Date of publication Jun-28, 2021, date of current version Aug-08, 2020.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.20.2.2163

Genetic-Based Task Scheduling Algorithm with Dynamic Virtual Machine Generation in Cloud Computing

AHMED A. A. GAD-ELRAB[1,2], TAMER A.A. ALZOHAIRY[2], KAMAL R. RASLAN[2], FAROUK A. EMARA[2]

[1]King Abdulaziz University, Jeddah, Saudi Arabia (e-mail: asaadgad@azhar.edu.eg) (www.kau.edu.sa)
[2]Department of Mathematics and Computer Science, Faculty of Science, Al-Azhar, University – Cairo, Egypt
(e-mail: asaadgad@azhar.edu.eg, tamer@azhar.edu.eg, kamal_raslan@yahoo.com, aly_emara86@azhar.edu.eg) (www.azhar.edu.eg)

Corresponding author: Ahmed A. A. Gad-Elrab (e-mail: asaadgad@azhar.edu.eg).

ABSTRACT Recently, cloud computing has become the most common platform in the computing world. scheduling is one of the most important mechanism for managing cloud resources. Scheduling mechanism is a mechanism for scheduling user tasks among datacenters, host and virtual machines (VMs) and is an NP completeness problem. Most of existing mechanisms are heuristic and meta-heuristic methods, developed to address a part of scheduling problem and did not consider the dynamic creation of VMs by taking into account the required resources for a user task and the capabilities of a set of available hosts. To deal with this dynamic behavior, this paper introduces a new mechanism that uses a genetic algorithm (GA) for establishing a flexible scheduling mechanism that can adapt the dynamic number of VMs based on the required resources by user tasks and the available resources of hosts. Simulation results show that the proposed algorithm can distribute any number of user tasks on the available resources and it achieves better performance than existing algorithms in terms of response time, makespan, FlowTime, throughput, and resource utilization.

KEYWORDS task scheduling; makespan; virtualization; virtual machine; dynamic creation.

I. INTRODUCTION

CLOUD computing (CC) is defined as the collection of computing and communication resources over the distributed datacenters and is shared by different users [1]. Datacenters in many physical servers are linked with high speed networks and ready for computing services by responding to specific requests and supporting multiple virtual machines (VMs) by dedicating to different tasks for each. The VMs run a task and when the task is completed or allocated to another task, it shuts down [2, 3].

In cloud computing, virtualization of resources is one of its characteristics, which is a backbone of cloud computing. It is another key technology and it allows creating a large number of less powered servers for a small number of high-powered servers while maximizing resource utilization efficiency and reducing the overall cost in power, space and other infrastructure by improving physical resource sharing [4]. Virtualization depends on less physical and more logical view of resources by isolating the storage and computing services away from the details of implementation. If there is any failure in a physical server for a certain reason, this server can be dropped from the pool of available resources. In this case, selecting other physical servers for deploying VMs until the failure is corrected and established (this process called migration). This dynamic migration process improves the service availability which will be attractive for many users. Uninterrupted service is one of the advantages of using cloud computing [5] that is needed to manage physical and virtual resources. Scheduling physical and virtual resources plays a vital role in resources management of cloud computing. Formally, scheduling tasks on VMs or scheduling VMs on physical resources to achieve the objective of a provider or a user such as load balancing,

resource utilization, energy efficiency, migration of tasks, Quality of Service (QoS) or another objective is important [6]. Datacenter broker is one of the main components in cloud computing environment which is the backbone of scheduling process [7]. The first process in scheduling is discovering and filtering of resources where datacenter broker discovers all available resources in the network and collects the related information status to them. The second process is resource selection, where required resource is selected using certain parameters of resources and a task. The final process is the task submission, where the selected resource receives the submitted task.

Generally, there are two main approaches which are introduced for performing scheduling in cloud computing. The first approach is task scheduling on VMs, which determines the efficient VM for executing the task. The second approach is scheduling VMs on physical resources which is finding the optimal allocation of VMs on the physical servers available in the datacenters. The simplest and trivial algorithm to scheduling resources is the First-Come-First-Serviced, FCFS (instead called, First-In-First-Out, FIFO). This algorithm depends on the arrival time of requesting a resource. It does not consider the execution time of a task or a resource utilization and selects the resources randomly. Also, a task may wait for a very long time to be submitted to a virtual machine. This means that the starvation problem can occur, which is one of the big problems for task scheduling in cloud computing. Round Robin (RR) is another method in cloud computing, which depends on VM that will execute its tasks based on a time interval called quantum. RR can solve starvation problem, but it does not consider the objective of a user or a cloud vendor, for example, minimizing the execution time or the load on resource and it selects the resources randomly [8].

In recent years, a lot of studies have been appeared to schedule tasks on VMs [9-16]. The main goal of these algorithms is searching for the solution that minimizes the execution cost and makespan by using intelligent approaches such as Genetic algorithm (GA), chaotic social spider algorithm (CSSA), ant colony optimization (ACO), fuzzy theory, Simulated Annealing (SA), particle swarm optimization (PSO) or hybrid algorithm. Nevertheless, they did not consider some important parameters such as time complexity of the algorithm and some tasks may be assigned to VM that does not have the minimum execution time for them. Other studies have appeared to schedule VMs on physical resources [17-19]. The main objective of these algorithms is searching for the solution that minimizes the wastage rescues, maximizes rescue utilization and achieves load balancing.

Disadvantages of these algorithms are: (1) the VMs is created based on the total available power processing in the hosts of the cloud (like processing cores, processing speed and memory) and (2) they did not take into account the required resources for user tasks.

This paper proposes a new algorithm for solving the scheduling problem in cloud computing to schedule VMs which are created dynamically on the cloud based on the available resources of hosts and the required resources for each task by using modified genetic algorithm (GA).

