Published in AJSE, Vol:22, Issue: 2
Received on 31th August 2022
Revised on 20th August 2023
Accepted on 22nd August 2023

AIUB JOURNAL OF SCIENCE AND ENGINEERING
ISSN: 1608 – 3679 (print) 2520 – 4890 (Online)

# A Centralized Multi-Criteria Method for Scheduling Tasks in a Cloud Computing Environment

Ehsan Shojaeian, Mehran Mohsenzadeh, and Mohammad Mehdi Sahrapour

***Abstract*— Task scheduling determines the order of mapping tasks to virtual machines to meet objectives. In this paper, a batch mode heuristic method that is centralized, dynamic, and multi-objective has been presented for scheduling independent tasks with a deadline and belonging to several user levels, using the cloud elasticity in the public cloud environment. In this method, it has been intended to improve the objectives of makespan, deadline violation, total execution cost, and load balancing by considering the tasks' prioritization based on the criteria of user level, deadline, task length, and selection of heterogeneous virtual machines according to processing power, workload and usage cost. The proposed method was simulated using the CloudSim tool. Besides, the method's ability to achieve the mentioned goals has been evaluated in comparison with similar methods. The evaluation results, established on standard test data, show that the proposed method has a good performance in improving its objectives.**

***Index Terms*— Cloud computing, Scheduling the tasks with deadline, Multi-criteria prioritization, Elasticity.**

## I. Introduction

CLOUD computing is a model for providing easy access to a set of changeable and configurable computing resources (networks, servers, storage space, applications, and services) based on user demand over the network. The access also can be provided or released quickly with the lowest resource management requirements and the service provider's direct intervention [1].

Scheduling tasks in cloud computing means mapping tasks to virtual machines in a way that the service requirements' quality requested by cloud customers are met mainly within the agreed service level [2, 3].

Cloud task scheduling is a challenging and macro issue following the main objectives of improving execution and service quality, reducing execution cost and response time while maintaining task's performance and integrity. An efficient scheduling algorithm can be created by considering existing methods and adding other criteria such as user level to it [4]. Furthermore, a scheduling algorithm must take into account the interests of the two parties involved in the interaction, which are the service provider and the service consumer [5-8].

Ehsan Shojaeian is with the Computer Engineering Department, Science and Research Branch, Islamic Azad University, Tehran, Iran (e-mail: ehsan.shojaeian@srbiau.ac.ir).
Mehran Mohsenzadeh is with the Computer Engineering Department, Science and Research Branch, Islamic Azad University, Tehran, Iran (e-mail: mohsenzadeh@srbiau.ac.ir).
Mohammad Mehdi Sahrapour is with the Business School, Monash University, 900 Dandenong Rd, Caulfield East VIC, Australia (e-mail: sahrapour@gmail.com).

Several users with different priority levels may be using the cloud service at the same time. Therefore, in addition to the common criteria such as length of execution time and deadline, users' priority levels should also be considered as one of the criteria in scheduling tasks. The criteria for prioritization in the proposed method are the users' priority levels and the tasks' length and deadline. Besides, the selection of virtual machine established on the criteria of processing power, workload and usage cost. Hence, it is possible to delete or add new criteria as needed. Depending on the situation and in accordance with the desired objective, by changing the weight of the criteria, the desired result can be achieved. This factor distinguishes the proposed method from other methods in addition to considering different effective criteria.

The proposed method is a dynamic multi-criteria method that can be used in different scenarios by providing the possibility of adjusting the effectiveness of these criteria. Dynamic prioritization has been also used to prevent tasks starvation and to consider their deadline during scheduling.

## II. Literature Review

Various task scheduling methods have been presented based on the criteria involved in scheduling and the objectives that the scheduler seeks to achieve. For example, energy consumption, makespan, load balancing, and resource utilization are the main objectives that various scheduling algorithms try to improve [9-14].

In the max-min algorithm, a task with the longest execution time is selected and assigned to the resource that provides the shortest completion time for execution. This continues until all tasks are scheduled. Although the min-min algorithm functions almost similarly to the max-min algorithm, a task with the shortest execution time is selected each time in the min-min algorithm [15-17].

When the number of short tasks in the tasks set is greater than the number of longer tasks, the min-min algorithm uses fewer resources and does not allow simultaneous execution of tasks. As a result, the makespan will increase. Although the max-min algorithm works better than the min-min algorithm in this case, if the number of longer tasks is higher, it will increase the makespan [18]. The min-min algorithm works in favor of short tasks and improves the system throughput. Also, starvation occurs for the shorter tasks in the max-min algorithm [19].

Chaudhary [20] presented the Deadline and Suffrage Aware Algorithm. This algorithm is similar to the min-min and max-min algorithms of a single-criterion method. The tasks' deadline, which is considered in the scheduling instead of the execution time, spots the difference here. In this algorithm, if the tasks are ranked in an ascending order based on deadline, a task with the smallest deadline is mapped to a virtual machine providing the least completion time for it. This method has a better performance compared to the max-min and min-min methods in all the considered goals.

Wu et al [21] provided a task scheduling algorithm based on QoS-driven in cloud computing. This algorithm (TS-QoS) prioritizes tasks based on the user privilege, urgency, workload (length), and latency time. In the scheduling phase, each task is assigned to the virtual machine which provides the lowest completion time. This method considers several criteria in prioritizing tasks, and the possibility of influencing each criterion in determining priority can be adjusted; however, it is more complex than single-criterion methods. In addition, it often has a weaker performance than the min-min method in terms of the makespan of tasks.

Most scheduling algorithms, including those mentioned above, have been developed based on basic algorithms such as Min-Min (which was previously used in computing environments such as the Grid) and considering more criteria for prioritizing tasks. Moreover, in most of these methods, the virtual machine is selected based on the criterion of completion time for the task with the highest priority. In the proposed method, in addition to prioritizing based on multiple criteria, we use the elasticity of the cloud environment and increase the flexibility of the scheduler by allowing the virtual machine to be selected based on several different criteria so that it is possible to remove or add different criteria in the selection of virtual machines according to the need.

## III. PROPOSED METHOD

The proposed method is a centralized, dynamic and multi-objective method that can be used in different scenarios by providing the possibility of adjusting the effectiveness of these criteria. Dynamic prioritization has been also used to prevent tasks starvation and to consider their deadline during scheduling.

In this method, in addition to the independence of tasks, it is assumed that tasks with a certain length and deadline are sent to the cloud from the users with different priority levels. these users are classified based on items such as payments, loyalty, etc, and are not fixed. he/she can increase or decrease the priority level of executing the tasks by paying higher or lower cost. Also, the scheduling platform is public cloud, and all tasks are soft real-time type.

The model shown in Fig. 1 is used to illustrate the proposed method. This model has been proposed and developed using the concepts of task scheduling in distributed systems as well as the concepts and solutions presented in [22-24]. Each cloud system consists of a number of data centers, and each data center contains several physical hosts. There are also a number of virtual machines on each of these hosts. Specifications and configurations of different data centers are registered in a service called cloud information service. Users from different levels interact with different applications, in which a series of tasks are created and recorded in the cloud scheduler.

