



## Article

# Alts: An Adaptive Load Balanced Task Scheduling Approach for Cloud Computing

Aroosa Mubeen <sup>1,†</sup>, Muhammad Ibrahim <sup>2,3,\*,†</sup> , Nargis Bibi <sup>1</sup> , Mohammed Baz <sup>4</sup> and Habib Hamam <sup>5,6</sup> and Omar Cheikhrouhou <sup>7</sup> 

<sup>1</sup> Department of Computer Science, Fatima Jinnah Women University, Rawalpindi 46000, Pakistan; aroosamubeen936@yahoo.com (A.M.); nargis@fjwu.edu.pk (N.B.)

<sup>2</sup> Department of Information Technology, University of Haripur, Haripur 22621, Pakistan

<sup>3</sup> Big Data Research Center, Jeju National University, Jeju 63243, Korea

<sup>4</sup> Department of Computer Engineering, College of Computer and Information Technology, Taif University, P.O. Box. 11099, Taif 21994, Saudi Arabia; mo.baz@tu.edu.sa

<sup>5</sup> Faculty of Engineering, Université de Moncton, Edmundston, NB E1A3E9, Canada; habib.hamam@umoncton.ca

<sup>6</sup> School of Elect. Eng. and Electronic Eng., University of Johannesburg, Auckland Park, Johannesburg 2092, South Africa

<sup>7</sup> CES Laboratory, National School of Engineers of Sfax, University of Sfax, Sfax 3038, Tunisia; o.cheikhrouhou20@gmail.com

\* Correspondence: ibrahimmayar@jejunu.ac.kr

† These authors contributed equally to this work.



**Citation:** Mubeen, A.; Ibrahim, M.; Bibi, N.; Baz, M.; Hamam, H.; Cheikhrouhou, O. Alts: An Adaptive Load Balanced Task Scheduling Approach for Cloud Computing. *Processes* **2021**, *9*, 1514. <https://doi.org/10.3390/pr9091514>

Academic Editor: Mengchu Zhou

Received: 13 July 2021

Accepted: 21 August 2021

Published: 26 August 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Abstract:** According to the research, many task scheduling approaches have been proposed like GA, ACO, etc., which have improved the performance of the cloud data centers concerning various scheduling parameters. The task scheduling problem is NP-hard, as the key reason is the number of solutions/combinations grows exponentially with the problem size, e.g., the number of tasks and the number of computing resources. Thus, it is always challenging to have complete optimal scheduling of the user tasks. In this research, we proposed an adaptive load-balanced task scheduling (ALTS) approach for cloud computing. The proposed task scheduling algorithm maps all incoming tasks to the available VMs in a load-balanced way to reduce the makespan, maximize resource utilization, and adaptively minimize the SLA violation. The performance of the proposed task scheduling algorithm is evaluated and compared with the state-of-the-art task scheduling ACO, GA, and GAACO approaches concerning average resource utilization (ARUR), Makespan, and SLA violation. The proposed approach has revealed significant improvements concerning the makespan, SLA violation, and resource utilization against the compared approaches.

**Keywords:** task scheduling; CloudSim; cloud computing; SLA; resource utilization

## 1. Introduction

Cloud computing provides dynamic virtualized resources and services over the web in a new paradigm [1,2]. By the configuration of basic hardware and software, it authorizes access to a shared pool of resources across the cloud data centers. In cloud computing, there are a huge number of user requests available for their execution because of the heterogeneous environment. The user first submits its required resource request as an input with the help of a broker in IaaS. In some cases, the user may have some specific requirements that need some form of agreement where the cloud service providers have to provide guaranteed services to the users also called service level agreement (SLA) [3]. The user requests are usually called tasks in the context of cloud computing. To execute the tasks on the available cloud datacenter (CDC) machines/VMs, various task scheduling approaches are employed [4–7]. The main purpose of the task scheduling algorithm is to focus on different factors related to quality of service (QoS) like throughput, response time,