II. RELATED WORK

While task scheduling algorithms are focused on the performance efficiency, VM scheduling algorithms are focused on the resource utilization efficiency of a cloud. Meta-heuristic and Heuristic techniques were developed for solving scheduling problem in cloud computing, which is an NP-completeness problem. In [20], the First In First Out (FIFO) algorithm for resources scheduling depends on the arrival time of the requesting resource task (first task arrived will be submit first). Also, Longest Job First (LJF) and Shortest Job First (SJF) are scheduling algorithms. SJF algorithm sorts tasks based on the number of instructions of each task in ascending order and submits the task to a resource based on the shortest order (the first task in the order will be submitted first), on the other hand, in the Longest Job First (LJF) algorithm, the last task in the order will be submitted first. The main problem of SJF, LJF and FIFO algorithms is that they do not consider the objective of a user or a cloud vendor, so these algorithms are useless, for example, minimizing execution time or load on a resource in the cloud homogeneous environment (the same number of instructions of each task and the same processing power of VM). Also, in SJF, LJF and FCFS algorithms the resources are selected randomly, and the task may wait for a very long time to be submitted to a virtual machine. This means that the starvation problem can occur, which is one of the big problems of task scheduling in cloud computing. Another algorithm in cloud computing is called Round Robin (RR) which depends on VM that will execute its tasks by using a time interval called quantum. This algorithm solves starvation problem, but it does not consider the objective of a user or a cloud vendor [8]. Fang et al. [21] proposed a scheduling task approach based on load balancing in cloud computing where the VM is described according to the needed resources for executing a task. Next it sorts the hosts in ascending order based on their processing power. Then VM selects a host that can provide the required resources and the load is lightest. Finally, if a task has been completed, the VM will be destroyed. Disadvantage of this work is creating VM for each task which is over head time. Also, if the resources that are needed to deploy VM in host are not available, then the VM waits for the second scheduling, which does not make it possible to achieve the objective of a user. Sindhu and Mukherjee [9] introduced two scheduling methods to schedule tasks, Shortest Cloudlet Fastest Processing Element (SCFP) and Longest Cloudlet Fastest Processing Element (LCFP). In LCFP, the task that has a large number of instructions is mapped to VM that has high computation power for minimizing the makespan. In SCFP, the task that has a small number of instructions is mapped to

VM that has high computations power for reducing FlowTime (completion time summation of a set of tasks). LCFP can minimize makespan but some tasks will be assigned to VM that does not have the minimum execution time for them, while SCFP can minimize FlowTime but maximizes makespan and decreases resource utilization. Also, LCFP may face starvation problem. Alworafi et al. [16] introduced Hybrid-SJF-LJF (HSLJF) algorithm which combines LJF and SJF algorithms. HSLJF sorts the tasks in ascending order, then selects one task based on SJF and another task based on LSF. Finally, the selected task is submitted to select VM that has minimum completion time. The main problem of HSLJF is its useless in the cloud homogeneous environment and the characteristics of VM (like processing speed, processing cores and memory) do not depend on the required resources for each task of a user.

Applying an optimization technique may help to find the solution to task scheduling problem in cloud computing. Many of intelligent approaches were developed to obtain optimal solution. Zhao et al. [10] proposed an optimized algorithm by using GA for scheduling independent tasks to minimize the execution time. It can minimize makespan and dead line time, but some tasks will be assigned to VM that does not have the minimum time for executing them. Also, the characteristics of VM do not depend on the required resources for each task of a user. Jena [12] used nested Particle Swarm Optimization for optimizing processing time and energy. The disadvantage of this algorithm is a time complexity. Also, the characteristics of VM do not depend on the required resources for each task of a user. Arul Xavier and Annadurai [14] proposed algorithm called chaotic social spider algorithm inspired (CCSA) which uses social spider for solving the problem of task scheduling. The objective of CCSA is minimizing makespan with effective load balancing. The disadvantage of CCSA is that VM is not selected according to the resources that are needed by a task and a time delay for each searching agent (SA) for sending and receiving data to others. In addition, each SA is a computing agent which consumes computing capability of cloud computing. Also, time complexity of algorithm is ignored. Nasr et al. [22] introduced efficient technique by converting the problem of scheduling task into an instance of the Traveling Salesman Problem (TSP), then applied one of TSP solution strategies to solve the problem. The disadvantage of this technique is that VM is not selected according to the resources that are needed by a task.

Most of current techniques can solve a part of the scheduling problem efficiently, but they are unable to solve all aspects of the problem. For example, they can minimize makespan, but some tasks will be assigned to VM that does not have the minimum execution time for them. Also, they cannot determine the idle number of VMs on idle hosts and idle cores of VMs or hosts, which will decrease resource utilization. In addition, some of them did not consider certain important parameters as time complexity of the algorithm.

III.TASK SCHEDULING PROBLEM IN CLOUD COMPUTING

A. PROBLEM DESCRIPTION

Cloud computing environment is a heterogeneous environment where the tasks of users are different in a number of instructions, input data, output data, etc. So, the required resources for each user task are different. Also, these tasks may be dependent or independent, the number of available hosts in datacenter are different, and each host has multiple core processors and the total available processing power of each host (like processing cores, processing speed and memory) are not the same. Virtualization is a backbone of cloud computing system, where tasks of users distributed on VMs are deployed on hosts. The characteristics of VMs (like processing cores, processing speed and memory) are different. VMs are deployed on hosts based on the total available processing power of each host. So, we are facing the problem of distributing tasks on VMs and deploying VMs on hosts, such that the number of VMs and the characteristics of VMs should depend on the required resources for the tasks of the users and the required resources of VMs that can be provided by hosts. Creating VMs based on the required resources for the tasks of the users may not only be deployed on hosts (required resources of VMs cannot be provided by hosts) or may be deployed with idle number of cores in hosts (wastage resources). Also, creating VMs based on the available resources in hosts may generate several VMs that are more than the number of tasks, this means that there is an idle number of VMs or a few VMs cores and it is more than the number of required cores for the tasks of the users. In addition, the system performance is optimized or the time consumption for processing tasks of users is minimized (objective of users). Furthermore, maximum time consumption for executing all tasks is minimized (objective of cloud vendor).