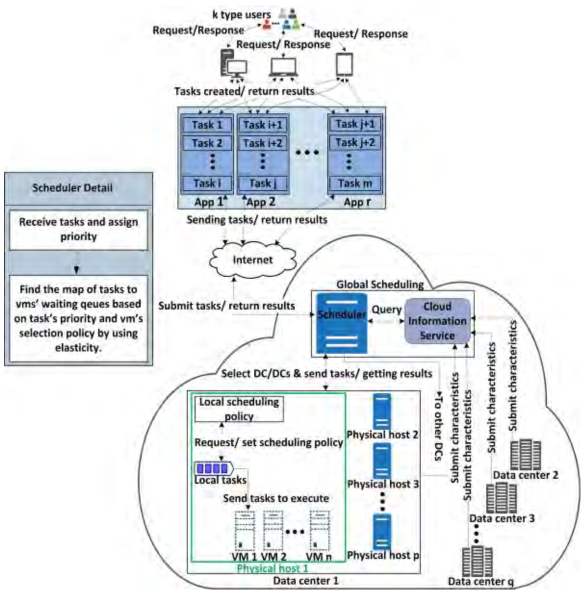


Fig. 1. proposed method model

Since scheduling operations are performed after the accumulation of a number of tasks in the scheduler, based on a set of rules and in order to improve several objectives, the CM$^3$ method falls into the category of batch mode heuristic scheduling methods and includes the following three main steps:

1) Prioritizing tasks
2) Selection of heterogeneous virtual machines
3) Mapping tasks to virtual machines using elasticity feature

### A. Definitions

Some basic concepts and technical terms of the CM$^3$ method, inspired by the definitions mentioned in [25-27] and with changes and developments have been redefined or newly defined below according to the objectives of our research.

Definition 1: The set of tasks $T = \{t_1, t_2, \dots, t_m\}$, where $t_i$ is independent of other tasks and each task is described by a quadrant as follows:

$$t_i = < Id_i, Length_i, UserLevel_i, Deadline_i > \tag{1}$$

The above characteristics are the unique identification, length, user level of the sender user, and the deadline of the task, respectively.

Definition 2: A set of heterogeneous virtual machines $VM = \{vm_1, vm_2, \ldots, vm_n\}$, where each virtual machine is described with a senary as follows:

$$vm_j = < Id_j, PrPow_j, Ram_j, Storage_j, Bw_j, Cost_j > \tag{2}$$

The above symbols indicate the unique identification, processing power, main memory, storage space, communication bandwidth and the usage cost of the virtual machine, respectively.

Definition 3: A set of physical machines $\mathrm{PM} = \{\mathrm{pm}_1, \mathrm{pm}_2, \ldots, \mathrm{pm}_\mathrm{p}\}$ where each physical machine can host one or more virtual machines.

Definition 4: Assignment function $f: T \times VM \to \{0, 1\}$ which is defined as follows:

$$f(t_i, vm_j) = \begin{cases} 1 & if\ t_i\ is\ assigned\ to\ vm_j \\ 0 & otherwise \end{cases} \tag{3}$$

The relationship $\sum_{j=1}^{n} f(t_i, vm_j) = 1$ is established for each $t_i$.

Definition 5: Task execution time of $t_i$ on the virtual machine $vm_j$ is determined as follows:

$$ExeTime_{ij} = \frac{Length_i}{PrPow_j} \tag{4}$$

Definition 6: The execution cost of task $\mathrm{t_i}$ on the virtual machine $\mathrm{vm_j}$ is specified as follows:

$$ExeCost_{ij} = ExeTime_{ij} \times Cost_j \tag{5}$$

Definition 7: The time it takes for a virtual machine to complete the mapped tasks is considered as the completion time for the virtual machine:

$$CompTime_j = \sum_{i=1}^{m} f(t_i, vm_j) \cdot ExeTime_{ij} \tag{6}$$

Definition 8: The completion time of task $t_i$ on the virtual machine $vm_j$ (if we want to send $t_i$ to $vm_j$ to run) is defined as follows:

$$ComTime_{ij} = ExeTime_{ij} + CompTime_j \tag{7}$$

According to the above relationship, the task completion time is obtained from the sum of the task execution time and the waiting time for the task to be executed.

Definition 9: The makespan for a set of scheduled tasks is defined as follows:

$$Makespan(f) = \mathrm{Max}\{CompTime_j \mid j = 1, \ldots, n\} \tag{8}$$

Definition 10: Assuming that the task $t_i$ is mapped to the virtual machine $vm_j$, the number of violated deadlines is specified as follows:

$$MissedDeadline(f) = Number\{t_i \mid Deadline_i < (ComTime_{ij} + Delay_i)\} \tag{9}$$

$Delay_i$ specifies the presence duration of the $t_i$ task in the scheduler as follows:

$$Delay_i = ScheduleTime_i - SubmitTime_i \tag{10}$$

$ScheduleTime_i$ specifies the task scheduling moment $t_i$ and $SubmitTime_i$ determines the time of recording task $t_i$ in the scheduler.

Definition 11: The (economic) cost of mapping (total execution cost) for a set of mapped tasks is calculated as follows:

$$ExeCost(f) = \sum_{i=1}^{m} \sum_{j=1}^{n} f(t_i, vm_j) \cdot ExeCost_{ij} \tag{11}$$

Definition 12: After scheduling a set of tasks, the following relation determines the distribution of the load on the virtual machines:

$$LoadBalance(f) = Standard\ Deviation\{CompletionTime_j \mid j = 1, \ldots, n\} \tag{12}$$

Definition 13: A parameter called overall improvement is defined as follows, which determines the overall improvement created by the CM$^3$ method based on all scheduling objectives.

$$OverallImprovement(f) = Normalize(Makespan(f)) \times Normalize(DeadlineViolating(f)) \times Normalize(ExecutionCost(f)) \times Normalize(LoadBalance(f)) \tag{13}$$

Based on the above definitions, the scheduling problem can be formulated as follows:

The input of the problem is a m-member set of tasks belonging to users with different priority levels that have a certain length and deadline, and the output is a function of assigning f such that:

$$\exists g \in \qquad Makespan(f) < Makespan(g) \tag{14}$$

AND

$$MissedDeadline(f) < MissedDeadline(g) \tag{15}$$

AND

$$ExecutionCost(f) < ExecutionCost(g) \tag{16}$$

AND

$$LoadBalance(f) < LoadBalance(g) \tag{17}$$

In the CM$^3$ method, we seek to simultaneously decrease/increase scheduling objectives in mapping tasks to virtual machines, so that the overall compromised response is created and therefore not necessarily all of them are optimal.