etc. [8–10]. Based on the task requirements, suitable resources [11–13] are allocated by the cloud environment. The user's specific requirements includes resources related to time, memory, operating system, etc. For example, if a user wants to demand 3D rendering with the help of reservation-based cloud, then the user must need to mention GPU and CPU time, start time, and the longest rendering time to the IaaS provider. After receiving such kinds of user requests, the scheduling approach finds a VM through mapping based on the available computing resources. When these task scheduling algorithms perform their function improperly, it will increase the makespan time and decrease the throughput of the whole cloud computing environment [14]. A problem related to the imbalanced mapping of tasks to the VMs can be defined as, for example, if there are 12 tasks and 4 VMs available in the Cloud computing environment [15]. The usage of resource utilization rate of each task is defined as T1 = 53%, T2 = 77%, T3 = 65%, T4 = 43%, T5 = 64%, T6 = 84%, T7 = 32%, T8 = 51%, T9 = 99%, T10 = 85%, T11 = 71%, T12 = 25%. Arrange the given ratios of tasks in ascending order. After ordering the first three tasks T9, T10, and T6 are assigned to VM1, T2, T11, and T3 are assigned to VM2, T5, T1, and T8 are assigned to VM3, and lastly, T4, T7, and T12 are assigned to VM4. Now, after finding their average resource utilization value, the usage of each VM can be defined as VM3 = 100%, VM2 = 75%, VM4 = 51%, VM1 = 50%. This mapping clearly shows that VM3 is completely utilized but VM2, VM4, and VM1 are under-utilized. So there is a need for a proper task scheduling algorithm that can utilize the resources of all VMs in a load-balanced way [16,17].

The task scheduling approaches can increase the task execution time and lessen the throughput of the entire cloud system. In this regard, cloud computing aspires to improve the overall performance and to make better use of these computing resources in a heterogeneous environment [18].

Several task scheduling algorithms are used in cloud computing environment like ant colony optimization algorithm (ACO) [19–21], particle swarm optimization algorithm (PSO) [22–25] and genetic algorithm (GA) [26–35]. In this work, we have integrated the GA approach with the ACO approach in a hybrid way to schedule the tasks in a load balanced way. GA has huge search space and does not provide a fixed length of solution to a particular problem. ACO algorithm performs various numbers of iterations and at the end of each iteration the ant's path is updated because path traversing information is stored in its memory in each iteration. For a variety of optimization problems, ACO is used for task scheduling. ACO provides a fast convergence rate that obtains a good solution at the end. Due to concurrency and expandability, the ACO algorithmic approach is quite suitable for task scheduling in the cloud computing environment. After each iteration, the traverse path is stored in ACO memory. To obtain a globally optimal solution, the ACO algorithm assists GA, and then the GA algorithm helps to supply the initial pheromone to the ACO task scheduling algorithm.

The rest of the paper is organized as follows: Section 2 provides details regarding various task scheduling algorithms for Cloud computing, as well as a discussion related to the ACO and GA. Moreover, a comparison of the different task scheduling algorithms is presented at the end of Section 2. Section 3.1 delineates the contribution of this work (i.e., Adaptive Load Balancing Task Scheduling Algorithm (ALTS)). Section 4 presents the experimental setup details for the cloud computing task scheduling environment and results and discussion of these results. The conclusion of the proposed research is presented in Section 5 with the future perspective of task scheduling in cloud computing.

## 2. Related Work

Cloud computing environment provides ubiquitous access to resources and services to its customers on a pay as you use model. Different types of cloud computing models exists, which are IaaS, PaaS, and SaaS [36]. In recent years, several task scheduling approaches have been proposed for Cloud computing. These task scheduling algorithms can be classified to heuristic-based task scheduling algorithms and meta-heuristic-based task scheduling algorithms. The heuristic-based task scheduling algorithms include the round-

robin (RR), first come first serve (FCFS) [37], priority-based task scheduling algorithms [38], etc. Meta-heuristic-based task scheduling algorithms include ACO [19–21], PSO [22–25], and GA [26–32], etc. These algorithms generally optimize one or more parameters concerning the task scheduling in the cloud environment.

Several scheduling algorithms were studied in-depth in [39], where methods, applications, and parameters of different task scheduling algorithms that include ACO, PSO, and GA, multi-processor-based scheduling, and clustering-based scheduling are discussed. Moreover, the comparison between different scheduling algorithms shows that these algorithms are suitable for a small number of tasks, but if the task number increases, these algorithms fail to provide efficient results. The authors in [40] presented a novel cloud task scheduling approach that utilizes the idle time slot-aware rules and particle swarm optimization (PSO) to meet the deadline constraints of delay sensitive applications. The paper proposed a novel particle encoding scheme to encode the VM type needed by each of the task and schedules the tasks accordingly. The experimental results prove the effectiveness of the proposed approach concerning the execution time and meeting the deadline of the tasks.