As a result, there are two scheduling approaches to using cloud computing. The first approach is scheduling the tasks of users to VMs and the second approach is scheduling VMs to host resources. To achieve the objective of a user, the characteristics of VMs are based on the required resources for each user task, but the VM may require resources which cannot be provided by the hosts. On other hand, to achieve the objective of a vendor, the characteristics of VMs are based on available processing power in the hosts but tasks may need resources that cannot be provided by the VMs. In two cases, there are idle resources of VMs or host.

B. PROBLEM FORMULATION

In cloud computing system there is a set of clients (users), $U = \{u_1,u_2,.....,u_n\}$ and each user, u_i has a set of tasks, $T_i = \{t_{i1},t_{i2},...,t_{im}\}$, where $\{1,2,.....m\}$ are the identifiers of tasks called the *id* of a task which is unique for each task. Assume that there is a set of objectives for users$OU = \{ou_1,ou_2,...,ou_n\}$. Also, there is a set of distributed datacenters $D=\{D_1,D_2,....D_l\}$ and each datacenter, D_j has a set of physical resources $PR_j=\{R_{j1},R_{j2},...,R_{jy}\}$ where $\{1,2,.....y\}$ are the identifiers of physical resources and each

identifier (*id*) is unique for each physical resource. For each physical resource R_{jk} in PR_j, there is a set of virtual machines, $VM_{jk} = \{vm_1^{jk}, vm_2^{jk}, ..., vm_z^{jk}\}$ will be deployed to obtain the objectives of a user or the objectives of a cloud vendor, where *{1,2,.....z}* are the identifiers of VMs and each identifier is unique for each *vm*.

Many parameters need to be considered to achieve the objectives of a user (e.g., completion time, cost, and response time) or the objective of a cloud vendor (e.g., resource utilization, fault tolerance, and power consumption). For a task, t_{is} of a user u_i, assume that the arrival time, the started execution time on vm_x^{jk}, and the waiting time (i.e, the interval time elapsed between the starting execution time and the arrived time) are denoted as at_{is}, $st_{is,x}^{jk}$ and wt_{is} respectively. The waiting time wt_{is} is calculated as follows:

$$wt_{is} = st_{is,x}^{jk} - at_{is}, \quad (1)$$

where $1 \le i \le n,\ 1 \le s \le m,\ 1 \le j \le l,\ 1 \le k \le y,\ 1 \le x \le z.$ Assume that the execution time of task $t_{is,}$ which is the expected interval time elapsed to execute task on virtual machines vm_x^{jk} is represented by $ext_{is,x}^{jk}$. And assume that the completion time of task $t_{is,}$ which is the expected time for task finished execution is represented by $ct_{is,x}^{jk}$. If the tasks of users are independent, then the completion time of a task $ct_{is,x}^{jk}$ is calculated as follows:

$$ct_{is,x}^{jk} = \mathrm{st}_{is} + ext_{is,x}^{jk}, \quad (2)$$

where $1 \le i \le n,\ 1 \le s \le m,\ 1 \le j \le l,\ 1 \le k \le y,\ 1 \le x \le z.$ If the scheduling is non-preemptive and tasks of users are dependent, then $ct_{is,x}^{jk}$ is calculated as follows:

$$ct_{is,x}^{jk} = \mathrm{st}_{is} + ext_{is,x}^{jk} + \sum_{a=1}^{m} ext_{ia,x}^{jk} \quad (3)$$

$\forall\ 1 \le i \le n,\ 1 \le s \le m,\ 1 \le j \le l,\ 1 \le k \le y,\ 1 \le x \le z\ s \ne a$

where

$$ext_{is,x}^{jk} = \frac{l_{is}}{p_x^{jk}}. \quad (4)$$

$\forall\ 1 \le i \le n,\ 1 \le s \le m,\ 1 \le j \le l,\ 1 \le k \le y,\ 1 \le x \le z$, and l_{is} is the total number of instructions of task t_{is} and p_{jkx} is the total processing power of vm_x^{jk}, which is deployed on a host k in datacenter j and $ext_{ia,x}^{jk}$ is the execution time of related tasks to task t_{is}. Assume that makespan is denoted as *mk*, which is the completion time of the last task of all users and is calculated as follows:

$$mk = max\ (ct_{is,x}^{jk}). \quad (5)$$

The main objective of the proposed solution is to minimize makespan Eq. (6):

$$minimze\ (mk), \quad (6)$$

in such a way that

$$\sum_{x=1}^{z} NCVM_{kx} * PSVM_{kx} \le NCH_k * PSH_k \quad (7)$$

$$\sum_{x=1}^{z} MVM_{kx} \le MH_k, \quad (8)$$

where $1 \le k \le y$. The first condition in eq. (7) means that the total processing capacity of all VMs, which are deployed on host H_k is less than or equal to the total processing capacity of it, where $NCVM_{kx}$ represents the number of cores of vm_{kx}, $PSVM_{kx}$ is the processing speed of each core of $vm_{kx,}$ which is measured and denoted by the number of Million Instructions Per Second (MIPS), NCH_k represents the number of cores of host H_k, and PSH_k is the processing speed of each core host $H_{k,}$ which is also measured and denoted by the number of Million Instructions Per Second (MIPS). While the second condition in eq. (8) means that the total memory size requested by all VMs, which are deployed on a host H_k is less than or equal to the memory size of a host H_k, where MVM_{kx} represents the memory size of vm_{kx} and MH_k represents the memory size of a host H_k.

IV. THE PROPOSED TASK SCHEDULING ALGORITHM

A. BASIC IDEA

Efficient task scheduling in the cloud environment for multiple tasks which are submitted by a user is one of the most challenging problems. The main process in task scheduling is the generation of virtual machines (VMs). To make the best task scheduling in cloud environment, it is necessary to take into account the required resources for tasks and the available resources of cloud hosts in the generation process of VMs. In addition, the resource scheduling must satisfy the objectives of users and cloud vendors and improve the overall performance of the cloud computing environment. To solve these problems and satisfy these objectives, an adaptive task scheduling algorithm called ***G**enetic-Based **T**ask **S**cheduling with **D**ynamic **V**irtual Machine **G**eneration Algorithm (GTSwDVG)* is proposed.