## B. Prioritization of Tasks

In the CM$^3$ method, tasks are prioritized based on multi-criteria and dynamic priority. Multi-criteria priority is a combination of the criteria of user level (type), deadline and task length. Therefore, the values corresponding to the mentioned criteria should be normalized to the desired interval [d$_1$, d$_2$] based on the following relation:

$$normX_i = d_1 + \left(\frac{X_i - MinValue}{MaxValue - MinValue}\right) \times (d_2 - d_1) \tag{18}$$

$$d_1, d_2 \in \mathbb{Q}^+$$

After normalizing the values corresponding to the criteria, the multi-criteria priority is determined as follows for each task $t_i$:

$$MultiCriteriaPriority_i = w_u \times UserLevelEffect_i + w_d \times Normalize(DeadlineEffect_i) + w_l \times Normalize(length_i) \tag{19}$$

$$w_u + w_d + w_l = 1$$

The weights w$_u$, w$_d$ and w$_l$ are specified by the system administrator based on user preferences as follows:

$$w_u = \alpha_{udl}$$

$$w_d = \beta_{dl} \times (1 - \alpha_{udl}) \qquad \alpha_{udl} \in [0,1] \tag{20}$$

$$w_l = (1 - \beta_{dl}) \times (1 - \alpha_{udl}) \qquad \beta_{dl} \in [0,1]$$

$\alpha_{udl}$ balances $w_u$ and $\{w_d, w_l\}$, and $\beta_{dl}$ balances $w_d$ and $w_l$. By selecting $\alpha_{udl}$ and $\beta_{dl}$ according to the users' preferences, the system administrator determines the effectiveness of user level criteria, deadline, and length in the task priority.

The user level for each task $t_i$ will be effective in determining the multi-criteria priority as follows:

$$UserLevelEffect_i = a + (UserLevel_i - 1) \times \frac{b-a}{k-1}$$
$$a, b \in [d_1, d_2] \qquad a < b \qquad (b-a) \propto K \tag{21}$$

The higher the level of a user, the earlier the execution of his tasks should be started. Therefore, the task's user level directly affects the determination of UserLevelEffect.
In relationship 21, K denotes the number of user levels, k represents the highest user level and 1 is the lowest user level.
Another effective criterion in determining the multi-criteria priority is the deadline for each task $t_i$ as following:

$$DeadlineEffect_i = \frac{1}{Deadline_i} \tag{22}$$

To improve the makespan of the tasks, the length of the task will directly affect the multi-criteria prioritization. In addition to considering multi-criteria in the priority, dynamic priority also plays a role in determining the final priority of tasks. For this purpose, a dynamic priority value for tasks is calculated during the scheduling period and after the time interval of Δt derived from Equation 23.

$$\Delta t = \frac{SD}{C+1} \tag{23}$$

In this formula, SD specifies the scheduling duration (duration of executing scheduling operations) and C represents the number of calculations of the dynamic priority. The scheduling duration is determined according to the time complexity of the algorithm. The value of C is also set by the system administrator.

After the time period of Δt, the time elapsed from the registration of the task in the scheduler, for the set of unmapped tasks, is calculated as follows:

$$Difference_i = CurrentTime - SubmitTime_i \tag{24}$$

CurrentTime and $SubmitTime_i$ specify the current time and the task registration time in the scheduler, respectively.
Dynamic priority is directly related to Difference value and inversely related to the task deadline which is determined for each task $t_i$ based on the following relation.

$$DynamicPriority_i = w_{diff} \times Normalize(Difference_i) + w_{ddp} \times Normalize(\frac{1}{Deadline_i}) \quad (25)$$

$$w_{ddp} \in [0,1] \qquad w_{diff} = 1 - w_{ddp}$$

The final priority of task $t_i$ is calculated from a combination of multi-criteria priority and dynamic priority as follows:

$$TaskPriority_i = w_{mcp} \times MultiCriteriaPriority_i + w_{dp} \times DynamicPriority_i \quad (26)$$

$$w_{mcp} \in [0,1] \quad w_{dp} = 1 - w_{mcp}$$

According to the above explanations, dynamic prioritization allows changing the task priority during scheduling and in favor of deadline criteria and the time duration of task registration in the scheduler.

## C. Selection of Virtual Machines

In order to meet the scheduling objectives, the most priority task must be mapped to a virtual machine with high processing power, low workload and low usage cost. Therefore, the virtual machine is selected based on the above three criteria as follows:

$$VMSelection_j = w_p \times Normalize(PrPow_j) + w_w \times Normalize(WorkloadEffect_j) + w_c \times Normalize(CostEffect_j) \quad (27)$$
$$w_p + w_w + w_c = 1$$

The values of the weights $w_p$, $w_w$ and $w_c$ are specified as follows according to the needs and importance of the different scheduling objectives:

$$\begin{gathered} w_c = \alpha_{cpw} \qquad \alpha_{cpw} \in [0,1] \\ w_p = \beta_{pw} \times (1 - \alpha_{cpw}) \qquad \beta_{pw} \in [0,1] \\ w_w = (1 - \beta_{pw}) \times (1 - \alpha_{cpw}) \end{gathered} \quad (28)$$

$\alpha_{cpw}$ balances $w_c$ and $\{w_p, w_w\}$, and $\beta_{pw}$ balances $w_p$ and $w_w$. By selecting the values for $\alpha_{cpw}$ and $\beta_{pw}$, the system administrator determines the effectiveness of the usage cost, processing power and workload criteria in selecting the virtual machine.

According to Equation 27 processing power directly affects the selection of virtual machine, and according the following equations, workload and usage cost have the inverse effect:

$$WorkloadEffect_j = \frac{1}{CompTime_j} \quad (29)$$

$$CostEffect_j = \frac{1}{Cost_j} \quad (30)$$

## D. Mapping Tasks to virtual machines using the elastic property

After determining the priority values of tasks and VMSelection of virtual machines, the operation of mapping tasks to virtual machines is started according to the presented explanations and by using the elastic feature.

It should be noted that in the first round of scheduling, WorkloadEffect values cannot be defined because the completion time of all virtual machines is zero. Therefore, VMSelection values are calculated only based on the processing power and usage cost. In the continuation of scheduling steps, in order to normalize and influence WorkloadEffect in selecting the virtual machine, a special method is used as follows.

Virtual machines with zero completion time have better status in terms of workload and their selection probability should be higher. Therefore, the WorkloadEffect values are normalized so that the normalized WorkloadEffect value of such machines be considered equal to $d_2$ and for other virtual machines of the range $[d_1, d_2)$. This normalization process continues until the completion time of all virtual machines gets a value; WorkloadEffect values are then normalized usually through Equation (18).

During task mapping, for the most priority task, the set of candidate virtual machines that satisfy the condition $ComTime_{ij} \leq Deadline_i - Delay_i$ is determined, and the machine with the largest VMSelection is selected in this set. The use of elasticity is considered if there is no candidate virtual machine.

The elasticity of the cloud allows to increase/decrease the capacity of the resources to adequately satisfy the demand. To this end, each virtual machine has an elastic capacity that is provisioned to respond to peak loads. In the CM$^3$ method, this capacity flexibility is used to cover as many deadlines as possible. This feature will be used for possible coverage of the tasks' deadline that will be violated by executing on the normal capacity of virtual machines.