To assign sub-tasks to the optimal resources [41], the authors introduce static task scheduling algorithms that combine heuristic-based HEFT search with genetic algorithm (GA). The results of the proposed algorithm show much improvement in the makespan. A task scheduling algorithm based on GA is introduced in [28] to reduce the makespan for users' cloudlets. The computational complexity of different tasks is considered in this research. In order to provide the answer to the problem of convergence in ACO, an adaptive task scheduling algorithm is proposed in [20]. The authors of this research [17], by improving the load balancing feature of different cloudlets in the cloud. This technical load balancing algorithm helps to lessen the response time and makespan by the use of the ant colony optimization algorithm (ACO). Authors also proposed a new hybrid task scheduling algorithm that optimizes four parameters, namely security, reliability, time and cost. The authors in [41] presented an improved genetic based task scheduling approach to optimize the task scheduling solutions across the cloud computing. The proposed approach harnesses both the advantages of evolutionary algorithm and heuristic methods. The proposed approach has shown best Makespan performance against the other compared contemporary approaches. The authors in [27] introduced a (MO-GA) algorithm for task scheduling in Cloud computing to improve the energy efficiency. Another work for scheduling the tasks and for energy-aware task scheduling in Cloud computing, an Energy-Aware multi-objective chiropteran algorithm (EAMOCA) is proposed in [42] to reduce execution time and to boost the ARUR. The authors in [43] proposed an improved differential evolution (IDE) model to improve the utilization of the VMs with low makespan. The proposed approach takes batch of user tasks as an input and are assigned to the VMs for optimal VMs utilization and minimizing the overall makespan.

In [44], the authors proposed SLA aware task scheduling approach. The proposed model considers time-varying performance with an aim to minimize the execution on the clouds considering the infrastructure as a service model. In the proposed approach, the predicted performance of VMs is provided as an input to a genetic algorithm that generates schedules at run-time.

To maximize the energy efficiency (profit) and minimize the energy consumption of cloud datacenters, the authors proposed a novel scheduling approach in [45]. To solve the scheduling problem, this paper presents an approach for tasks to virtual machine allocation that achieves maximum profit for reserved services in cloud data centers. By the use of a traditional cloud simulator that involves a time-consuming process, this approach gives accurate estimation for energy consumption that simulates all the virtual machine allocation updates. For this, the researchers have designed a simulation engine for CloudSim. This approach combines more virtual machines with a few physical machines and obtained energy savings from 24% to 41%. Furthermore, these techniques can handle many user requests that arrive in cloud data centers. An efficient task scheduling multi-objective task

scheduling algorithm has been proposed in [24] that minimizes the execution and waiting time of different cloudlets on the basis of ranking technique among VMs. Another objective of this approach is optimizing the throughput for cloud data centers that reduce the overall execution time is implemented in [10].

The authors in [22] proposed an improved particle swarm optimization (IPSO). The IPSO algorithm achieves 50% efficiency in terms of makespan time. The authors in [9] proposed an intelligent bio-inspired task scheduling algorithm for large IoT infrastructure, by the combination of two popular task scheduling algorithms, i.e., ACO and GA for multiprocessor heterogeneous environment. The proposed approach lead to reduced execution time of the different length of tasks, as the number of processors increases and the results are analyzed with state of the art approaches. Authors combined two existing algorithms in order to optimize the parameters of cost and time [29].

For the purpose of the task scheduling, authors [46] combined the cuckoo algorithm (CA) and ACO. In order to provide an efficient hybrid algorithm, the researchers combined the best features of the cuckoo algorithm (CA) and ACO that outperforms the algorithm when these two algorithms are applied individually. However, in this research, the cuckoo algorithm (CA) is suitable to provide accurate results within its local minima, not for global minima. The categorization represented in Table 1 is based on the scheduling applications that have been taken from recent research papers.

**Table 1.** Comparison of task scheduling algorithms.

Task Scheduling Algorithms	Cost	ARUR	Makespan	Energy Consumption
IPSO [22]	✓	✓	✓	
ICGA (Farnoosh et al., 2019) [32]		✓		✓
PSO (Near et al., 2017) [23]		✓	✓	
LBACO (Selvakumar et al., 2019) [7]		✓	✓	✓
ICDFS (Shadi et al., 2018) [47]		✓	✓	
HC (Keshanchiet al., 2017) [41]	✓		✓	✓
ACO (Sayantani et al., 2018) [31]		✓	✓	✓
GAACO (Sayantani et al., 2018) [9]	✓	✓	✓	✓

Different scheduling applications such as a job or task scheduling, dynamic or static scheduling, and task scheduling for cloud environment or grid environment are considered in this research.

The authors in [19] introduced a multi-objective optimization scheduling approach that employs makespan and user's budget costs to improve the response time with a reduced cost. The proposed model utilize two constraint functions to evaluate and provide response concerning the budget and performance. For the proper distribution of resources to lessen cost, makespan, energy consumption, a study is performed for multi-objective workflow scheduling [12]. For heterogeneous cloud computing environments, parameters like average cloud utilization, execution cost, and makespan are explained in [18]. A multi-objective algorithm is proposed in [3], which used backfilling heuristic for the mapping of resources from large scale to medium-scale low-level to minimize energy

consumption, makespan, and several cloudlets that violate the service level agreement (SLA) requirements.