The basic idea of *GTSwDVG* is based on: (1) sending the information about all available resources of each host (available number of core, speed of core, available memory and network bandwidth) in each each datacenter and the properties of available VMs in each host to a datacenter broker; (2) sending the information about the tasks of users (required resources) to a datacenter broker; (3) using genetic algorithm (GA) by a datacenter broker to distribute tasks on hosts, dynamically; and (4) generating VMs adaptively to execute tasks on these hosts based on the required resources

and the available resources of hosts. Based on these issues, *GTSwDVG* can get the best distribution of tasks on the available hosts with suitable properties of VMs to execute these tasks. In addition, *GTSwDVG* can create and destroy VMs, dynamically based on the resulted distribution of GA for task scheduling.

B. THE PROPOSED ALGORITHM

For task scheduling, *GTSwDVG* uses genetic algorithm (GA) for distributing tasks and generating VMs dynamically on available hosts of datacenters by using three tuples: (1) a tuple for representing the IDs of tasks, (2) a tuple for representing the IDs of hosts, and (3) a tuple for representing the IDs of VMs. Therefore, *GTSwDVG* proposed a matrix structure for representing the chromosome of GA, which combines the three different tuples. By using this matrix structure, the flow chart of GA processes is shown in Fig. 1.

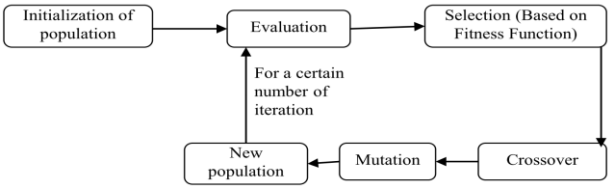

Figure 1. Flowchart of genetic algorithm

In the next subsections, these processes will be described in details.

B.1 INITIALIZATION PROCESS

Genetic algorithm begins with the initialization of population. The initial population is the set of all individuals that are used in the GA to represent the possible solution to the scheduling problem. Every individual is represented as a chromosome, which is one such solution. Every chromosome consists of a set of genes. Gene is one element position of a chromosome. The value of gene is taken for a particular chromosome called Allele. The proposed solution represents the chromosome by 3 x *m* matrix, where *m* is the number of tasks. And the first row represents id of a task, the second row represents id of a host, and the third row is a random number that is less than *m* and represents the identifier of VMs as shown in Fig. 2. After that the proposed solution determines the properties of VMs based on required resources for executing tasks and resources provided by hosts.

As shown in Fig. 2, tasks having ids 0, 4 will be executed on *vm* having id 1, while task having id 1 will be executed on *vm* having id 2. VMs having ids 1,2 will be deployed on host having id 1, so the properties of *vm* having id 1 should be based on the required resources by tasks that have ids 0,4 and the properties of *vm* having id 2 are based on the task having id 1. Also the properties of VMs that have ids 1,2 should take into account resources that can be provided by host having id 1.

T_id	0	1	2	3	4	5	6	7	8	9	10	11	12
H_id	1	1	2	3	1	2	3	1	2	2	3	3	3
Vm_id	1	2	4	6	1	5	6	3	5	4	6	6	6

Figure 2. Chromosome representation

B.2 EVALUATION PROCESS

Each chromosome is evaluated by a fitness function (scheduling objective). Fitness function measures the fitness value of a chromosome. The fitness value determines the performance of an individual in the population. The Fitness function of the proposed solution is to minimize the makespan Eq. (6).

B.3 SELECTION PROCESS

This process selects individuals from the population for mating. There are various selection mechanisms to select the best chromosomes such as selection based on rank, roulette wheel, tournament selection, and Boltzmann strategy.

B.4 CROSSOVER PROCESS

Crossover operation can be applied by selecting two individuals and using one of the two kinds of the crossover operators, that are single point crossover and order crossover operators. Crossover operation is applied on the second row of the matrix (id of host). Then a random number is generated for the third row (id of VM). After that the properties of VM are determined based on the required resources for executing tasks and the resources provided by hosts (see Fig. 3).

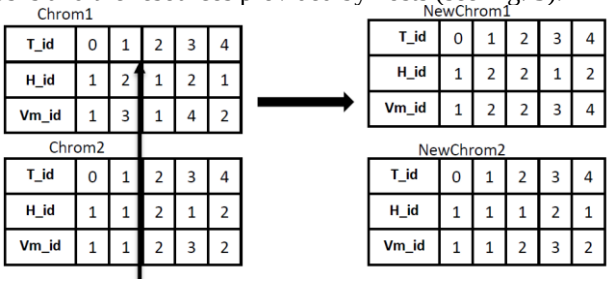

Figure 3. One point crossover operator

B.5 MUTATION PROCESS

After crossover, mutation process takes place to prevent the population of individuals from changing into the same as one other. It occurs during evolution according to mutation probability. Mutation operation changes one or more gene values in the individual from its initial state. Mutation operation is applied on the second row of the matrix (id of host). This can produce the entirely new id of host. With this new host id, the genetic algorithm may be able to produce a better solution than it was previously. Then a random number is generated for the third row (id of VM). After that the properties of VM are determined based on the required resources for executing tasks and the resources provided by hosts (see Fig. 4.)

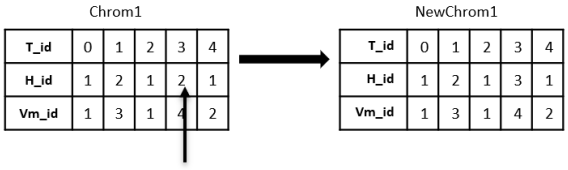

Figure 4. Mutation operator

V. SIMULATION

Evaluating the performance of cloud provisioning policies, application workload models, and resources performance models in a repeatable manner under varying system and user configurations and requirements is difficult to achieve. To overcome this challenge simulation tools are proposed. Simulation tools are especially important for cloud computing research because many clouds are also still in development. CloudSim [23] models and simulates cloud computing environments and supports multiple VMs within a datacenter node.