If $t_i$ is the most priority task, $vm_j$ is the virtual machine with the largest value of VMSelection in the virtual machines set, and the sum of completion time of $vm_j$ (before $t_i$ is mapped) and $Delay_i$ is less than the deadline of task $t_i$; the proper execution time to cover the deadline is calculated using the following equation:

$$ProperExeTime_i = Deadline_i - (CompletionTime_j + Delay_i) \quad (31)$$

To achieve proper execution time and coverage of $t_i$ deadline, adequate processing power must be calculated through the following equation:

$$AdequatePrPow_i = \left\lceil \frac{Length_i}{ProperExeTime_i} \right\rceil \quad (32)$$

If the elasticity processing power of $vm_j$ is higher than or equal to the adequate processing power, the processing power of the virtual machine increases to the value calculated in Equation 32, the task $t_i$ is executed on it and the processing power returns to its original value at the end of its execution.

According to the explanations, the time deadline and effort to cover it, in addition to prioritizing the tasks, are also considered in the selection of the virtual machine by using elasticity and determining the candidate virtual machines.

In CM$^3$ method, all tasks are assumed to be of soft real time type. Therefore, the penalty function for each $t_i$ task is defined as follows:

$$P(t_i) = \begin{cases} 0 & if\ C_{ij} + Delay_i \le D_i \\ PenaltyRate_i \times (C_{ij} - D_i) & if\ C_{ij} + Delay_i > D_i \end{cases}$$
$$C_{ij}: ComTime_{ij} \quad D_i: Deadline_i \quad (33)$$
$$\text{PenaltyRate}_i = \frac{UserLevel_i \times MinDeadlineValue \times Length_i}{MaxUserLevelValue \times Deadline_i \times MaxLengthValue}$$

Where, $(C_{ij} - D_i)$ represents the deadline violation rate and $PenaltyRate_i$ also determine the penalty rate of the task $t_i$. Fig. 2 shows the steps of the CM$^3$ method in the form of an operational schema.

## IV. Results and Discussion

The proposed method with three methods of max-min (Max-Min) [17] as the basic method and deadline and suffrage aware method (DSAware) [20] and based on QoS-driven (TS-QoS) [21] due to the nature of the proposed method is compared and evaluated. Table I compares the mentioned methods subjectively.

NetBeansIDE 12.6 code editor and CloudSim 4.0 tools have been used to simulate the methods. It is also assumed that the tasks are computation-intensive tasks, and their main requirement is processing requirement. The data set in [28] has been used to create the required tasks.

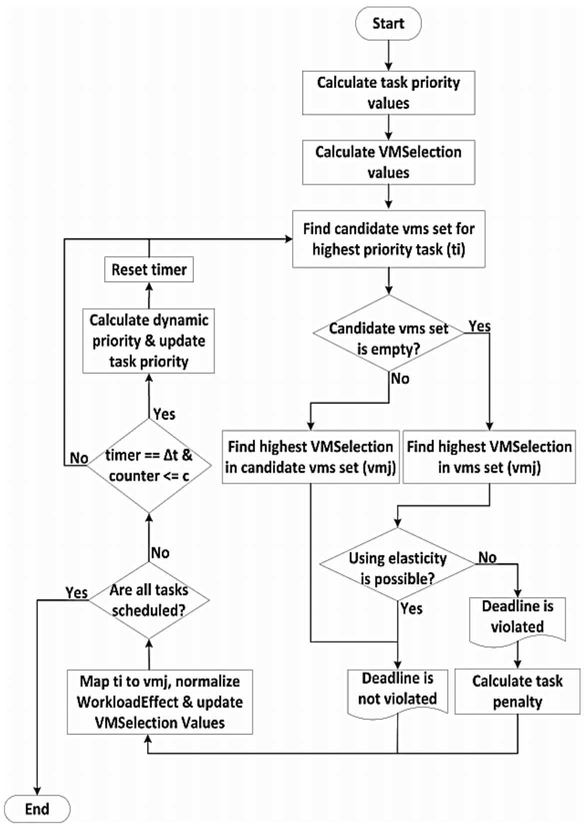

Start
Calculate task priority values
Calculate VMSelection values
Find candidate vms set for highest priority task (ti)
Candidate vms set is empty?
Yes
No
Find highest VMSelection in candidate vms set (vmj)
Find highest VMSelection in vms set (vmj)
Using elasticity is possible?
No
Yes
Deadline is violated
Calculate task penalty
Deadline is not violated
Map ti to vmj, normalize WorkloadEffect & update VMSelection Values
Are all tasks scheduled?
Yes
No
timer == Δt & counter <= c
Yes
No
Calculate dynamic priority & update task priority
Reset timer
End

Fig. 2. Operational schema of CM$^3$ method

TABLE I
SUBJECTIVE COMPARISON OF METHODS

| Algorithm | Objectives | Scheduling criteria | Type | Advantages | Disadvantages |
|---|---|---|---|---|---|
| TS-QoS | makespan, load balancing | user level, length, urgency, duration of presence in the scheduler | heuristic, batch mode, dynamic | possibility of tuning different criteria | selecting virtual machine |
| DSAware | deadline coverage, makespan | deadline | heuristic, batch mode, static | easy implementation, effective deadline coverage | single criterion, selecting virtual machine |
| Max-Min | makespan, load balancing | length | heuristic, batch mode, static | suitable makespan, easy implementation | single criterion, selecting virtual machine |
| CM$^3$ | makespan, deadline coverage, execution cost, load balancing | user level, length, deadline, duration of presence in the scheduler | heuristic, batch mode, dynamic | multi-objective, possibility of tuning different criteria, using elasticity | centralized |

## A. Test 1 (Identical Criteria Weight)

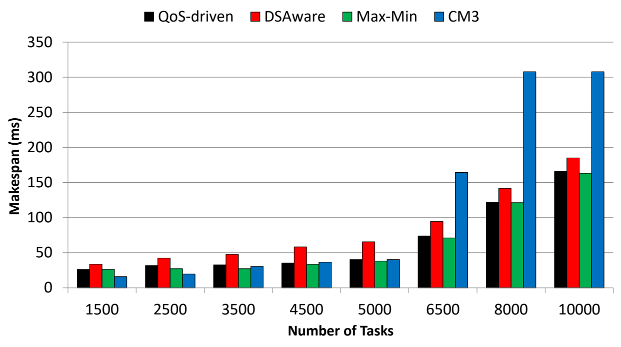


Fig. 3. Makespan – Test 1

The virtual machine selected in the CM$^3$ method does not necessarily provide the minimum completion time for the task. This issue leads to an increase in the completion time of virtual machines and, according to definition 9, an increase in the makespan especially in mapping tasks with a long execution time.

