_{VMj}$

The processing time of a virtual machine, the processing times of all virtual machines, and the standard deviation of load are calculated through (11), (12), and (13), respectively:

11. $PT_j = \frac{L_{VMj}}{C_j}$
12. $PT = \frac{L}{C}$
13. $\delta = \sqrt{\frac{1}{m}\Sigma^{m}_{j=1}(PT_j - PT)^2}$

If δ for a virtual machine is equal or lower than [0–1] (δ ≤ Ts), the system is balanced. Otherwise, it is unbalanced, and may or not have extra load. If the magnitude of load in a virtual machine is greater than the permissible capacity, the virtual machine is overloaded and load balancing is impossible. When the load on a system is balanced, tasks are assigned to virtual machines. Table III shows the pseudo code of the load balancing algorithm.

TABLE III. PSEUDOCODE OF LOAD BALANCING ALGORITHM

Pseudocode of load balancing algorithm
Input: the set task; the set VM.
While there are tasks in the list **do**
for all VMs of a host **do**
The C_i and $L_{VMj,t}$ of every VMs are calculated.
The PT_i and PT of every VMs are calculated.
The σ of every VMs is calculated.
If $\sigma \leq T_s$ **Then**
System is balanced, and Send Task to Partition;
The Pos_j of every VMs is calculated;
The P_j and CT_i of every tasks are calculated.
If Pos_{jmax} and CT_{max}
Assign $task_i$ to VM
Exit
If L > maximum capacity
Load balancing is not possible
Else
Trigger load balancing
end for
end while

F. Calculating Priority of Virtual Machines

Following partitioning, the priority of under loaded virtual machines is calculated through (14). The priority per virtual machine is set according to speed and cost. POS, Speed, and Cost, respectively, indicate the priority per node in the virtual machine, the speed per node in the virtual machine, and the cost per node in it. The coefficients α and β indicate the importance of the speed and cost indices, respectively. As users assign varying priorities to indices of α and β, their values change in the interval [1, –1].

14. $Pos_j = \alpha^* Speed_j + \beta^* Cost_j$

IV. EVALUATION AND RESULTS OF SIMULATION

The accuracy of the proposed method was tested (according to a CloudSim simulation) and the performance of the HBB-LBP algorithm assessed in comparison with the HBB-LB, FCFS, and WRR algorithms. For the simulation, a data center and four groups of tasks of 10, 20, 30, and 40 were used. Table IV lists the values of the simulation.

TABLE IV. VALUES OF SIMULATION

Type	**Parameters**	**Value**
Data center	Number of data centers	1
	Number of hosts	3
Virtual machine	Number of virtual machines	25
	Value of misp in every processor	250–2000
	Bandwidth	10000–1000
Tasks	Number of tasks	10–40
	Length of tasks	Random
	Number of needed processors	1–4

Fig. 3 shows the average time for task completion before and after load balancing in the HBB-LBP algorithm (the proposed algorithm) with different numbers of tasks. The x-axis shows the number of tasks and the y-axis their completion times in seconds. The completion times of tasks continued to decrease. If the number of tasks increased, the algorithm exhibited better performance. Fig. 4 shows a comparison between the completion times of tasks for HBB-LBP, HBB-LB, FCFS, and WRR. The x-axis shows the number of tasks and the y-axis their completion times. According to the results, the HBB-LBP algorithm was the most efficient. Fig. 5 shows a comparison between the average times of response of HBB-LBP, HBB-LB, FCFS, and WRR.

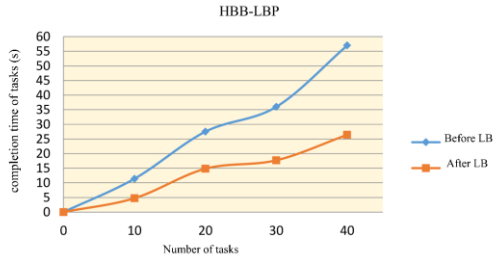

Fig. 3. Completion times of tasks before and after load balancing in the proposed algorithm.

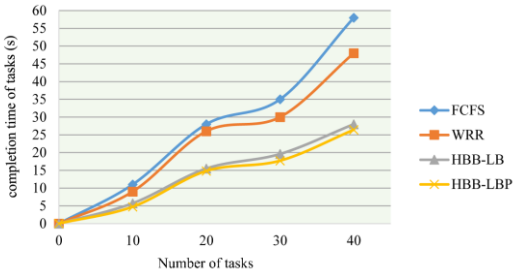

Fig. 4. Comparison among the completion times of tasks in the algorithms.