In this section, the performance of the proposed algorithm, *GTSwDVG*, is compared with four algorithms: SJF, which uses the shortest job first criterion, LCFP [9], which is based on the Longest Cloudlet Fastest Processing Element (LCFP), HSLJF [16], which combines shortest job first and longest job first criteria and CSSA [14], which uses one of the strategies of bee families called the social spider strategy.

In the reminder of this section, the simulation parameters, performance criteria, and results will be presented and discussed.

Table 1. Simulation Parameter

Number of datacenters	1
Number of cloud hosts	2
Host MIPS 1	100000
Number of CPUs per host	6
Host memory	16 GB
Host storage	1 TB
Host bandwidth	100 GB/s
Number of tasks	500,1000, 1500,2000
Number of Millions of instructions per task	1000-10000
Number of required cores per task	1-6
Number of Virtual machines	16, 32, 64
Number of CPUs per Virtual machine	1-6
Virtual machine MIPS	1000, 2000
Virtual machine size	10 GB
Virtual machine memory	0.5 GB
VM Policy	Time Shared

A. SIMULATION PARAMETERS AND PERFORMANCE CRITERIA

The parameters setting of simulation environment is described in Table 1.

B. SIMULATION RESULTS AND ANALYSIS

The performance metrics such as response time, makespan, FlowTime, throughput, resource utilization and trad-off FlowTime and makespan compared with existing works are used to analyze and evaluate the performance of the proposed algorithm, *GTSwDVG* to obtain the objective of a user or the objective of a cloud vendor. These metrics are described in the next subsections.

B.1 RESPONSE TIME (RT)

Response time is the taken time by a task to start responding (the time from the submission time of a task *sb* until the first response is produced). Response time metric gives the total time needed to receive the first response from a cloud computing system and this metric should be minimized. The response time (RT) of cloud computing system is calculated as follows [16]:

$$RT = min\left(ct_{is,x}^{jk} - sb_{is}\right), \tag{9}$$

where $1 \le i \le n$, $1 \le s \le m$, $1 \le j \le l$, $1 \le k \le y$, $1 \le x \le z$.

Figs. 5, 6, and 7 show the comparisons of the proposed algorithm with SJF, LCFP [9], HSLJF [16] and CSSA [14] algorithms for different sets of cloudlets and VMs. The response time obtained by the proposed algorithm is less than that of other algorithms. That is clear that the proposed algorithm is better than other works and it can minimize the response time, efficiently.

B.2 FLOWTIME (FT)

FlowTime (*FT*) is the total sum of completion time of all tasks (also called schedule length) and is calculated as follows:

$$FT = \sum_{i=1}^{n} \sum_{s=1}^{m} ct_{is,x}^{jk}, \tag{10}$$

where $1 \le j \le l$, $1 \le k \le y$, $1 \le x \le z$. Minimize FlowTime (minimizing the total completion time for all tasks) means that all VMs have finished tasks execution earlier to achieve the goal of the user, so this metric should be minimized.

Figs. 8, 9, and 10 show the comparisons of the proposed algorithm with SJF, LCFP [9], HSLJF [16] and CSSA [14] algorithms for different sets of cloudlets and VMs. The FlowTime obtained by the proposed algorithm is less than that of other algorithms. That is clear that the proposed algorithm can minimize the FlowTime.

B.3 MAKESPAN

Makespan *(mk)* metric is calculated in Eq. (5). It is the important parameter that measures the quality of the results obtained by any scheduling algorithm for minimizing the

time consumed for finishing execution time of all tasks to satisfy the objective of the user and objective of the cloud vendors. So, makeSpan metric should be minimized. The main objective of the work is to minimize makeSpan for fast execution of tasks.

Figs. 11, 12, and 13 show the makespan values of the proposed *GTSwDVG*, SJF, LCFP [9], HSLJF [16] and CSSA [14] algorithms for different sets of cloudlets and VMs. At the most test cases, the proposed *GTSwDVG* finds the solutions with lower makespan than all other Algorithms except the HSLJF. This is because, in some test cases (1500 and 2000 tasks), HSLJF uses 64 VMs with core speed equal to 2000MIPS and the result of makespan is better than of the proposed *GTSwDVG*. However, there is problem, if there are three tasks from three different users $\{t_1, t_2, t_3\}$ with a number of instructions {300,2000,1000}, respectively. Let there be 3 VMs: vm_1,vm_2, and vm_3 and one core for each where the core speeds are {200,100,50}, respectively. In case of distributing t_1 on vm_2, t_2 on vm_3, and t_3 on vm_1 the execution times are 3, 40, and 5, makespan is 40, and FlowTime is 48. While, in case of distributing t_1 on vm_1, t_2 on vm_2, and t_3 on vm_3, the execution times are 15, 20, and 20, the makespan is 20, and FlowTime is 55. As a result, in the second case the makespan is minimized, the FlowTime is maximized, and the execution times of t_1 and t_3 are maximized. This means that if the objective of the users is minimizing the execution time, there is a problem to users 1 and 3. In addition, the result of makespane using HSLJF for 64 VMs with core speed equals 2000MIPS is better than that of the proposed algorithm for executing 1500 and 2000 tasks, while the FlowTime of the proposed algorithm is better than when using HSLJF.