By examining the scheduling steps for the cases of 8000 and 10000 tasks, it is determined that in the first scheduling cycle, a task with a longer execution time than other tasks in the tasks set (350 million instructions) is mapped to a virtual machine with a processing power of 1137 million instructions per second at a cost of approximately 0.0017 \$/s which is the lowest usage cost among all virtual machines. It helps to improve the total execution cost.

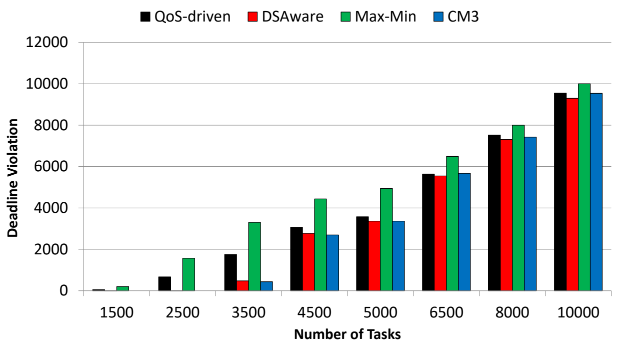


Fig. 4. Deadline violation – Test1

Given that the task deadline for all cases (from 1500 tasks to 10000 tasks) is selected from the range [5, 20], as the number of tasks increases, the number of candidate virtual machines available for the most priority task decreases. Increasing the number of tasks, especially when accompanied by increasing the length of tasks (from 5000 tasks onwards) intensifies the short age of candidate virtual machines. One of the criteria involved in prioritizing a task is length. Very long tasks, especially those with a short deadline and a high user level, reduce the number of candidate virtual machines for subsequent tasks by mapping on virtual machines and increasing machine completion time in the initial stages of scheduling. In addition, the possibility of using the elasticity feature decreases with increasing the completion time (workload) of virtual machines. Thus, the CM$^3$ method is not effective in terms of time deadline coverage for higher number of tasks compared to lower number of tasks as shown in Fig. 4.

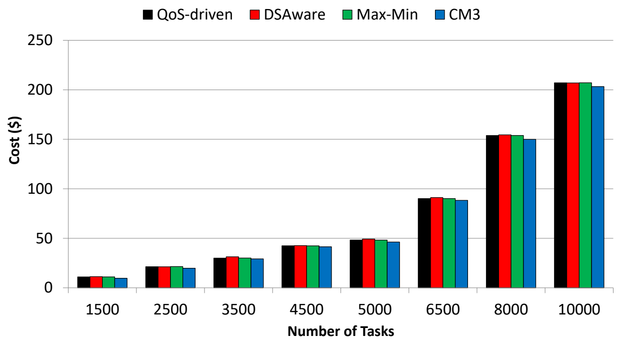


Fig. 5. Total execution cost – Test 1

Since CM$^3$ tends to select virtual machines with low usage costs, this method decreases total execution cost in all cases according to Fig. 5.

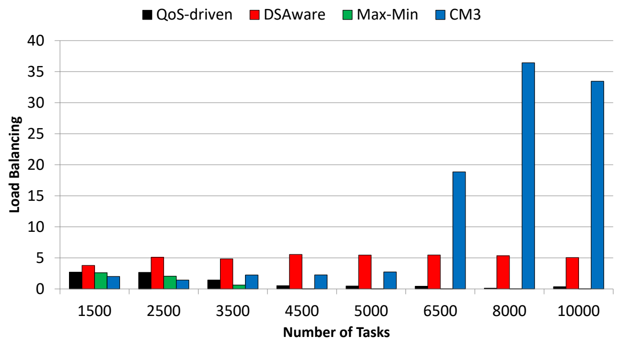


Fig. 6. Load balancing – Test 1

Mapping a task with a long execution time on the virtual machine, which does not necessarily provide the minimum completion time, causes a difference between the completion time of that virtual machine and other virtual machines and causes problem for the load balancing according to definition 12. This happened in the CM$^3$ method in scheduling 6500, 8000 and 10000 tasks, where the data set includes tasks with long execution time as shown in Fig. 6.

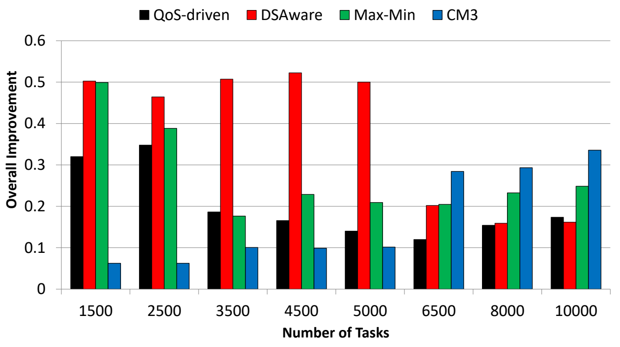


Fig. 7. Overall improvement – Test 1

As the number of tasks increases, the performance of the CM$^3$ method in terms of makespan, load balancing, and to some extent the number of deadline violations decreases. Since the overall improvement rate of the four factors of makespan, number of deadline violations, total execution cost and load balancing are affected equally, therefore, this method has not suitable performance generally for a greater number of tasks according to Fig. 7.

## *B. Test 2 (Increasing the Number of Virtual Machines)*

In the second test, the results are examined by increasing the number of virtual machines to 100. Figures 8, 9, 10, 11, 12 show the effect of this change on the intended objectives.

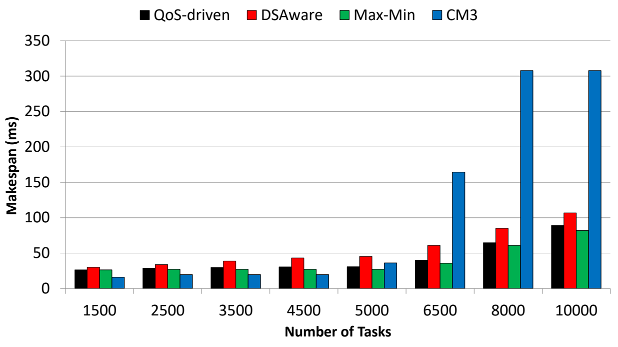


Fig. 8. Makespan – Test 2

The weighting of the criteria and the mapping of the tasks to the virtual machines have not changed in this test. Therefore, the CM$^3$ method according to Fig. 8 has not the proper makespan in the cases of 6500, 8000, and 10000 tasks.

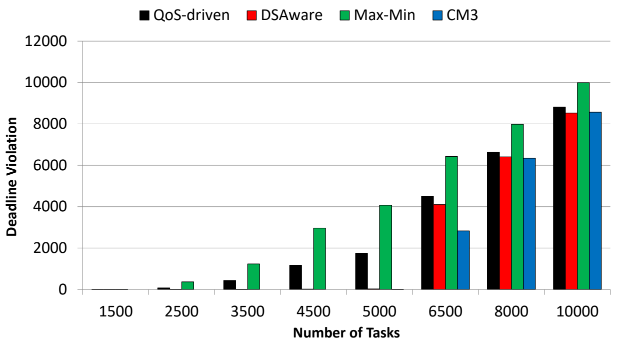


Fig. 9. Deadline violation – Test 2

As the number of virtual machines increases, there will be an appropriate number of candidate virtual machines at each stage of the scheduling and for mapping each task. In addition, due to the reduction of the workload of virtual machines, it is possible to use the elastic property more effectively; therefore, better performance in covering the deadline than the first test has been achieved by comparison of Figures 4, 9.

