Egyptian Informatics Journal 24 (2023) 100387

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

LBCC-Hung: A load balancing protocol for cloud computing based on Hungarian method

Imane Aly Saroit *, Dina Tarek

Information Technology Department, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt

ARTICLE INFO

Article history:
Received 5 March 2023
Revised 23 May 2023
Accepted 27 May 2023
Available online 7 June 2023

Keywords:
Cloud computing
Virtual machine
Load balancing
Task scheduling
Hungarian method

ABSTRACT

Cloud Computing can be defined as enabling computing resources to whoever want remotely upon demand. This necessarily requires using virtualization. A virtual machine (VM) is an emulation of a computer that runs in virtualization software. Tasks coming from different users are passed to the virtual machines to be processed. This paper studies the load balancing problem among various VMs; to ensure that the network resources are distributed in a fair way between various clients. This paper proposes solving the load balancing problem using a combinatorial optimization approach named the Hungarian method. The proposed protocol is named LBCC-Hung (Load Balancing Protocol for Cloud Computing Based on Hungarian Method). Using simulation, the performance of LBCC-Hung is measured and compared with two well-known methods; MIN-MIN and First Come First Serve FCFS methods. The simulation results proved that LBCC-Hung overperforms the others two protocols, in terms of both the Makespan and the throughput and the virtual machine utilization deviation.

1. Introduction

Cloud computing is defined as "a model to enable on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1].

Virtualization must be used with cloud computing. A virtual machine (VM) is an emulation of a computer that runs in virtualization software. Tasks coming from different users are passed to the virtual machines to be processed.

Task scheduling and load scheduling for cloud computing are two related topics. Task scheduling means distributing the arrived tasks over the cloud resources, in simple words, over the cloud's virtual machines. Load balancing is to distribute the tasks in a fair way among the virtual machines.

Many papers addressed these two topics [2–12], but in different ways. Some of them uses the regular techniques such as; First-Come First-Serve (FCFS), Min-Min, Shortest Job First (SJF) . . . etc. While others use soft computing techniques, such as; genetic algorithm, particle swarm, slap swarm . . . etc. The first type techniques have low complexity, but they are primitive, which sometimes lead to low performance, while the techniques of the second type can improve the performance but have high complexity, i.e. high execution time, which is not suitable for real-time traffic.

This paper proposes diverting the load balancing problem into an assignment problem, then solving it using Hungarian method [13–15], which is a combinatorial optimization one with main advantage of using simple matrix operations s.a addition and subtraction to reach optimal solution in polynomial time. The processing time, considering not only the tasks needed to be processed but also the waiting ones already queued, is used as factor for choice.

This rest of this paper is divided into four sections. Section II overviews some of the works considering the same problem. Section III explains in details the proposed protocol; Load Balancing Protocol for Cloud Computing Based on Hungarian Method (LBCC-Hung). Section IV evaluates the proposed protocol. Finally, section V concludes the results obtained in the paper and suggests future works.

2. Load balancing in cloud computing

Many papers delt with the load balancing problem in cloud computing. This section overviews some of the recent ones.

In [2], M. B. Gawali et al. assigned each task a rank according to its length and run time using analytic hierarchy process (AHP), the

* Corresponding author.
E-mail addresses: i.saroit@fci-cu.edu.eg (I. Aly Saroit), d.tarek@fci-cu.edu.eg (D. Tarek).

https://doi.org/10.1016/j.eij.2023.100387

tasks are arranged into various queues. Then uses a bandwidth aware divisible scheduling (BATS) + BAR optimization to allocate the resources needed for each task. The optimization method considered both the bandwidth and load of the cloud. Each VM uses a table to determine its status. The proposed system uses divide and-conquer methodology to break up the task and distributes over virtual machines. The efficiency of the proposed system is proven when comparing its performance with that of BATS & IDEA using turnaround time and response time as metrics.

The Effective Load Balancing Algorithm with Deadline constraint (ELBAD) was proposed by K.I.Arif in [3]. It allocates the nearest deadline tasks to the highest speed virtual machines, then it balances the workload among the virtual machines. The algorithm arranges the arrived according to their deadline constraint, and the sorts available VMs in terms of their speeds. It then distributes the tasks to VMs with respect to task deadline time and VM capacity. For load balancing; VMs are classified as; underload, load and overload. The overloaded VMs transfer some of their tasks with longest deadline to underload VMs. To prove the superiority of ELBAD efficiency; it is compared with FCFS, SJF, Min-Min and EDF. It minimizes the Makespan and maximizes the resource utilization.

A distributed dynamic load balancing method based on hybrid approach named EDLBHA was proposed in [4]. Its main idea is to identify unused machines and resources earlier in shortest access time. The cloud is dynamically divided into partitions. Depending on the load, the queue length and processing time the status of each cloud partition is defined; idle or normal or overloaded. When a new task arrived, the right partition is selected. Priority may also be used. Using cloud simulator; the proposed method is compared with existing Load balancing algorithm. Simulation results proved that the proposed method outperforms the others in terms of energy utilization, waiting time, response time and turnaround time.

S. Jeyalaksshmi et al. proposed a modified round robin and modified honey bee algorithms to gain effective load balancing [5]. Virtual machines are considered as overburdened and under burdened according to their status. Operations are uninstalled from overburdened, and given priority to serve. The authors also suggest assigning weights to virtual machines depending on their computer power. For tasks with priority, they employ the Honeybee Influenced load balancing algorithm to use virtual machine weight and delegate non-preemptive tasks to underused virtual machines. They perform proactive tasks where there is no prioritization. Experimental results proved that the proposed algorithm accelerates reaction and loading time of the data center.

K.R. Remesh et al. proposed modifying the bee colony algorithm to solve the load balancing problem in cloud [6]. The tasks with lowest priority are migrated from overloaded to underloaded virtual machines. The honey bees foraging behavior is used to balance load across virtual machines. The migrated tasks are considered as honeybees, while the underloaded virtual machines are considered as the food sources. The minimization of both the Makespan and the number of migrations are the main goals. Experimental results proved that the proposed algorithm outperforms the existing bee colony algorithm. It minimizes both the Makespan and number of migrations, this leads to better QoS to the end users.