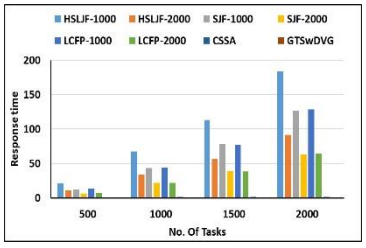

Figure 5. Comparison of response time using 16 VMs

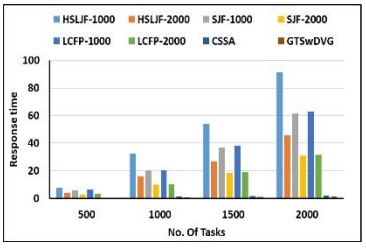

Figure 6. Comparison of response time using 32 VMs

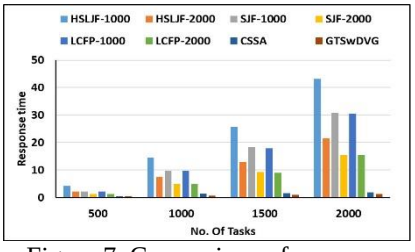

Figure 7. Comparison of response time using 64 VMs

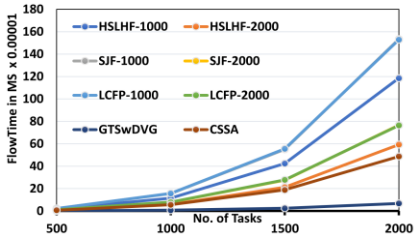

Figure 8. Comparison of FlowTime using 16 VMs

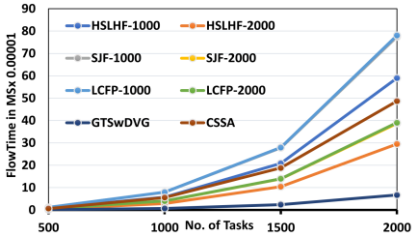

Figure 9. Comparison of FlowTime using 32 VMs

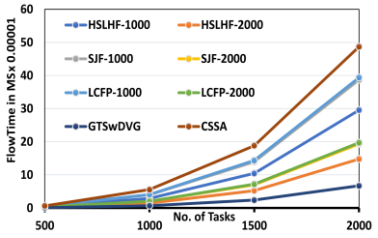

Figure 10. Comparison of FlowTime using 64 VMs

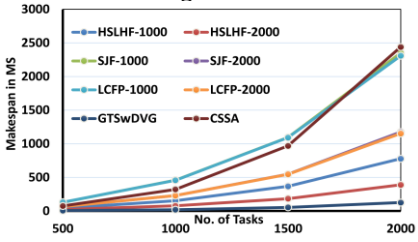

Figure 11. Comparison of makespan using 16 VMs

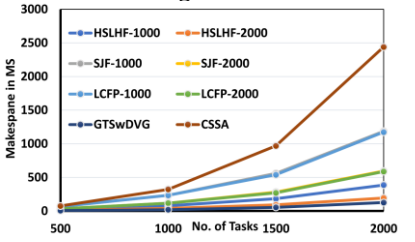

Figure 12. Comparison of makespan using 32 VMs

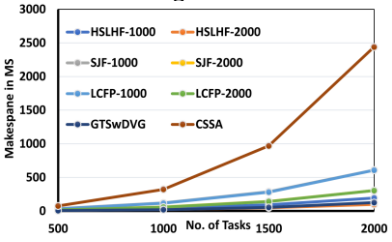

Figure 13. Comparison of makespan using 64 VMs

B.4 TRADE-OFF BETWEEN MAKESPAN AND FLOWTIME

Trade-off between makespan and FlowTime metric is denoted as *TMF* and is calculated as follows:

$$TMF = \frac{A}{B} \tag{11}$$

where

$$A = min(\,mk) * min(\,FlowTime) \tag{12}$$

$$B = mk * FlowTime, \tag{13}$$

for each algorithm. It is the important parameter to measure the user satisfaction and the objective of cloud vendors. This

metric solves the problem, which was described above in makespan metric. The better value of algorithm is closer to 1.

Figs. 14, 15, and 16 show the TMF of the proposed algorithm, SJF, LCFP [9], HSLJF [16] and CSSA [14] algorithms. It is clear that the proposed algorithm achieves TMF values higher than other algorithms. This is because, the proposed algorithm takes into account the tradeoff among makespan and flowtime metrics.

B.5 THROUGHPUT (TH)

The throughput *(TH)* is the maximum rate of executing the tasks in a time unit (i.e., is the total number of tasks, whose execution has been finished successfully per a time unit) and is calculated as follows [16].

$$TH = \frac{m}{mk}, \tag{14}$$

where m is the number of tasks and it should be maximized for giving the improved performance of the system. The best scheduling algorithm should maximize the throughput of the system. It is used to measure the performance of scheduling algorithm. Figs. 17, 18, and 19 show the throughput of the proposed algorithm, SJF, LCFP [9], HSLJF [16] and CSSA [14] algorithms. It is clear that the proposed algorithm is better than other algorithms.

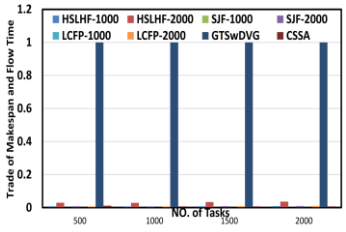

Figure 14. Comparison of trade of makespan and FlowTime using 16 VMs

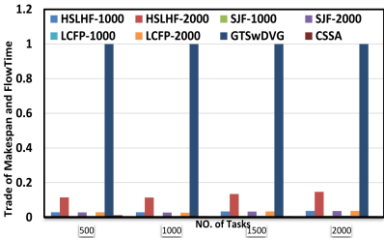

Figure 15. Comparison of trade of makespan and FlowTime using 32 VMs

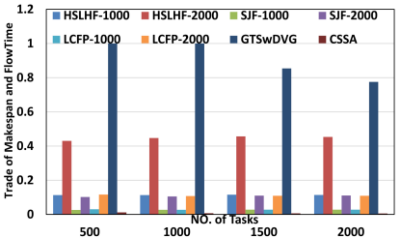

Figure 16. Comparison of trade of makespan and FlowTime using 64 VMs

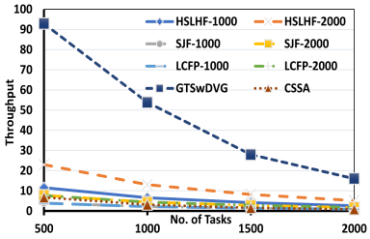

Figure 17. Comparison of throughput using 16 VMs

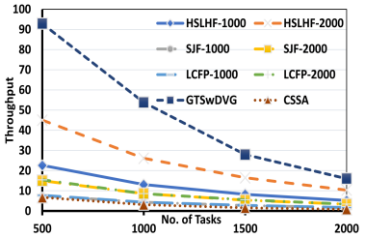

Figure 18. Comparison of throughput using 32 VMs

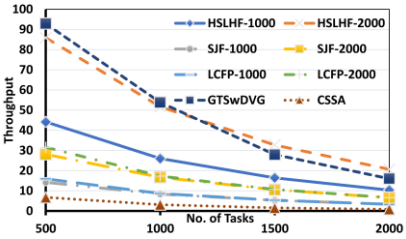

Figure 19. Comparison of throughput using VMs

B.6 RESOURCE UTILIZATION (RU)