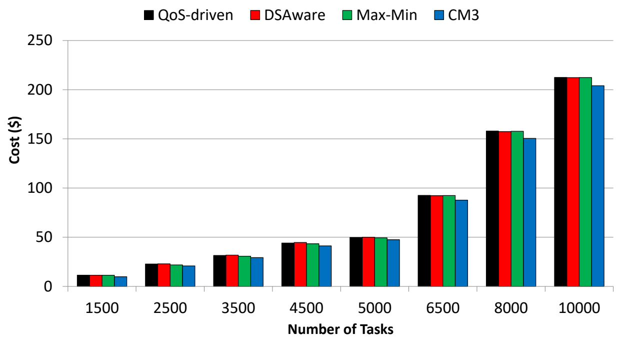


Fig. 10. Total execution cost – Test 2

The CM$^3$ method considers the usage cost in the virtual machine selection, so it has a lower total execution cost in all cases in this test, as shown in Fig. 10.

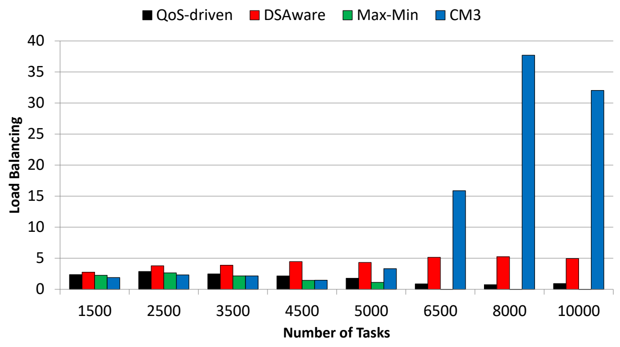


Fig. 11. Load balancing – Test 2

The weights of the criteria and the tasks' mapping rules have not changed in this test. Therefore, the load is more imbalanced for the cases of 6500, 8000, and 10000 tasks in the CM$^3$ method, as shown in Fig. 11.

Increasing the weights of $w_p$ and $w_w$ during scheduling and for specific tasks can help to improve the completion time of virtual machines and thus the load balancing to some extent, although it reduces the role of usage cost in choosing virtual machines.

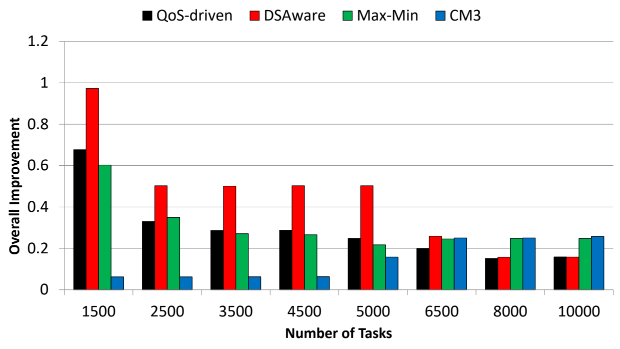


Fig. 12. Overall improvement – Test 2

Increasing the number of virtual machines has had a positive effect on the deadlines covered by the CM$^3$ method, especially for the cases where the tasks set includes a greater number of tasks. Therefore, this method has a better overall performance than the first test and for more tasks according to Figures 7, 12.

### C. *Test 3 (Changing Weights to Reduce Makespan)*

To improve the situation of makespan and load balancing, we consider the weight of the criteria as follows and we repeat the tests on 50 virtual machines.

$$w_u = 0.32 \qquad w_d = 0.33 \qquad w_l = 0.35$$
$$w_c = 0.3 \qquad w_p = 0.34 \qquad w_w = 0.36$$

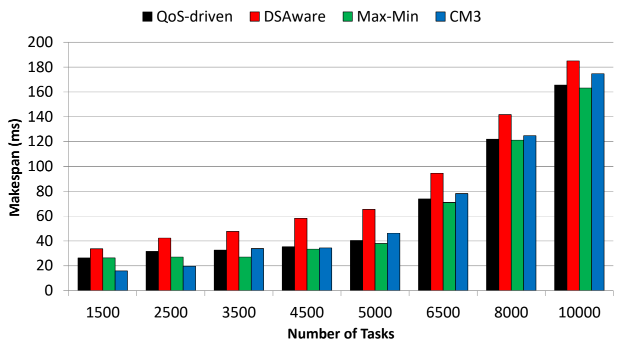


Fig. 13. Makespan – Test 3

By the weights changing and increasing the role of workload and processing power, a virtual machine is selected in each cycle of scheduling operations which provides less completion time for the task compared to the first test. Therefore, a more appropriate mapping is provided in terms of task completion time, and according to the definition 9, the makespan becomes more appropriate as shown in Fig. 13.

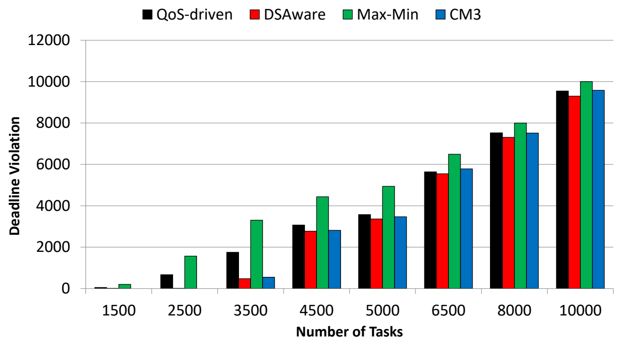


Fig. 14. Deadline violation – Test 3

Due to the increase in the effectiveness of the task length and the decrease in the effectiveness of the deadline in determining the priority of tasks, by comparison of Figures 4, 14 is determined that deadline violations increase compared to the first test. Also, as mentioned in the first test, if the set of tasks includes greater number of tasks, with the progress of scheduling stages, the number of candidate virtual machines as well as the possibility of using elasticity decreases due to the increase in completion time (workload) of virtual machines. Therefore, the deadline violations of the method for 6500, 8000, and 10000 tasks, are more than other cases.

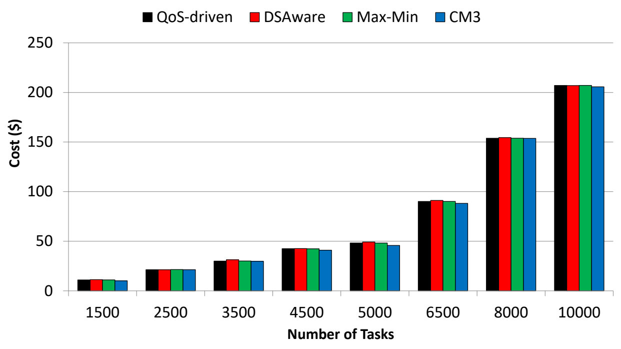


Fig. 15. Total execution cost – Test 3

The CM$^3$ method still has better performance than the other methods in terms of total execution cost due to considering the cost of using virtual machines in scheduling operations. However, according to the reduction in the weight of $w_c$ in the selection of the virtual machine, the total execution cost has increased compared to the first test and in most cases.