In a multi cloud system, an efficient multi-objective cloudlet scheduling algorithm is proposed by authors in [18], which is used to reduce the makespan and cost of Cloud computing. This algorithm increases performance in terms of ARUR. A dynamic task scheduling framework is designed in [47] that is based on chicken swarm and improved raven roosting optimization methods in cloud computing. In cloud computing, a new genetic approach is presented by [28] that solves the issues related to task scheduling and provides the best optimal results. In fog computing systems, a load-balanced hill climbing algorithm is designed by the authors of [48]. This algorithm has a temporary storage area, that is secure and easily accessible by cloud consumers. The authors in [49] proposed a trust aware task scheduling approach known as matching and multi-round allocation (MMA) for optimizing the makespan and the associated cost for each task. The paper also deals with the security constraints concerning the tasks that works in two phases for executing the scheduling of tasks. In first phase the tasks requirements are find out such as security, performance and reliability in the multi-cloud environment; in second phase the tasks are assigned to the resources based on the task requirements to reduce the execution time and associated cost. In [50], the authors proposed a novel task scheduling strategy known as two-stage to improve the resource utilization and load balancing of resources across the cloud environment. For this, they employed Bayes classifier that classify the tasks based on the historical scheduling data. In the next step, VMs are created in an optimal way to perform scheduling of the required tasks. In [51], Yuan et al. presented PSO based cost minimization problem for task scheduling in Cloud computing. The proposed approach considers the delay of the task and schedule the tasks based on the delay requirements of the tasks. A solution for non-deterministic polynomial time task scheduling algorithm is given by [34] that provides best task allocation strategy based on the parameters of throughput, task allocation and response time. A hybrid combination of priority-based scheduling and shortest job first proposed and implemented in the cloud computing environment [38]. This hybrid algorithm provides promising results in minimizing turnaround time and average waiting time and boost the efficiency of cloud computing resources. To improve the processing power of cloud applications, a novel job scheduling strategy is proposed in [35] to improve the performance and reliability of cloud computing applications that reduces the waiting in queues in an adaptive way. In a static environment, an efficient algorithm [15] is developed for an independent number of batch tasks. This algorithm minimizes the makespan for multi-cloud systems, as it is designed for offline batch mode multi-cloud systems. This algorithm (PIMSTA) provides reliability and scalability to the users with a minimum execution time.

This research work contributes an Adaptive Load Balancing Task Scheduling (ALTS) approach to boost the utilization rate of VMs in cloud and minimizing the makespan. Moreover, the ALTS approach identifies the factors that are responsible for the violation of SLA requirements and develops a model for optimum resource utilization of cloud computing. The next section provides details concerning the proposed ALTS approach.

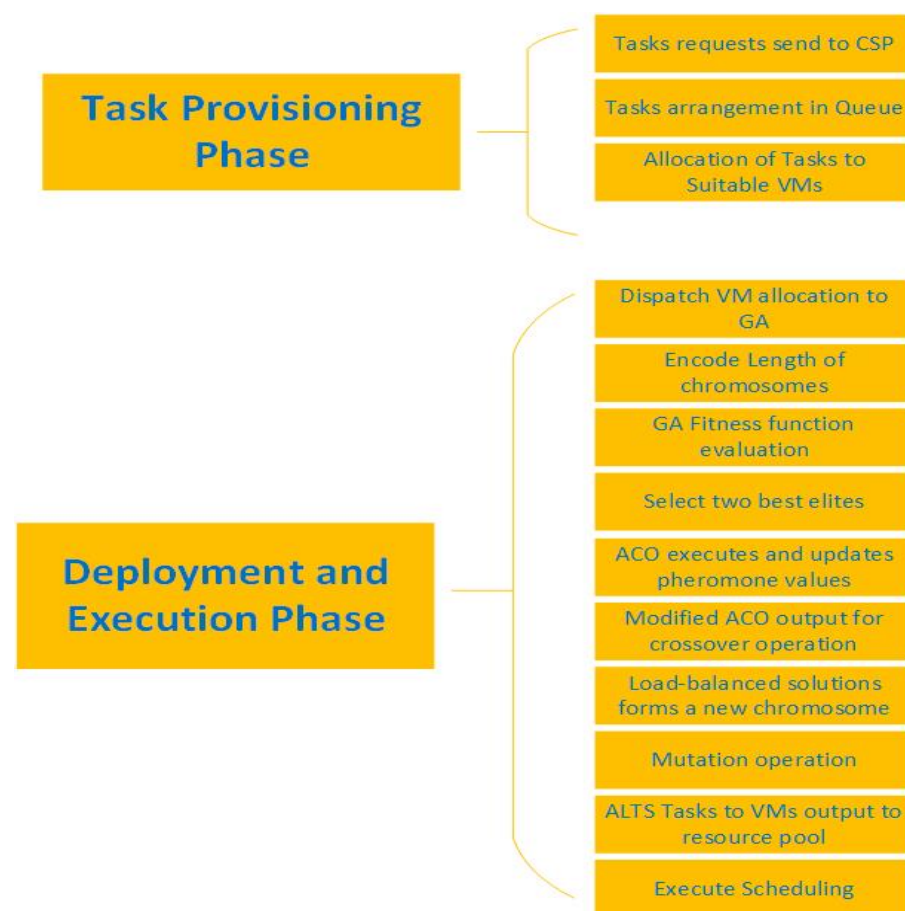
### 3. Adaptive Load Balancing Task Scheduling (ALTS) Approach and Architecture