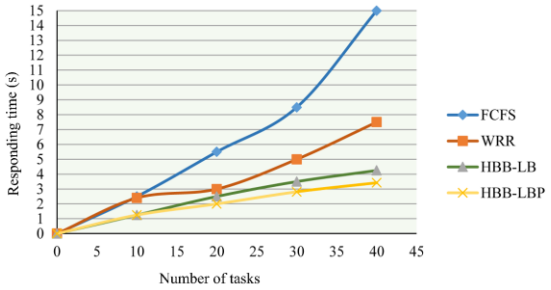

Fig. 5. Comparison among average times of response by the algorithms.

In general, the HBB-LBP method reduced the completion times of tasks and their response times in comparison with the three other algorithms. According to Table V, it reduces completion times by 9.02%, 44.08%, and 51.97% compared with HBB-LB, WRR, and FCFS, respectively.

Moreover, HBB-LBP reduced response time by 13.80%, 44.34%, and 63.94% in comparison with HBB-LB, WRR, and FCFS, respectively.

TABLE V. INDICES USED TO ASSESS THE PROPOSED ALGORITHM (HBB-LB) AND THE PERCENTAGE OF IMPROVEMENT

Algorithms / **Evaluations**	**FCFS**	**WRR**	**HBB-LB**
Completion times of tasks	51.97	44.08	9.02
Response times	63.94	44.34	13.80

V. Conclusion and Future Research

In this study, a method to partition a cloud system and investigate system load in heterogeneous environments was proposed. In the proposed model, the cloud is partitioned into several sectors and a load balancing method is used in the smaller sectors. This was inspired by the food-finding behavior of honey bees. This load balancing algorithm can consider each resource based on its capabilities and accessibility to tasks service providers, which increases the efficiency of the cloud system. Moreover, the indices of speed and cost are considered for the prioritization of virtual machines. Therefore, the proposed algorithm selects efficient resources for performing tasks based on the indices of speed and cost for the prioritization of virtual machines and the amount of load on the resources, which minimizes the time needed to service all tasks and balances system load. For future resources, the effect of fixed cost as well as the time can be evaluated, and a pricing model for user payments to cloud service providers can be organized. Moreover, cloud system modeling can be accomplished through reliable models and a hierarchical scheduler that considers the reliability of an application. This kind of load balancing can be expanded for independent tasks and the algorithm can be improved to include other factors pertaining to service quality. Moreover, the magnitude of load within the entire cloud system can be investigated, and migration can be used to distribute load between under loaded and overloaded resources.

References

[1] D. Babu, P.V. Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing, 13*(5), 2292–2303, 2013.

[2] S. Domanal, G. R. M. Reddy, Load balancing in cloud computing using modified throttled algorithm. *Cloud Computing in Emerging Markets.* Doi:10.1109/CCEM.2013.6684434, 2013.

[3] K. Li, G. Xu, G. Zhao, Y. Dong, D. Wang, Cloud task scheduling based on load balancing ant colony optimization. *Sixth Annual Chinagrid Conference* (pp. 3–9). Liaoning, 2011.

[4] K. Nishant, P. Sharma, V. Krishna, C. Gupta, Load balancing of nodes in cloud using ant colony optimization. *14th International Conference on Computer Modeling and Simulation* (pp. 3–8). Cambridge, 2012.

[5] H. Ren, Y. Lan, C. Yin, The load balancing algorithm in cloud computing environment. *2nd International Conference on Computer Science and Network Technology* (pp. 925–928). Changchun, 2012.

[6] G. Soni, M. Kalra, A novel approach for load balancing in cloud data center. *IEEE International Advance Computing Conference* (pp. 807–812). Gurgaon, 2014.

[7] H-S. Wu, C-J. Wang, J-Y. Xie, Terascaler ELB-an algorithm of prediction-based elastic load balancing resource management in cloud computing. *27th International Conference on Advanced Information Networking and Applications Workshops* (pp. 649–654). Barcelona, 2013.

[8] G. Xu, J. Pang, X. Fu, A load balancing model based on cloud partitioning for the public cloud. *Tsinghua Science & Technology, 18*(1), 34–39, 2013.

[9] K. Q. Yan, S. C. Wang, C. P. Chang, J. S. Lin, A hybrid load balancing policy underlying grid computing environment. *Computer Standards & Interfaces, 29*(2), 161–173, 2007.