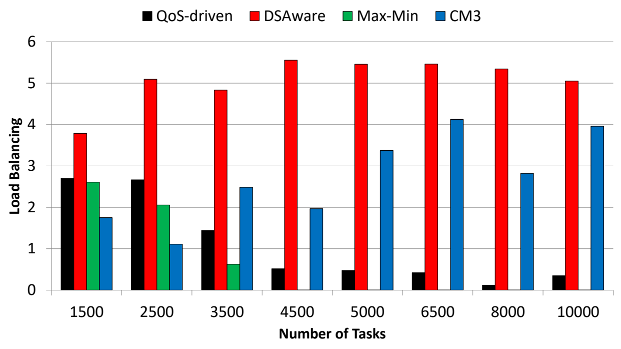


Fig. 16. Load balancing – Test 3

Given the changes in the weight of the criteria, the selected virtual machine in each mapping provides a less completion time for the task. So the completion time of the virtual machines does not deviate much from the average completion time at the end of the scheduling operation, the load balancing has been improved compared to the first test by comparison of Figures 6, 16.

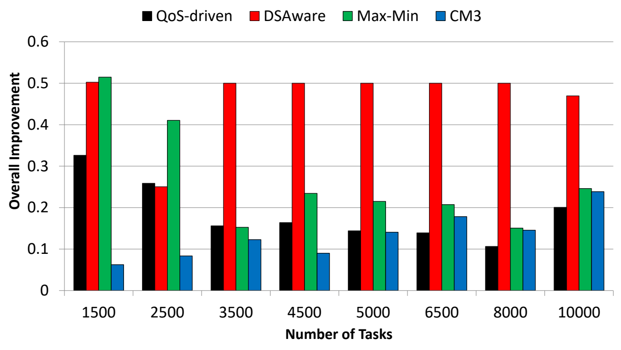


Fig. 17. Overall improvement – Test 3

In this test, the better status of makespan and load balancing have improved the overall performance of the CM$^3$ method, especially for 6500, 8000, and 10000 tasks, compared to the first test and according to Figures 7, 17.

## D. *Different Applications of the Proposed Method*

In the CM$^3$ method, the weights of effective criteria in prioritizing tasks and selecting virtual machines can be changed as needed. For example, in the forthcoming tests the results of the changes of $w_u$, $w_c$ will be examined and evaluated.

Increasing the value of $w_u$ in Equation 19 will decrease the completion time of the tasks with higher user level. Fig. 18 has shown the effect of increasing this weight at the completion time of the tasks of level 3.

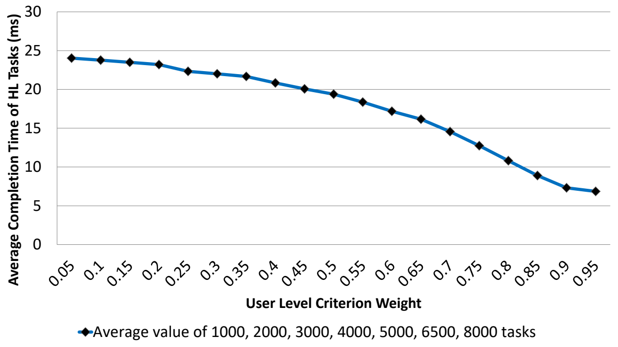


Fig. 18. Influence of user level criterion weight on the completion time of highest-level tasks

To reduce the total execution cost of the tasks, the weight of the relevant criterion in the virtual machine selection relation should be increased. In this test, the weight of different criteria is determined as follows:

$$w_c = 0.05 + i \times 0.05 \qquad i = 0, 1, .., 18$$
$$w_p = w_w = 0.475 - i \times 0.025 \qquad w_u = w_d = w_l$$

## V. Conclusion

The results of various tests show that the CM$^3$ method in most cases, and especially compared to the TS-QoS method which is a multi-criteria method, has been able to improve the makespan and load balancing. Although CM$^3$ sometimes has weaker performance than the single criterion (DSAware) method, it has acceptable performance in terms of number of deadline violations. It also works better than other methods in terms of execution cost in almost all cases. In addition, depending on the situation and per the desired objective, by changing the weight of the criteria, the desired result can be achieved. This factor distinguishes the proposed method from other methods in addition to considering different effective criteria.

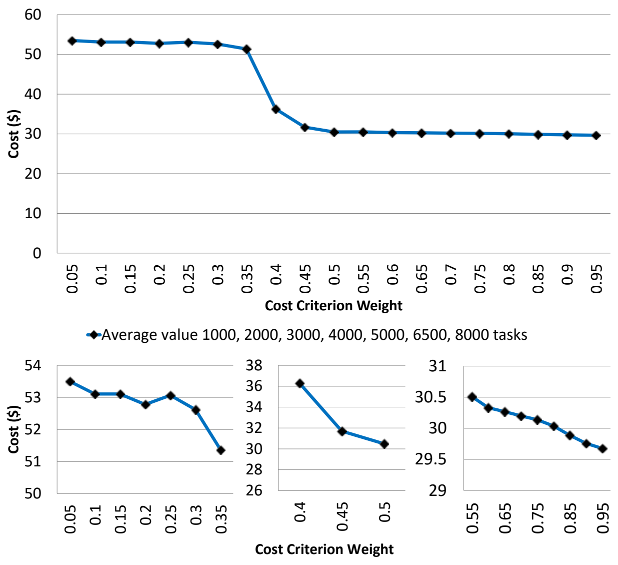


Fig. 19. Influence of cost criterion weight to decrease execution cost

Although the statement of the proposed method's adaptability may be seemed ambitious, the use of artificial intelligence algorithms to adjust the weight of the criteria does not take this away from the mind. Therefore, as a future work, the CM$^3$ method can be adapted based on scheduling objectives and during scheduling operations. Other suggestions for future studies are the distribution of the scheduling unit, determination of the value of parameter C to achieve the best state of makespan and load balancing, using the CM$^3$ method in a hybrid cloud environment to further reduce the total execution cost, generalizing the criteria for selecting virtual machines, and using the method for dependent tasks.

## References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, 2011, doi: 10.6028/NIST.SP.800-145.

[2] H. Ji et al., "Adaptive workflow scheduling for diverse objectives in cloud environments," Transactions on Emerging Telecommunications Technologies, Vol.28, No.2, pp. e2941, 2015, doi: 10.1002/ett.2941.

[3] A.V. Lakra and D.K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," Procedia Computer Science, Vol.48, pp. 107-113, 2015, doi: 10.1016/j.procs.2015.04.158.