This section defines the high-level architecture and workflow of the proposed approach. Further details concerning the mathematical model and deep analysis of the ALTS algorithm are also part of this section.

#### 3.1. ALTS Algorithm Workflow

The workflow of the ALTS algorithm represents the automation tasks scheduling in the cloud. To be specific, at every layer of ALTS architecture, the ALTS workflow can be categorized into several segments as revealed in Figure 1. Initially an accessibility phase of the ALTS algorithm deals with user's that interact with the cloud environment. Clients send an appeal to the cloud service provider (CSP) for the provisioning of tasks. The

scheduler performs the scheduling of the tasks in queues on the foundation of memory size and execution time values. Suitable resources (VMs) are allocated to tasks that fulfill their requirements by the scheduler. The deployment and execution phase dispatches the suitable virtual machine list and tasks to the ALTS algorithm. In the ALTS algorithm, initially, the execution of the genetic algorithm (GA) starts by encoding chromosomes. The length of the chromosomes is equal to the number of tasks and gene value represents the number of VMs. The fitness function of the GA is used to evaluate the entire population, which is based on parameters of minimum makespan time, minimum SLA violation and maximum resource utilization rate. In the selection operation of the genetic algorithm, two best chromosomes recognize as two optimal chromosomes ( $n$ th and  $n$ th - 1) are selected. Here, the  $n$ th best chromosome elite is reserved for crossover operation and the second  $n$ th - 1 best chromosome elite is modified by the ACO. In an ant colony optimization algorithm (ACO), each is placed on the starting node and then the state transition rule is applied by all ants and they generate a solution. When all ants build a complete solution, then the pheromone value is updated. The best solution of an ACO is used for crossover operation of a GA. The finest two solutions from the GA and ACO are combined to form a new chromosome, which performs its functionality as the best resource (VM) allocation technique. To uphold the individuality in population, mutation operation is necessary for a GA. Then the resulted resource allocation scheme by GA and ACO algorithm are submitted to the resource pool of virtual machine manager, which execute tasks on suitable virtual machines (VMs) that are handled by the host machine. Simulation parameters are initialized and its execution starts.