Resource utilization *(RU)* is one of the objectives of cloud providers. The scheduling technique should improve the system performance and takes resource utilization into consideration. Resource utilization is calculated as follows [16]:

$$RU = \frac{\sum_{k=1}^{y} mk_k}{y * (Max(mk_k))}, \tag{15}$$

where $1 \leq k \leq y$ and y is the number of hosts and mk_k is the makespan of all tasks that are executed on host k.

The cloud service providers want to earn maximum profit by reducing the amount of resources in use.

Fig. 20, 21, and 22 show that the proposed algorithm can achieve resource utilization up to 95%. This means that the proposed algorithm can improve the system performance without loss of efficient resource utilization.

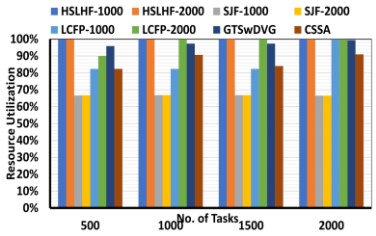

Figure 20. Comparison of resource utilization using 16 VMs.

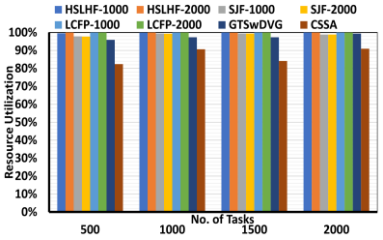

Figure 21. Comparison of resource utilization using 32 VMs.

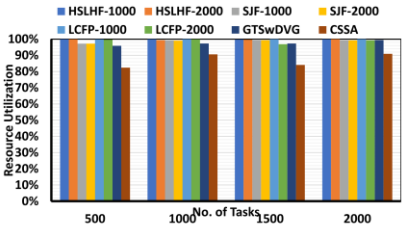

Figure 22. Comparison of resource utilization using 64 VMs

VI. CONCLUSION

In this paper, a new efficient scheduling mechanism was developed for solving the scheduling problem in cloud computing environment. The proposed algorithm is extended to add possibility dynamic number and characteristic of VMs. The number of VMs is related to the number of the user tasks and the properties of VMs based on the resources that are required for the user tasks and resources provided by hosts. The new algorithm is based on Genetic Algorithm (GA). The results obtained by the proposed algorithm have shown that it achieves better performance in terms of response time, makespan, FlowTime, throughput, and trade-off FlowTime without loss of efficient resource utilization than some of the existing algorithms. In future work, the task priorities and types (dependent) in our optimization model will be considered. In addition, the proposed model will be implemented in a real cloud environment. Also, the performance of proposed algorithm will be compared with more advanced variants of GA, for example, island models, other strategies of bee families, or other models that use not only mutation and crossover of chromosomes but additional operation of chromosomes inversion.

References

[1] W. T. Tsai, X. Sun, and J. Balasooriya, "Service oriented cloud computing architecture," *Proceedings of the Seventh International Conference on Information Technology: New Generations*, Las Vegas, NV, USA, April 12-14, 2010, pp. 684-689, https://doi.org/10.1109/ITNG.2010.214.

[2] B. Furht, A. Escalante, *Handbook of cloud computing*, Springer, 2010, 655 p. https://doi.org/10.1007/978-1-4419-6524-0.

[3] D. Sullivan, "The definitive guide to cloud computing," *Real Time Nexus*, pp. 4-11, 2010.

[4] G. Soni and M. Kalra, "A novel approach for load balancing in cloud data center," *Proceedings of the International Advance Computing Conference (IACC)*, Gurgaon, India, Feb 21-22, 2014, pp. 807-812, https://doi.org/10.1109/IAdCC.2014.6779427.

[5] Y. Jadeja and K. Modi. "Cloud computing – concepts, architecture and challenges," *Proceedings of the International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, Kumaracoil, India, March 21-22, 2012, pp. 877-880, https://doi.org/10.1109/ICCEET.2012.6203873.

[6] S. H. H. Madni, M. S. Abd Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Resource scheduling for infrastructure as a service (IAAS) in cloud computing: Challenges and opportunities," *Journal of Network and Computer Applications*, vol. 68, pp. 173-200, 2016, https://doi.org/10.1016/j.jnca.2016.04.016.

[7] B. A. Hridita, M. Irfan, and M. S. Islam, "Task allocation for mobile cloud computing: State-of-the art and open challenges," *Proceedings of the 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, Dhaka, Bangladesh, May 13-14, 2016, pp. 752–757, https://doi.org/10.1109/ICIEV.2016.7760102.

[8] H. Shoja, H. Nahid, and R. Azizi, "A comparative survey on load balancing algorithms in cloud computing," *Proceedings of the Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, Hefei, China, July1-13, 2014, pp. 1–5, https://doi.org/10.1109/ICCCNT.2014.6963138.

[9] S. Sindhu and S. Mukherjee, "Efficient task scheduling algorithms for cloud computing environment," *Proceedings of the International Conference on High Performance Architecture and Grid Computing*, Chandigarh, India, July 19-20, 2011, pp. 79–83, https://doi.org/10.1007/978-3-642-22577-2_11.

[10] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent tasks scheduling based on genetic algorithm in cloud computing," *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, Beijing, China, September 24-26, 2009, pp.1–4, https://doi.org/10.1109/WICOM.2009.5301850.