Youssef Fahim et al. [7] proposed a load balancing approach using *meta*-heuristic algorithms. The proposed approach is divided into two phases, the first one is the pre-classification of tasks into dynamic number of levels, according to the requested resources, based on the Bat-algorithm. Then using also bat-algorithm and through proposed mathematical functions, the system is divided into a number of virtual machines with either nearly equal performance or using different classes of virtual machines, such that each one of them should contain machines with similar characteristics. For the second phase, the most efficient number of virtual machines are used to execute all the tasks. For the approach evaluation, CloudSim was used to build it and calculate the processing time and the workload. But it was not compared with any other ones. Note that the authors claimed that this proposed approach is adaptable with any *meta*-heuristic rather than the bat algorithm.

In [8], A. Pradhan et al. used Particle Swarm Optimization (PSO) algorithm to solve the problem of load balancing and task scheduling in cloud computing. They proposed a protocol named Load Balancing Modified PSO (LBMPSO). They utilized a fitness function to evaluate the ideal arrangement of every particle. The fitness value of each particle computes the execution time of each VM and returned the most elevated execution time. The proposed protocol was implemented using CloudSim simulator and compared with other existing techniques that used PSO to solve the same problem, but with different fitness function. Simulation results showed the superiority of the LBMPSO protocol in terms of Makespan and the resource utilization.

Self-adaptive salp swarm optimization was used by A. Rath et al. in [9] to reach load balancing in cloud computing. As the salp swarm optimization is a continuous phenomenon, in contraire of the task assignment to VMs, so first a sigmoid transfer function has been integrated to the salp swarm optimization to produce a binary self-Adaptive salp swarm optimization algorithm. Then a task fitness function was proposed that merge the Makespan, the response time, the degree of imbalance and VM utilization, all with aims of minimization except for the VM utilization that needs to be maximized. The efficiency of the proposed algorithm was proposed by comparing its performance with recent metaheuristic approaches used to solve the same problem. Results proved that it outperformed the other protocols in terms of Makespan, response time, degree of imbalance and VM utilization.

Many papers illustrate surveys on task scheduling for cloud computing [10–12].

All the above techniques use machine learning techniques, that are usually used in order to find patterns and predicting next states. While mathematical optimization techniques are usually used in scheduling, finding the optimal placement of facilities or minimizing/maximizing costs for a problem [13].

3. Proposed load balancing protocol for cloud computing using Hungarian method

This section illustrates the proposed protocol for virtual machines' load balancing in cloud computing. It is divided into three subsections. The first one formulates the problem, while the second one gives a brief overview about the Hungarian method used to solve it. And the last one explains how the Hungarian method is used to solve the load balancing problem for cloud computing.

Table 1 illustrates the various symbol used in the rest of the paper.

3.1. Problem formulation

The problem can be formulated as follows; having a number of independent tasks, each of a specific length, and a number of virtual machines, each can execute up to a number instructions per second. The goal is to distribute the tasks over the virtual machines, such that: 1) each task is executed on only one virtual machine. 2) Each virtual machine executes only one task t at a time. 3) Considering the length of the tasks, they must be distributed in a fair way over the virtual machine, taking into consideration the whole network performance.

As the allocation problem is distributing the available resources among competing alternatives in order to minimize the total cost

Table 1
List of abbreviations.

Symbol	What is stands for
t	Task
t_L	length of task t (in millions of instructions)
T	Total number of tasks
v	Virtual machine
MIPS	Millions of instructions per second to be executed in a task
v_s	Number of instructions in MIPS the VM v can execute per second.
U_V	Utilization of a virtual machine V
Busy Time $_v$	the time VM V is busy
V	Total number of virtual machines
a[t][v]	TxV allocation matrix at,v = 1 if the task t is allocated to v
PT[T][V]	TxV processing time matrix PTt,v represents the processing time VM v needs to process task t
L[T][V]	TxV load matrix Lt,v represents the load for the tasks already queued for the virtual machine v in addition to that will be added if task t is assigned to virtual machine v.
VMUD	All VMs' utilization deviation
$\bar{U}$	All VMs' utilization mean

or maximize the total income [14–16], so the problem addressed in the paper can be transformed into an allocation problem as follows:

At each time period P -as shown in Fig. 1- having:

1. T tasks to be served, each task has a specific length, defined as the number of instructions to be executed.
2. Virtual machines to be used, each virtual machine has
   - o A specific number of instructions that it can execute per second.
   - o A queue that includes the tasks waiting to be served.
3. Various weights, represents the value of the metric used as factor for choice. Wi,j is the value of the metric if task i is allocated to virtual machine j.

Define the following matrices:

4. a is a TxV allocation matrix.

$$a_{tv} = \begin{cases} 1 & \textit{task t is allocated to VM v} \\ 0 & \textit{otherwise} \end{cases} \quad (1)$$

- o Each task t is served by one virtual machine.

$$\sum_{j=1}^{v} a_{tj} = 1 \qquad \forall t = 1,2,3\cdots T \quad (2)$$

- o Each virtual machine v is assigned one task t at a time, but as all tasks arrived in the period must be assigned to a virtual machine. And as the number of tasks may be greater than the number of virtual machines, so each virtual machine can be assigned more than one task. Those tasks are queued. Each virtual machine serves the queued tasks one by one. Of course, if the number of tasks is less than the number of virtual machines, one or more virtual machine may have not assigned any tasks.

$$\sum_{i=1}^{t} a_{iv} \geq 0 \qquad \forall v = 1,2,3\cdots V \quad (3)$$

- o All tasks must be served.