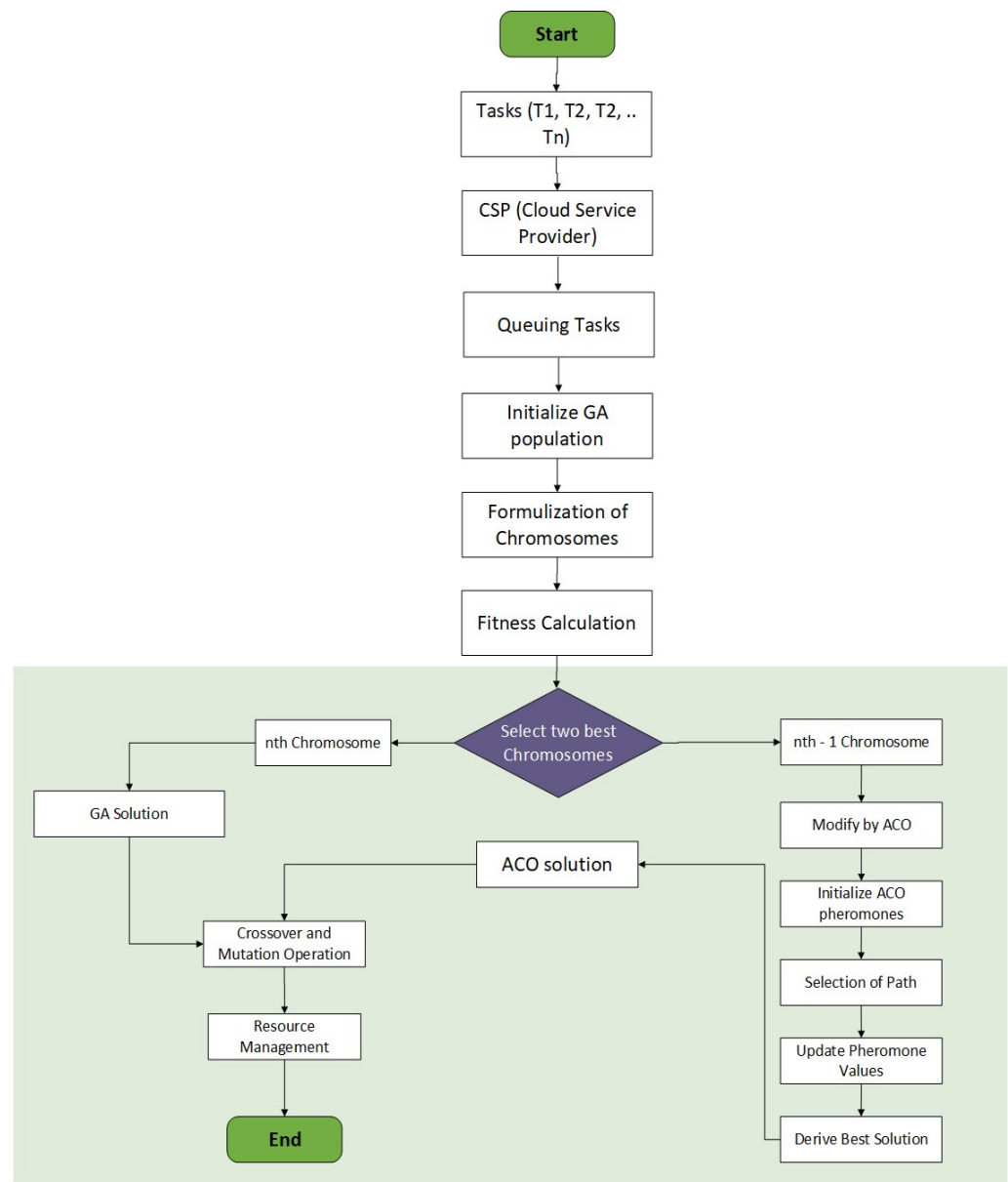


**Figure 1.** ALTS workflow.

### 3.2. Execution of ALTS Approach

The task scheduling problem can be defined as the allocation of a set of  $n$  tasks to a set of  $m$  virtual machines with several aims in mind depending on the objective of

the scheduling problem. In our case, the main focus is to improve the utilization of the resources in the minimum makespan while minimizing the SLA violation. In the proposed adaptive load-balanced task scheduling (ALTS) algorithm, the first initial population is provided as an input to the genetic algorithm (GA). In this case, the GA uses a fitness function to find the best task to VM mapping with an aim to minimize the makespan, SLA violation and boost the throughput and resource utilization in a load-balanced way of the tasks scheduling across the cloud data centers. All the steps of the adaptive load-balanced task scheduling (ALTS) algorithm are demonstrated in Figure 2.



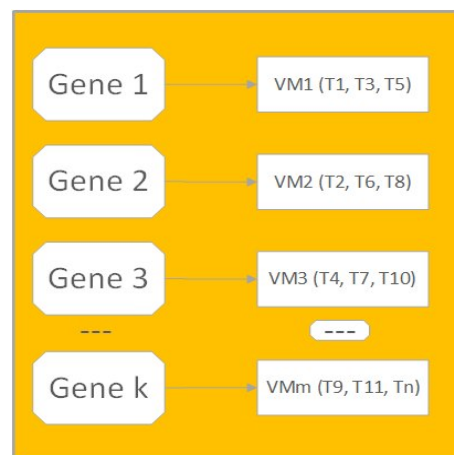
**Figure 2.** Flow chart of proposed ALTS approach.

In the proposed approach the main objective function is utilizing three parameter in order to achieve load balancing. In the proposed approach Equation (1) represents the fitness function, which is comprised of makespan, SLA violation and ARUR (Average resource utilization). The ARUR parameter controls the load balancing of the resources in the run time. The proposed approach employs an adaptive approach to achieve load balancing as well as not to compromise the SLA violation, thus the SLA violation parameter is used. Secondly, the makespan parameter is used to improve the execution time of the

tasks on the available cloud resources. The following are the main aspects of GA and ACO, which are used in the proposed technique of adaptive load balancing task scheduling ALTS algorithm. These are as follows:

### 3.2.1. GA Initial Population and Formation of Chromosomes

It consists of all individuals that find an optimal solution for a genetic algorithm (GA). In the population, every solution is considered as an individual that is represented as chromosomes. These chromosomes are selected on the basis of specific criteria. In a GA, the main requirements of chromosomes are the details of the VMs and the tasks. In the proposed task scheduling (ALTS) algorithm, representation of tasks and virtual machines (VMs) are exemplified in Figure 3.