[11] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, vol. 16, issue 3, pp. 275–295, 2015, https://doi.org/10.1016/j.eij.2015.07.001.

[12] R.K. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework," *Proceedings of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)*, 2015, pp. 1219–1227, https://doi.org/10.1016/j.procs.2015.07.419.

[13] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," *Proceedings of the 8th International Conference on Computer Engineering Systems (ICCES)*, Cairo, Egypt, November 26-28, 2013, pp. 64–69, https://doi.org/10.1109/ICCES.2013.6707172.

[14] V. M. Arul Xavier and S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing," *Cluster Computing*, vol. 22, issue 1, pp. 287-297, 2018, https://doi.org/10.1007/s10586-018-1823-x.

[15] K. Naik, G. Meera Gandhi, S. H. Patil, "Multiobjective virtual machine selection for task scheduling in cloud computing," in: N. K. Verma, A. K. Ghosh (Eds), *Computational Intelligence: Theories, Applications and Future Directions*, Springer Singapore, Singapore, 2019, pp. 319-331, https://doi.org/10.1007/978-981-13-1132-1_25.

[16] M. A. Alworafi, A. Dhari, S. A. ElBooz, A. A. Nasr, A. Arpitha, S. Mallappa, "An enhanced task scheduling in cloud computing based on hybrid approach," in: P. Nagabhushan, D. S. Guru, B. H. Shekar, Y. H. Sharath Kumar (Eds), *Data Analytics and Learning*, Springer Singapore, Singapore, 2019, pp. 11–25, https://doi.org/10.1007/978-981-13-2514-4_2.

[17] J. Gu, J. Hu, T. Zhao, G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment," *Journal of Computers*, vol. 7, issue 1, pp. 42–52, 2012, https://doi.org/10.4304/jcp.7.1.42-52.

[18] M. A. Tawfeek, A. B. El-Sisi, A. E. Keshk, and F. A. Torkey, "Virtual machine placement based on ant colony optimization for minimizing resource wastage," *Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications*, Cairo, Egypt, March 28-30, 2014, pp. 153–164, https://doi.org/10.1007/978-3-319-13461-1_16.

[19] S. K. Sonkar and M. U. Kharat, "A review on resource allocation and vm scheduling techniques and a model for efficient resource

management in cloud computing environment," *Proceedings of the International Conference on ICT in Business Industry Government (ICTBIG)*, Indore, India, November 18-19, 2016, pp. 1–7, https://doi.org/10.1109/ICTBIG.2016.7892646.

[20] B. Pavithra and R. Ranjana, "A comparative study on performance of energy efficient load balancing techniques in cloud," *Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET*), Chennai, India, March 23-25, 2016, pp. 1192-1196, https://doi.org/10.1109/WiSPNET.2016.7566325.

[21] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in: F.-L. Wang, Z. Gong, X. Luo, and J. Lei (Eds), *Web Information Systems and Mining*, Springer Berlin Heidelberg, 2010, pp. 271–277, https://doi.org/10.1007/978-3-642-16515-3_34.

[22] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, A. El-Sayed, "Using the tsp solution strategy for cloudlet scheduling in cloud computing," *Journal of Network and Systems Management*, vol. 27, issue 2, pp. 366–387, 2019, https://doi.org/10.1007/s10922-018-9469-9.

[23] A. Beloglazov, C. A. F. De Rose, R. Buyya, R. N. Calheiros, R. Ranjan, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, issue 1, pp. 23–50, 2011, https://doi.org/10.1002/spe.995.

ASAAD AHMED (AHMED A.A. GAD-ELRAB) *is an Assoc. Prof. and he received his BS. Degree in Computer Science, from Faculty of Science, Alexandria University, Egypt in 1999. He received his MS. Degree in Computer Science from Faculty of Science, Cairo University, Egypt in 2008. He received his Ph.D. from Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), a national corporation university located in NARA, Japan in 2012. He is an associate professor of computer science at the Department of Mathematics, Faculty of Science, Al-Azhar University, Cairo, Egypt. Currently, he is a Consultant for Vice President of Development, King Abdul-Aziz University (KAU), Jeddah, Saudi Arabia. His research interests include cloud computing, mobile computing, Internet of Things applications, smart home, data science, sensor networks, dynamic distributed systems, big data, and mobile crowd sensing.*

TAMER A. ALZOHAIRY *received the B. Sc. Degree in computer science from faculty of Science, Ain Shams University, Cairo, Egypt in 1989. M.Sc degree in Computer science from Math. Dept., Faculty of Science, Al-Menoufia University, Egypt, in 1997 and Ph.D in Computer Science from Math. Dept. Faculty of Science, Suez Canal University, Egypt in 2003. He is working currently as an Associate Professor in Computer science Dept., Science college, Al Azhar University, Cairo, Egypt. Main work is on neural networks, deep learning, pattern recognition and image processing.*

KAMAL R RASLAN *received the M.Sc. and Ph.D. degrees from the Faculty of Science, Menoufia University and Al-Azhar University, Egypt, in 1996 and 1999, respectively. He is currently full Professor of Mathematics with the Faculty of Science, Al-Azhar University, Egypt. He has authored/coauthored over 114 scientific papers in International Journals. His research interests include Numerical Analysis, Finite Difference Methods, Finite Element Methods, Approximation Theory, and Computational Mathematic.*

ALI FAROUK EMARA (FAROUK A. EMARA) *received B.S. Degree in pure math and Computer Science, from the Department of Mathematics, Faculty of Science, Al-Azhar University – Cairo, Egypt in 2008. He received the master's degree in quality of service management in mobile cloud computing from the Department of Mathematics, Faculty of Science, Al-Azhar University – Cairo, Egypt, in 2016. He is active in research, cloud computing and mobile cloud computing. In 2016 he has started to study at the Department of Mathematics, Faculty of Science, Al-Azhar University – Cairo, Egypt PhD degree with topic of efficient schemes for data and resources management in mobile cloud computing.*