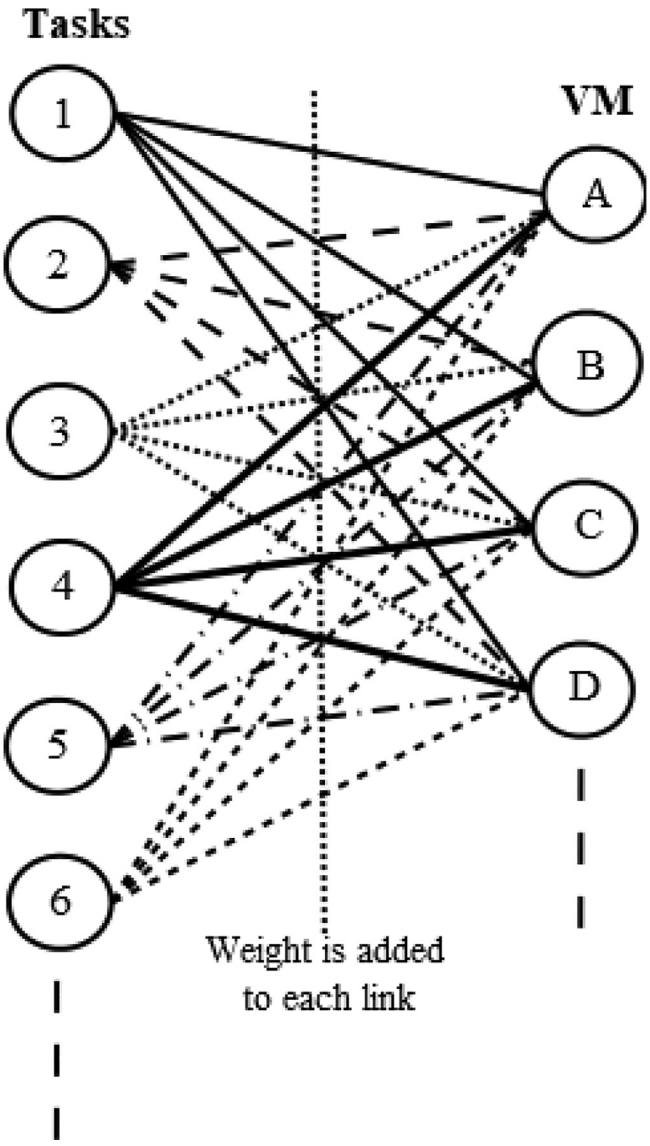


Fig. 1. Task-VM Allocation Problem.

$$\sum_{i=1}^{t} \sum_{j=1}^{v} a_{ij} = t \quad (4)$$

5. **PT** is an TxV processing time matrix.
   - o PTt,v represents the processing time VM V needs to process task t.

$$PT_{t,v} = \frac{L_{t,v}}{v_s} \quad (5)$$

where:vs is the number of instructions the VM v can execute per second.Lt,v represents the load (number of instructions to be executed) for the tasks already queued for the virtual machine v in addition to that will be added if task t is assigned to virtual machine v.

$$L_{t,v} = \sum_{i=1}^{x} v_{qi_L} + t_L \quad (6)$$

Where: x is the number of tasks queued in vq
v_{qi_L} is the length of task i queued in vq
t_L is the length of task t.

The main goal of the proposed protocol is gaining load balancing, i.e distribute the tasks T over the virtual machines V in a fair way, while keeping a good network performance.

As the tasks' lengths are not equal, so using only the number of tasks assigned to each VM, cannot be used to measure the load balancing, instead the tasks' length must be taken into consideration. The best way to measure the load balancing, is by using the virtual machines' utilization standard deviation. Note that the standard deviation is the most commonly parameter used for measuring variability. So, a low value for the virtual machines' utilization deviation, means that most of the virtual machines are almost occupied in the same way.

In order to calculate the virtual machines' standard utilization deviation (called only deviation for simplicity); first, the utilization of a virtual machine is calculated using not only the number of tasks served by the virtual machine, but also their lengths. It is calculated as follows:

$$\boldsymbol{U_v = \frac{Busy\_Time_v}{Makespan}} \tag{7}$$

where: Uv is the utilization of virtual machine v.

Busy_Time v is the time VM V is busy.

Makespan is the whole time taken to complete execution of all the tasks (see equation (11).

$$\boldsymbol{Busy\_time_v = \sum_{i=1}^{n} \frac{t_{Li}}{v_s}} \tag{8}$$

where: n is the number of tasks t served by v.

$\boldsymbol{t_{Li}}$ is the length of task i.

vs is the number of instructions the virtual machine v can execute per second.

Then, the virtual machines' utilization deviation is calculated as follows:

$$\boldsymbol{VMUD = \sqrt{\frac{\sum_{i=1}^{V} \left(U_i - \bar{U}\right)^2}{V}}} \tag{9}$$

where: VMUD is the VMs' utilization deviation

U_i is the utilization of virtual machine v.

$\boldsymbol{\bar{U}}$ is the VMs' utilization mean.

V is the total number of VMs.

$$\bar{U} = \frac{\sum_{i=1}^{K} U_i}{V} \tag{10}$$

Of course, the utilization of a VM, and the VM utilizations' deviation are not calculated at each period. So, it is no practical to use either of them as a metric for decision making at each period. But as both the utilization (if multiplied by the Makespan) and the processing time are calculated by dividing the load over the number of instructions to be executed per second, but in a different time. So, the processing time matrix can be used as the cost matrix.

3.2. Hungarian method

The Hungarian method is a computational optimization technique, it solves the assignment problem in polynomial time. It is used to solve weighted matching problems to find a perfect matching between resources and competing alternatives so that minimum cost / maximum profit is obtained. The cost/profit is calculated using the weight defined between the resources and competing alternatives.

The Hungarian Method is based on the following principle: If a constant is added/subtracted to every element of a row/column of a square matrix, then the optimum solution of the resulting assignment problem is the same as the original problem and vice versa. So, the original square matrix can be reduced to another matrix (of the same size) by adding/subtracting constants to the elements of rows/ columns where the total cost or the total completion time of an assignment is zero. Since the optimum solution remains unchanged after this reduction, this assignment is also the optimum solution of the original problem [14–16]. The worst-case computational complexity of the Hungarian method is $O(n^3)$, where n is the order of the square matrix [14].

3.3. Proposed load balancing protocol for cloud computing using Hungarian method (LBCC-Hung)

Using the Hungarian method to solve the load balancing problem is drawn in Figs. 2 & 3.

Having: At the first time period.

- T tasks to be assigned to V virtual machines.
- Cost matrix as the processing time matrix, calculated using equation (5).