**Figure 3.** Representation of tasks and VMs.

In cloud computing environment,  $n$  number of tasks are assigned to  $m$  number of available resources. Here, the length of chromosome is represented by  $n$  and the virtual machine (VM) number is represented by gene value that ranges between 1 and  $m$ .

### 3.2.2. Calculation of Fitness Function

Individual chromosomes are selected, mainly dependent on the selection criteria of fitness function. In the population, fitness function shows the performance of every individual. Therefore the motivating factor of GA is the formula of fitness function. In the proposed task scheduling ALTS algorithm, fitness function is based on three parameters that are minimizing makespan, minimizing SLA violation and maximizing resource utilization in a load-balanced way as illustrated in Equation (1).

$$Fitness_{func} = Makespan + SLA + ARUR \quad (1)$$

The Makespan is calculated in Equation (2):

$$Makespan = Max\{CT_j\} = Max\{CT_1, CT_2, \dots, CT_m\} \quad (2)$$

where  $CT_j$  is the completion time of last task assigned to the  $VM_j$  and  $m$  corresponds to the number of VMs.

SLA violation occurs when any host reach their 100% CPU utilization and the user task need more CPU cycles for the execution. Equation (3) calculates the SLA violation.

$$SLA \text{ Voilation} = \sum_{i=1}^n \frac{T_{si}}{T_{ai}} \quad (3)$$

where,  $T_{si}$  is the time when the host is in violation and  $T_{ai}$  shows the hosts total active time.



The ARUR can be obtained using Equation (4).

$$ARUR = \frac{avgMakespan}{Makespan} \quad (4)$$

The avgMakespan is calculated in Equation (5).

$$avgMakespan = \frac{\sum_{j=1}^m Makespan_j}{m} \quad (5)$$

### 3.2.3. Selection Operation

This step selects the suitable solution for the next generation. In the genetic algorithm (GA), there are many selection techniques that select the best chromosomes from the population. In the proposed task scheduling (ALTS) approach, the selection criterion basically rely on the fitness function that includes parameters of makespan, SLA, and ARUR. In the proposed ALTS approach, the GA selects the best fitted elite chromosome, whose maximum value is near to 1 among all the calculated fitness function values of all chromosomes. When the best-fitted elite is selected, and then it is considered as an initial pheromone for ACO algorithm for further evaluation of optimal results.

### 3.2.4. Execution of ACO Algorithm

The best solution derived from the GA is further refined by an ACO algorithm. The transformation of the genetic algorithm (GA) solution into the initial pheromone of ACO is performed by Equation (6) as:

$$Init_{pher} = \varepsilon * GA_{BestSol} \quad (6)$$

where  $GA_{BestSol}$  is the optimal genetic algorithm (GA) solution and  $\varepsilon$  is a constant value. Initially, the ant colony optimization algorithm (ACO) assigns a pheromone value to the genetic algorithm (GA) best solution. Then ant colony optimization (ACO) allocates  $m$  resources to  $n - 1$  number of tasks in the form of a matrix as  $A(m, n - 1)$ . In the next move ACO, ant  $A_i$  changes its position from one state to the other state.

After that remaining VMs are updated and those ants moves that are infeasible deposited in a tabu list. When the first task is executed, the ant moves from the existing state to the next state. Then in the next iterations, ACO assigns the available resources to the upcoming tasks. This process continues until the desired best optimal solutions to all cloudlets are obtained.

During each single iteration, node transition rule is applied by an ant that is given in Equation (7) and it moves from the current state to the next state.

$$PSel_{m,n} = \frac{1}{\varepsilon + ARUR * SLA * makespan} \quad (7)$$

At this point, to avoid its division by 0,  $\varepsilon$  is a positive integer and ARUR, SLA and makespan are the average resource utilization rate, service level agreement and minimum makespan, respectively.