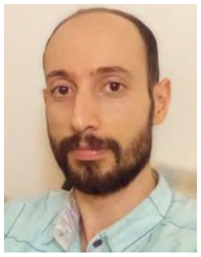
[4] D. Kaur and T. Sharma, "Scheduling Algorithms in Cloud Computing," International Journal of Computer Applications, Vol.178, No.9, pp. 16-21, 2019, doi: 10.5120/ijca2019918801.

[5] J. Samriya and N. Kumar, "A QoS Aware FTOPSIS-WOA based task scheduling algorithm with load balancing technique for the cloud computing environment," Indian Journal of Science and Technology, Vol.13, No.35, pp. 3675-3684, 2020, doi: 10.17485/IJST/v13i35.1314.

[6] S. Xue et al., "QET: a QoS-based energy-aware task scheduling method in cloud environment," Cluster Computing, Vol.20, No.4, pp. 3199-3212, 2017, doi: 10.1007/s10586-017-1047-5.

[7] A. Chhabra et al., "QoS-Aware Energy-Efficient Task Scheduling on HPC Cloud Infrastructures Using Swarm-Intelligence Meta-Heuristics," CMC-COMPUTERS

MATERIALS & CONTINUA, Vol.64, No.2, pp. 813-834, 2020, doi: 10.32604/cmc.2020.010934.
[8] S. Mira, "Task Scheduling Balancing User Experience and Resource Utilization on Cloud," A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering, Rochester Institute of Technology, 2019.
[9] R. Khorsand and M. Ramezanpour, "An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing," International Journal of Communication Systems, Vol.33, No.9, pp. e4379, 2020, doi: 10.1002/dac.4379.
[10] G. Muthusamy and S.R. Chandran, "Cluster-based Task Scheduling Using K-Means Clustering for Load Balancing in Cloud Datacenters," Journal of Internet Technology, Vol.22, No.1, pp. 121-130, 2021.
[11] S.E. Shukri et al., "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," Expert Systems with Applications, Vol.168, pp. 114230, 2020, doi: 10.1016/j.eswa.2020.114230.
[12] M. Hussain et al., "Energy and Performance-Efficient Task Scheduling in Heterogeneous Virtualized Cloud Computing," Sustainable Computing: Informatics and Systems, Vol.30, pp. 100517, 2021, doi: 10.1016/j.suscom.2021.100517.
[13] A. Gupta et al., "Load balancing based hyper heuristic algorithm for cloud task scheduling," Journal of Ambient Intelligence and Humanized Computing, 2020, doi: 10.1007/s12652-022-04238-5.
[14] H. Yuan et al., "Revenue and energy cost-optimized biobjective task scheduling for green cloud data centers," IEEE Transactions on Automation Science and Engineering, pp. 1-14, 2020, doi: 10.1109/TASE.2020.2971512.
[15] A. Thomas et al., "Credit Based Scheduling Algorithm in Cloud Computing Environment," Procedia Computer Science, Vol.46, pp. 913-920, 2015, doi: 10.1016/j.procs.2015.02.162.
[16] O. Elzeki et al., "Overview of scheduling tasks in distributed computing systems," International Journal of Soft Computing and Engineering (IJSCE), Vol.2, No.3, pp. 470-475, 2012.
[17] S. Mohapatra et al., "A Comparative Study of Task Scheduling Algorithm in Cloud Computing," (Springer, edn.), pp. 325-338, 2020, doi: 10.1007/978-981-15-1483-8_28.
[18] S. Devipriya and C. Ramesh, "Improved max-min heuristic model for task scheduling in cloud," (IEEE, edn.), pp. 883-888, 2013, doi: 10.1109/ICGCE.2013.6823559.
[19] T. Mathew et al., "Study and analysis of various task scheduling algorithms in the cloud computing environment," (IEEE, edn.), pp. 658-664, 2014, doi: 10.1109/ICACCI.2014.6968517.
[20] V. Poonam Chaudhary, "Deadline and Suffrage Aware Task Scheduling Approach for Cloud Environment," International Research Journal of Engineering and Technology, Vol.4, No.8, pp. 972-977, 2017.
[21] X. Wu et al., "A task scheduling algorithm based on QoS-driven in cloud computing," Procedia Computer Science, Vol.17, pp. 1162-1169, 2013, doi: 10.1016/j.procs.2013.05.148.
[22] R.N. Calheiros et al., "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, Vol.41, No.1, pp. 23-50, 2011, doi: 10.1002/spe.995.
[23] N. Rajak and D. Shukla, "A Systematic Analysis of Task Scheduling Algorithms in Cloud Computing," (Springer, edn.), pp. 39-49, 2020, doi: 10.1007/978-981-15-2071-6_4.
[24] R. Gulbaz, "Task Scheduling Optimization in Cloud Computing," A thesis submitted in partial fulfillment for the degree of Master of Science, Capital University of Science and Technology, 2020.
[25] R. Chen et al., "A Cloud Task Scheduling Algorithm Based on Users' Satisfaction," (IEEE, edn.), pp. 1-5, 2013, doi: 10.1109/ICNDC.2013.11.
[26] H. Han et al., "A Qos Guided task Scheduling Model in cloud computing environment," (IEEE, edn.), pp. 72-76, 2013, doi: 10.1109/EIDWT.2013.17.
[27] Y. Fan et al., "Executing Time and Cost-Aware Task Scheduling in Hybrid Cloud Using a Modified DE Algorithm," (Springer, edn.), pp. 74-83, 2015, doi: 10.1007/978-981-10-0356-1_8.
[28] "The LCG Grid log," Available: http://www.cs.huji.ac.il/labs/parallel/workload/l_lcg/index.html, 2005.



**Ehsan Shojaeian** received the B.S. degree in Computer Engineering from Razi University, Kermanshah, Iran, and the M.S. degree in Computer Engineering from Science and Research University, Tehran, Iran. Currently, He is a research fellow in the Department of Computer Engineering, Islamic Azad University Science and Research Branch, Tehran, Iran. His research interests include cloud computing, security and privacy issues in internet of things devices, and big data analysis.

**Mehran Mohsenzadeh** is a faculty member in the Department of Computer Engineering, Islamic Azad University Science and Research Branch, Tehran, Iran. He received the B.S. degree in Computer Engineering from Shahid Beheshti University, Tehran, Iran, and the M.S. degree and Ph.D. degree in Computer Engineering from Science and Research University, Tehran, Iran. He is an associate editor of the Journal of Advances in Computer Engineering and Technology. His main research interests include database, big data, cloud computing, and software engineering and he has published more than eighty articles (author/coauthor) in international conferences and journals.

**Mohammad Mehdi Sahrapour** received the B.S. degree in Electrical Engineering from Islamic Azad University, Tehran, Iran, and the M.S. degree in Business Administration from Amirkabir University, Tehran, Iran. He pursued and successfully completed a second Master's degree in Project Management from Monash University, Caulfield East VIC, Australia. His research interests include digital transformation and the potential of computing and artificial intelligence in revolutionizing digital services.