The proposed LBCC-Hung works as follows:

1. Square the cost matrix to be XxX
   - Define X as the largest of either the number of tasks (to be executed) or the number virtual machines.
   - Enlarge the cost matrix to be XxX, by adding dummy row (s)/column(s) with zero elements.
2. Subtract row minima
   - For each row in the matrix:
   - Find the lowest value in the row.
   - Subtract this lowest value from each element in the row.
3. Subtract column minima
   - For each column in the matrix:
   - Find the lowest value in the column.
   - Subtract this lowest value from each element in the column.
4. Cover zeros with minim number of lines
   - Cover all the zeros in the matrix with the minimum number of lines.
   - If this number of lines = X, go to step 6.
5. Create New zeros
   - Find the value of the smallest element in the matrix not covered by any line.

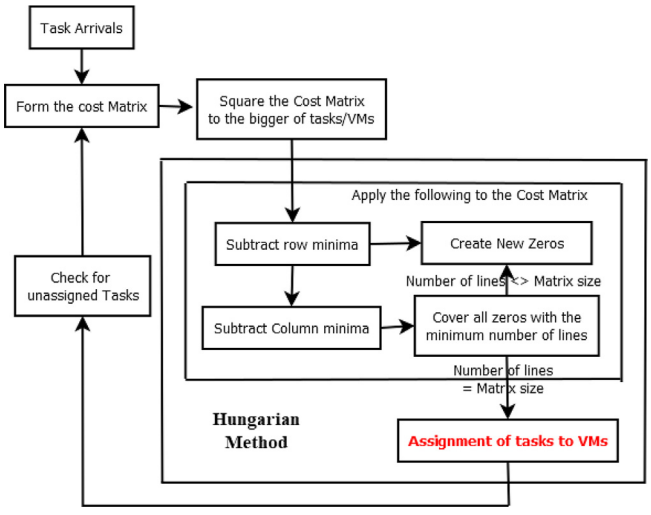


Fig. 2. Functional Diagram for the LBCC-Hung Protocol.

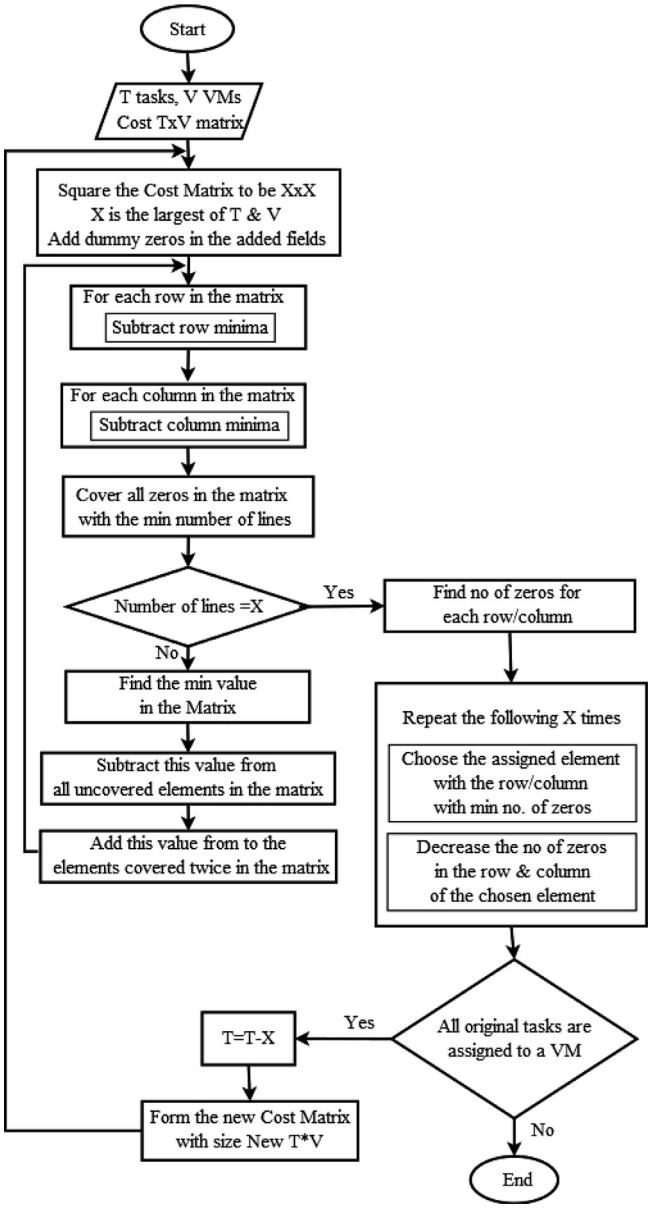


Fig. 3. Flow Chart for the LBCC-Hung Protocol.

- Subtract this value from all uncovered elements in the matrix.
- Add this value to all elements covered twice in the matrix.
- Go to step 2.

6. Assignment
   - Find the number of zeros in each row and in each column.
   - Repeat the following X times (for the X tasks' assignment process):
     - Choose the assigned element within the row or column with minimum number of zeros.
     - This element is located in row R and column C. This means that the task R is assigned to VM C.
     - Decrease the number of zeros of the row and column of the chosen element.

7. Check for the unassigned tasks
   a. Find the number of tasks assigned to virtual machines.
   b. If all tasks are assigned (=T), Stop.
      - Else:* Decrease T by X.
      - *Form the new cost matrix (new X = T-X).
      - *Go to step 1.

4. Evaluation of the proposed load balancing protocol

In this section, the proposed load balancing technique (LBCC-Hung) is evaluated. Using simulation, the performance of LBCC-Hung is measured and compared with two well-known used methods; MIN-MIN [17] and First Come First Serve FCFS methods. This section is composed of three subsections. The first one states the performance metrics used for the evaluation. The second one illustrates the simulation parameters. The third part presents and analyses the obtained simulation results.

4.1. Performance evaluation metrics

The performance of the load balancing methods may be measured with many parameters. The most used ones are the Makespan, the throughput and the virtual machine utilization deviation. They are defined and calculated in this subsection.