[10] M. Effatparvar, M. Dehghan, A. M. Rahmani, A comprehensive survey of energy-aware routing protocols in wireless body area sensor networks. *Journal of medical systems, 40*(9), 201, 2016.

[11] M. Effatparvar, M. S. Garshasbi, A genetic algorithm for static load balancing in parallel heterogeneous systems. *Procedia-Social and Behavioral Sciences, 129*, 358-364, 2014.

[12] S. Molaiy, M. Effatparvar, Scheduling in Grid Systems using Ant Colony Algorithm. *International Journal of Computer Network and Information Security, 6*(2), 19, 2014.

[13] M. Effatparvar, M. Dehghan, A. M. Rahmani, Lifetime maximization in wireless body area sensor networks. *Biomedical Research, 28*(22), 2017.

[14] M. Effatparvar, F. Rezanezhad, S. Aghayi, Load balancing and resource allocation management with Data Envelopment Analysis method for cloud computing environment. *Recent Applications of Data Envelopment Analysis, 978*(1), 92, 2016.

[15] M. Effatparvar, S. S. Madani, Evaluation of Fault Tolerance in Cloud Computing using Colored Petri Nets. *Evaluation, 7*(7), 2016.

[16] A. Jalalat, B. Mahdavi, M. Salemi, M. Effatparvar, Comparative Study of Heuristic Algorithms and Nature-Inspired Algorithms for Scheduling in Grid Computing

[17] B. Azizpour, M. Effatparvar, M. S. Garshasbi, A New Fuzzy-based Job Scheduling Algorithm for Cluster Computing. *International Journal of Computer Applications, 77*(1), 2013.

[18] M. EffatParvar, A. Bemana, M. Dehghan, Determining a central controlling processor with fault tolerant method in distributed system. In *Information Technology, 2007. ITNG'07. Fourth International Conference on* (pp. 658-663). IEEE. 2007.

[19] M. S. Garshasbi, M. Effatparvar, High performance scheduling in parallel heterogeneous multiprocessor systems using evolutionary algorithms. *International Journal of Intelligent Systems and Applications, 5*(11), 89, 2013.

[20] J. S. Ardabili, N. Aghayi, A. Monzali, New efficiency using undesirable factors of data envelopment analysis. *Adv. Modeling & Optimization, 9*(2), 249-255, 2007.

[21] N. Aghayi, B. Maleki, Efficiency measurement of DMUs with undesirable outputs under uncertainty based on the directional distance function: Application on bank industry. *Energy, 112*, 376-387, 2016.

[22] N. Aghayi, M. Tavana, M. A. Raayatpanah, Robust efficiency measurement with common set of weights under varying degrees of conservatism and data uncertainty. *European Journal of Industrial Engineering, 10*(3), 385-405, 2016.

[23] N. Aghayi, Cost efficiency measurement with fuzzy data in DEA. *Journal of Intelligent & Fuzzy Systems, 32*(1), 409-420, 2017.

[24] P. Shankar, M. Fazelpour, J. D. Summers, Comparative study of optimization techniques in sizing mesostructures for use in NonPneumatic tires. *Journal of Computing and Information Science in Engineering, 15*(4), 041009, 2015.

[25] M. Fazelpour, P. Shankar, J. D. Summers, Developing design guidelines for meso-scaled periodic cellular material structures under shear loading. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. V02BT03A002-V02BT03A002). American Society of Mechanical Engineers. 2016.

[26] M. Yoder, Z. Satterfield, M. Fazelpour, J. D. Summers, G. Fadel, Numerical Methods for the design of meso-structures: a comparative review. In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. V02BT03A003-V02BT03A003). American Society of Mechanical Engineers, 2015.

[27] Rostamy-malkhalifeh, M., & Aghayi, N. (2011). Measuring Overall Profit Efficiency with Fuzzy Data. Journal of Mathematical Extension.

[28] Aghayi, N., & Ghelejbeigi, Z. (2016). Improving Cost Efficiency by Resource Allocation. Far East Journal of Mathematical Sciences, 99(11), 1633.

[29] Aghayi, N. (2016, January). Revenue Efficiency Measurement with Undesirable Data in Fuzzy DEA. In Intelligent Systems, Modelling and Simulation (ISMS), 2016 7th International Conference on (pp. 109-113).

[30] Salehpour, S., & Aghayi, N. (2015). The most revenue efficiency with price uncertainty. International Journal of Data Envelopment Analysis, 3(1), 575-592.