A. **Makespan**

The Makespan is defined as the elapsed time between the start and finish of a sequence of tasks in a set of virtual machines, in another way it is the maximum completion time of all tasks. It is preferred to obtain lower Makespan.

$$Makespan = \max\left(\forall_t Completiontime(t)\right) \tag{11}$$

where;

t is the number of tasks.

Completion time (t) = Waiting time (t) before its allocation to a VM + Queuing time (t) in the chosen VM before being served + Execution time (t) on the chosen VM.

B. **Throughput**

Throughput is the number of instructions completed in a second. In order to obtain good performance, the throughput must be maximized.

$$Throughput = \frac{\sum_{j=1}^{t} Length(j)}{Makespan} \tag{12}$$

where;

t is the total number of tasks to be served.

Length(i) is the length of task i (as number of instructions needed to be executed).

C. **Virtual Machine Utilization Deviation**

As explained in section 3, the virtual machine utilization deviation (VMUD) is defined as the standard deviation of the virtual machine utilization. To obtain good load balancing over various virtual machines, the main goal was to obtain minimum deviation between their utilization. It is calculated using equation (9).

4.2. Simulation parameters

Table 2 represents the values of the parameters used in the simulation. Note that the value of some of these parameters will be changed to test the effect of their change on the measured performance metrics. To get accurate results, the tasks' length (generated randomly) is added in a separate file to be used as similar inputs for the three techniques.

Table 2
Simulation parameters' value.

Parameter	Used Value(s)
Task Rate	30
Task Length (in instructions)	generated randomly from 1000 to 1,000,000
Number of Virtual Machines (VM)	4
VM speed (MPIS)	10
Period Length (in msec)	250
Simulation time	30 sec

4.3. Simulation results

To evaluate the proposed method (LBCC-Hung), the three-performance metrics calculated using equations (9), (11) & (12) are measured using a built simulation program, in addition to two well-known methods; MIN-MIN and FCFS. The FCFS is a scheduling protocol, that processes the tasks by their order of arrival, while the min-min scheduling protocol, that minimizes the total completion time of the tasks.

For the simulations, some parameters remain constant, while others are changed to study the impact of their change on the performance. Note that each point in each of the graphs in this subsection represents the average of the results of five runs of the simulation program.

A. Changing Tasks' Rate

The graphs in Fig. 4 show the value of the three evaluated metrics measured when applying the three methods, the proposed method (LBCC-Hung) and the two old ones Min-Min and FCFS, using various task rates, that varied from 20 to 60 tasks/sec.

As clear from the above graphs, as the tasks' rate increases, the load increases over the cloud, so it takes more time to serve them, so the Makespan increases for the three techniques, this also holds for the throughput. But for the VM's utilization deviation, it increases with the increase of the data rate for the MIN-MIN and the proposed LBCC-Hung, while decreases for the FCFS.

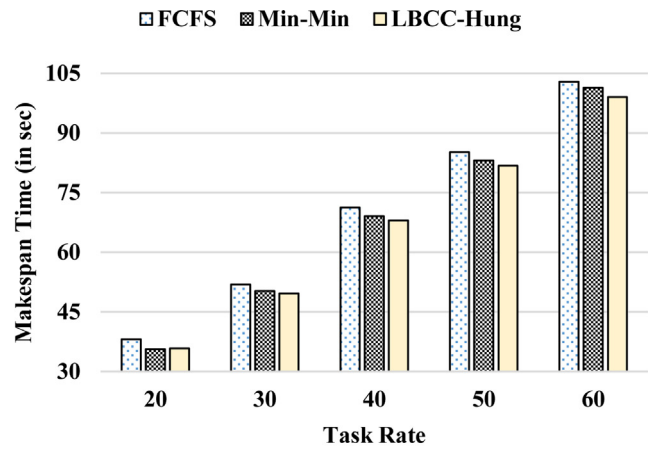
It is also clear that using LBCC-Hung always lead to better results than using the other two methods, lower Makespan, higher throughput and lower virtual machines' utilization deviation. Percentage of improvement varies from 1% to 7% for both the Makespan and the throughput, while it can reach 91% for the VMs' utilization deviation.

B. Changing Number of Virtual Machines

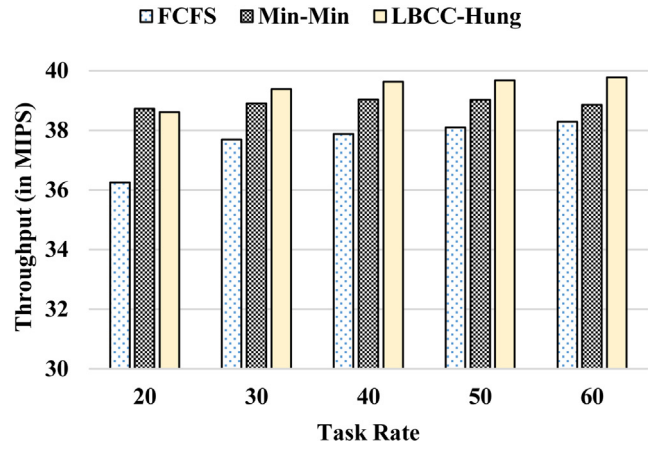
The graphs in Fig. 5 outline the three evaluated metrics measured when applying the three methods, the proposed LBCC-Hung technique and the two old ones Min-Min and FCFS, using various number of virtual machines, that varied for 2 to 6.

As clear from the graphs in Fig. 5, as the number of virtual machines increases, the cloud will be able to serve more tasks in a shorter way, so it takes less time to serve the existing tasks, so less Makespan and more throughput, this holds for the three techniques, but with different way.

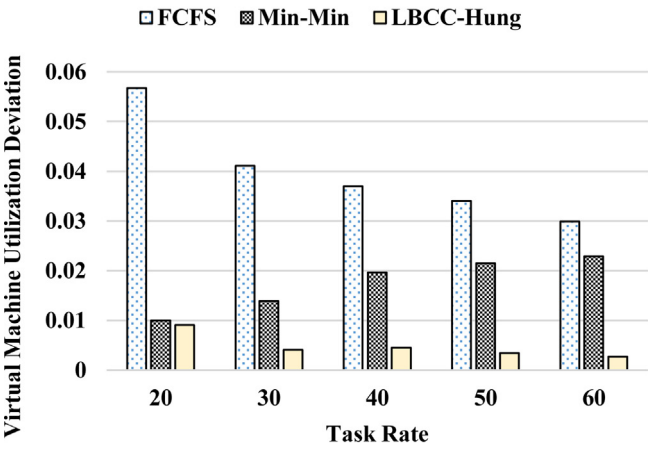
It is also clear that using LBCC-Hung always lead to better results than using the other two methods, lower Makespan, higher throughput and lower load deviation over virtual machines. Percentage of improvement varies from 1% to 8% for both the Makespan and the throughput, while it can reach 90% for the VMs' utilization deviation.



(a) Makespan



(b) Throughput

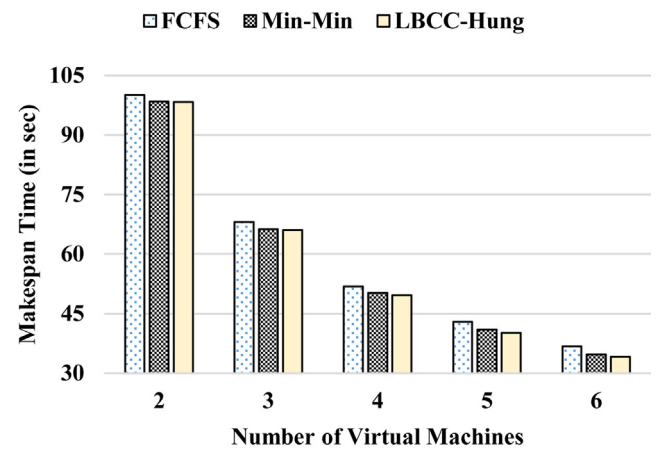


(c) Virtual Machines' Utilization Deviation

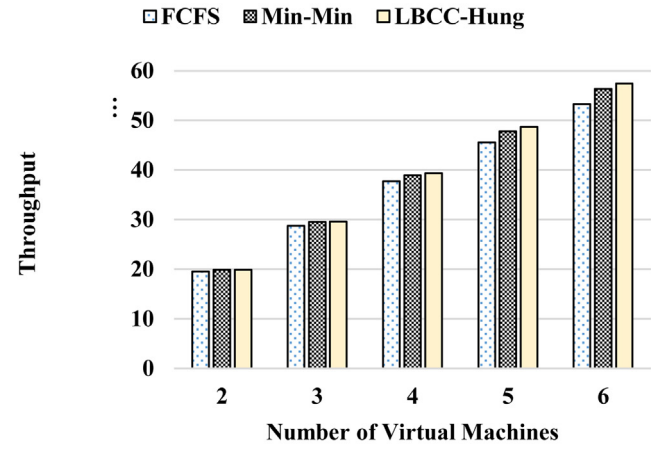
Fig. 4. Performance Evaluations Using Various Task Rates.

C. Changing the Virtual Machines' Speed

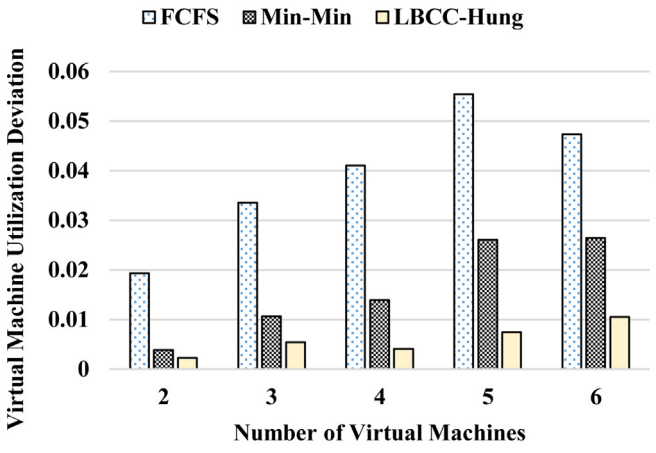
The graphs in Fig. 6 illustrate the three evaluated metrics measured when applying the three methods, the proposed one (LBCC-



(a) Makespan

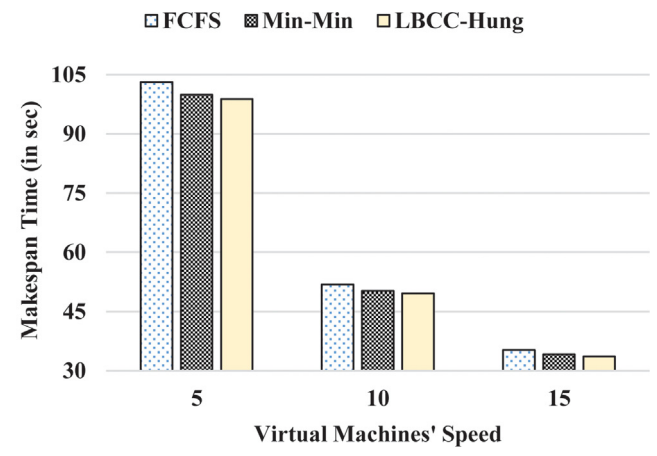


(b) Throughput

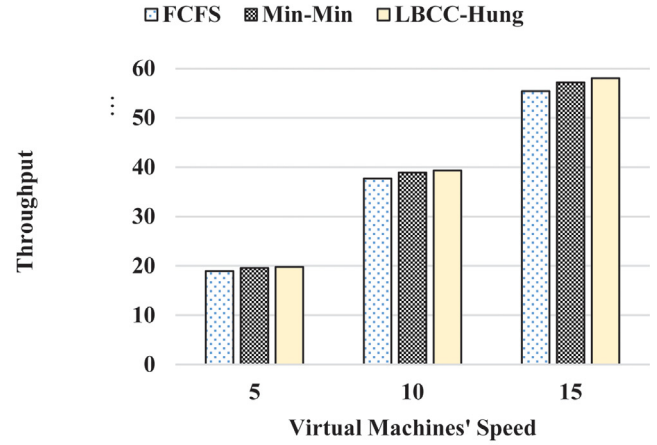


(c) Virtual Machines' Utilization Deviation

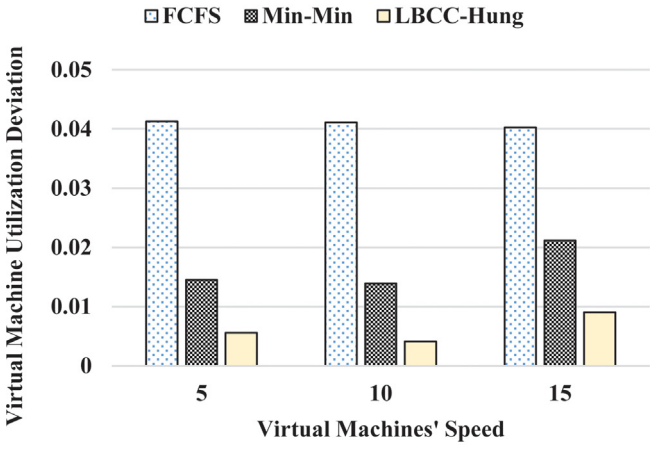
Fig. 5. Performance Evaluations Using Various Number of Virtual Machines.



(a) Makespan



(b) Throughput



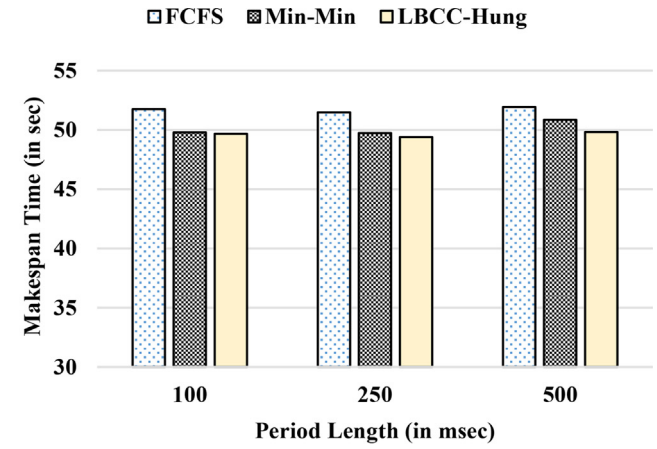
(c) Virtual Machines' Utilization Deviation

Fig. 6. Performance Evaluations Using Various Virtual Machines' Speed.

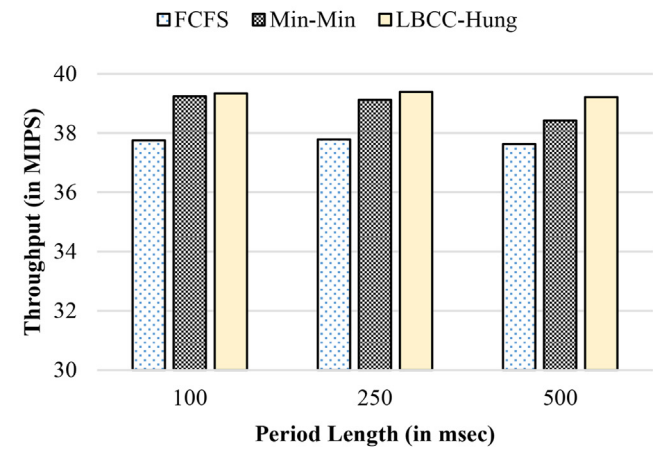
Hung) and the two old ones Min-Min and FCFS, using various VMs' speed (number of executed instructions by the VM per second) in MIPS. It varies for 5 to 15 MIPS.

As clear from the graphs in Fig. 6, as the number of instructions each virtual machine can process increases, so each VM will be able to serve more tasks in a shorter time, i.e. all the cloud will be able to serve more tasks in a shorter way, so it takes less time to serve the existing tasks, so less Makespan and more throughput, this holds for the three techniques, but with different rate.

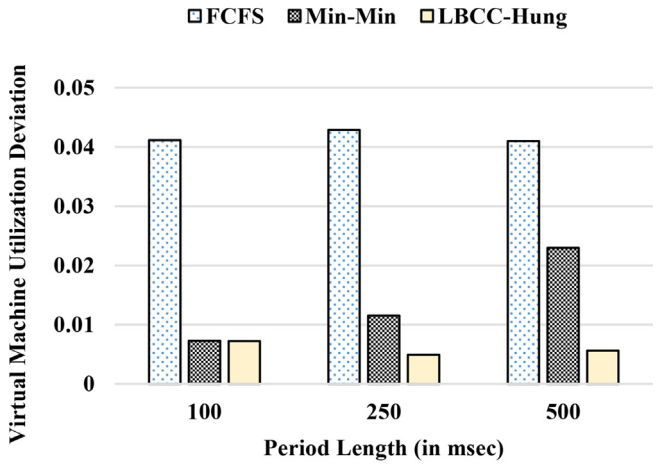
It is clear that using LBCC-Hung always lead to better results than using the other two methods, lower Makespan, higher throughput and lower utilization deviation over virtual machines. The improvement varies from 1% to 2% for both the Makespan and the throughput while it can reach 90% for the VMs' utilization deviation.



(a) Makespan



(b) Throughput



(c) Virtual Machines' Utilization Deviation

Fig. 7. Performance Evaluations Using Various Period Length.

D. Changing the Period Length

The graphs in Fig. 7 illustrate the three evaluated metrics resulted when applying the three methods, the proposed method and the two old ones Min-Min and FCFS, using various period length, that varied for 100 to 500 msec.

As clear from the graphs in Fig. 7, as the period length increases, the Makespan and the throughput have minor changes (they do not almost change), while there is a small change in the utilization deviation.

It is also clear that using LBCC-Hung always lead to better results than using the other two methods, lower Makespan, higher throughput and lower load deviation over virtual machines. The improvement reaches 2% for both the Makespan and the throughput, and 75% for the VMs' utilization deviation.

E. Analysis of the results

The graphs in Figs. 4 to 7 shows the three evaluated metrics; the Makespan, the throughput and the VM utilization deviation resulted when applying the three methods, the proposed LBCC-Hung protocol and the two old ones Min-Min and FCFS, under various condition. All the results proved the preference of the proposed protocol over the other two protocols. The reason for that is; the Hungarian method is an easy method of finding the feasible solution in a short time, so it can easily find the solution in order to assign a group of tasks to the group of available virtual machines in an optimal (maximum/minimum) balanced way. The min-min method; that also assigns the tasks to virtual machine; is not optimal with respect to fairness as it always prefers the smaller tasks [18], this leads to results worse than that of the Hungarian. For the FCFS method, although it is simple, but it suffers from convey effect which leads to long average delay [19], this leads to the worst results compared to the other two ones.

Note that the improvement is low for the Makespan and throughput but very high for the VM utilization deviation. The improvement increases in case of overloaded status, such as high the task rate, low number of virtual machines, low virtual machine speed and long period.

5. Conclusion and future work

This paper addresses the problem of load balancing for the VM in cloud computing, meaning distributing the arriving tasks over the virtual machines in fair way, taking into account the cloud whole performance. The problem is turned it into an assignment problem and solved it using the Hungarian method, that is a combinatorial optimization that uses simple matrix operations, so has a low complexity = O(n3). The proposed protocol is named LBCC-Hung (Load Balancing Protocol for Cloud Computing Based on Hungarian Method). The paper explained the proposed protocol in details. For the evaluation, the proposed LBCC-Hung protocol was simulated and built together with another two well-known used protocols; MIN-MIN and First Come First Serve (FCFS) protocols. The number of tasks and the length of each task were generated randomly. Using various the tasks' rate, number of virtual machines, virtual machine speed and period length, the performance of the three protocols were measured. The results proved that under various condition; the proposed LBCC-Hung outperformed the other two protocols in terms of Makespan, throughput and VM utilization deviation. The improvement is minor in both of the Makespan and throughput (up to 8%), but it is major for the VM utilization deviation (up to 90%).

For future work, we plan to update the protocol to deal with dependent tasks. Also, priority among tasks will be considered.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Sikka and M. Ojha, "An overview of cloud computing," Int J Innov Res Comput Sci Technol, vol. 2, no. 3, pp. 135–138, 2021, 10.55524/ijircst.2021.9.6.31.

[2] Gawali MB, Shinde SK. Task scheduling and resource allocation in cloud computing using a heuristic approach. J Cloud Comp 2018;7(1).

[3] Ibraheem Arif K. An effective load balancing algorithm based on deadline constraint under cloud computing. IOP Conf Ser: Mater Sci Eng 2020;928 (3):032070.

[4] Rai S, Sagar N, Sahu R. An efficient distributed dynamic load balancing method based on hybrid approach in cloud computing. Int J Comput Appl 2017;169 (9):16–21.

[5] Jeyalaksshmi S, Anita Smiles J, Akila D, Mukherjee D, Obaid AJ. Energy-Efficient Load Balancing Technique to optimize Average response time and Data Center Processing Time in Cloud Computing Environment. J Phys Conf Ser 2021;1963 (1):1–6. doi: https://doi.org/10.1088/1742-6596/1963/1/012145.

[6] K. R. R. Babu and Philip Samuel, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud," in Innovations in Bio-Inspired Computing and Applications (IBICA 2015), India, 2015, pp. 67–78. 10.1007/978-3-319-28031-8.

[7] Youssef Fahim H, Rahhali M, Hanine E-H. El-Houssine Labriji, and Ahmed Eddaoui, "Load balancing in cloud computing using Meta-Heuristic Algorithm". J Inf Process Syst 2018;14(3):569–89. doi: https://doi.org/10.3745/JIPS.01.0028.

[8] Pradhan A, Bisoy SK. A novel load balancing technique for cloud computing platform based on PSO. J King Saud Univ Comput Inform Sci 2022;34 (7):3988–95. doi: https://doi.org/10.1016/j.jksuci.2020.10.016.

[9] B. R. Parida, A. K. Rath, and H. Mohapatra, "Binary Self-Adaptive Salp Swarm Optimization-Based Dynamic Load Balancing in Cloud Computing," International Journal of Information Technology and Web Engineering, vol. 17, no. 1, pp. 1–25, 2022, 10.4018/ijitwe.295964.

[10] Mesbahi M, Masoud Rahmani A. "Load Balancing in Cloud Computing: A State of the Art Survey", International Journal of Modern Education and Computer. Science 2016;8(3):64–78. doi: https://doi.org/10.5815/ijmecs.2016.03.08.

[11] A. Pradhan, S. K. Bisoy, and P. K. Mallick, "Load Balancing in Cloud Computing: Survey," in Innovation in Electrical Power Engineering, Communication, and Computing Technology (IEPCCT 2019), 2019, pp. 99–111.

[12] Shafiq DA, Jhanjhi NZ, Abdullah A. Load balancing techniques in cloud computing environment: A review. J King Saud Univ Comput Inform Sci 2022;34(7):3910–33.

[13] Hennie de Harder, "Five ways to combine Mathematical Optimization and Machine Learning," 2022. https://towardsdatascience.com/four-ways-to-combine-mathematical-optimization-and-machine-learning-8cb874276254 (accessed May 24, 2023).

[14] Mills-tettey GA, Stentz A, Dias MB. The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs. Naval Research Logistics Quarterly 2007;no. July:83–7.

[15] Sultana F, Nizam M. An Alternative Proposed Method for Solution of Assignment Problem. Int J Sci Basic Appl Res (IJSBAR) 2020;52(1):40–50.

[16] "Transportation, Assignment, and Transshipment Problems." [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.producao.ufrgs.br/arquivos/disciplinas/382_winston_cap_7_transportation.pdf.

[17] Vignesh Joshi, "Load Balancing Algorithms in Cloud Computing Environment," International Journal of Research in Engineering and Innovation, vol. 3, no. 6, pp. 550–532, 2019, 10.26483/ijarcs.v9i2.5837.

[18] Kaur D, Singh S. An Efficient Job Scheduling Algorithm using Min-Min and Ant Colony Concept for Grid Computing. International Journal of Engineering Computer Science 2014;03(07):6943–9.

[19] Bharathi S, Mp C, Sn D. Comprehensive Analysis OF CPU Scheduling Algorithms. Int Res J Moderniz Eng Technol Sci. 2022;4(9):180–5.