In an ant colony optimization (ACO) algorithm, before an ant enter into the next state, the amount of pheromone values on every path is rationalized by using Equation (8).

$$T_{m,n} = (1 - \rho) * T_{mn}(t - 1) + \Delta T(t - 1) \forall m, n \quad (8)$$

Here pheromone evaporation rate is denoted by  $(1 - \rho)$ , and updated pheromone trail values of each movement is represented by  $T_{mn}$  and  $\Delta(t - 1)$  is the amount of pheromone available on each single path. This value is computed when an ant traverse from the existing state to a new state, otherwise it remains 0. Thereafter, the ACO algorithm calculates the probability of each ant by using Equation (9).

$$p_m^n = \left\{ \frac{T_{mn}^a * PSel_{mn}^\beta}{\sum_{m \notin T_a} ((\alpha * T_{mn}) + (1 - \alpha) PSel_{mn})}, \forall m \notin T_a \right\} \quad (9)$$

where the quantity of pheromone value on each path is denoted by  $T_{m,n}$  and  $\alpha, \beta$  must be positive values, as these are used for updating the pheromone value. While assigning resources to the tasks, every movement of the ant is defined by the information of heuristic function and it is denoted by  $T_{mn}$ . The solution is decoded after producing the best optimal solution. Then that solution is submitted to the GA for the crossover and mutation operation to obtain a new offspring.

### 3.2.5. Crossover and Mutation Operation of GA

The crossover and mutation operations are achieved by selecting two individuals and then forming a new individual chromosome. In the proposed ALTS approach, two best solutions obtained from the GA fitness and ACO are selected for crossover operation. These best elites are composed of three parts (i.e., (i) left, (ii) middle and (iii) high). In the crossover operation, middle portion is interchanged and the left and right portions remain unchanged. The mutation process starts after the completion of the crossover operation. This process is used when the population becomes homogeneous due to the continuous use of crossover operation. Mutation change one or more gene values from its initial stage in the chromosome. Then new gene value is generated and added to the gene pool. While in mutation, to generate offspring, the spot of two gene values is exchanged. After performing all these steps, the solution result is produced and added to the actual population. This process continue until all tasks are allocated to their suitable resources. In the end, the proposed adaptive load balancing (ALTS) algorithm produced the optimal solution.

### 3.3. Steps for Proposed ALTS Algorithm

The main phases of proposed ALTS algorithm are given below:

1. Initialize the parameter values for GA and ACO that are required for population size, mutation, crossover, iteration number and pheromone evaluation, etc.
2. Set the values of the initial iteration number and initial population to 0.
3. Set iteration number to 1.
4. Fitness function of the GA is calculated using Equation (1).
5. When the iteration number approaches the highest value then check the value of fitness function if it reach then go back to step 6 otherwise go to step 5.
6. After the evaluation of fitness value, then we select two best chromosomes as ( $n$ th and  $n$ th  $- 1$ ) for next level.
7. Provide the  $n$ th  $- 1$  solution to the ACO as initial pheromone, where ants are represented by  $m$  and points are represented by  $n$ .
8. Equation (6) builds a mapping of cloudlets to VMs after selecting next state for each cloudlet.
9. Then the algorithm updates the pheromone value for each path and make a survey by checking that all the cloudlets are completed or not, if yes go to step 11, otherwise go to step 8.
10. To obtain a new offspring, perform the crossover and mutation operations.
11. Now analyze if the optimal solution is reached or not; if yes then move to step 12, otherwise move backward to Equation (9).
12. Returns the best optimal cloudlet to VM allocation.

The Pseudo Code of the ALTS approach is shown Algorithm 1.

### 3.4. ALTS Algorithm Pseudo Code

---

#### Algorithm 1: Adaptive Load Balancing Task Scheduling (ALTS) Algorithm

---

**Input:** Number of tasks ( $T_1, T_2, \dots, T_n$ ) arranged in queues

**Output:** Optimal allocation strategy for each task.

Step 1: Initialize tasks population size and VMs capacity.

Step 2: Evaluate the fitness function.

Step 3: While (until all tasks are allocated to their suitable resources) do

Step 4: Selection operation generates two foremost solutions ( $n$ th and  $n$ th – 1 chromosome).

Step 5: Modify the best solution ( $n$ th – 1 chromosome) from the GA fitness function to ACO

Step 6: Initialize values of pheromone,  $m \leftarrow 0, n \leftarrow 0$ .

Step 7: For all VMs  $m$  in tasks  $n$

Step 8: Calculate heuristic information by Equation (6).

Step 9: Calculate current pheromone trail.

Step 10: Calculate the probability of pheromone trail value by Equation (7).

Step 11: Choose the task with the highest probability value.

Step 12: Update pheromone trail values on each path.

Step 13: Modified solution of ACO ( $n$ th – 1) chromosome and GA solution ( $n$ th chromosome) combined for crossover operation to form a new optimal allocation strategy.

Step 14: End While

Step 15: Return optimal solution

Step 16: End.

---

## 4. Simulation Modeling, Design, and Analysis

This section discuss the experimental results and performance evaluation of ALTS approach against the existing compared task scheduling algorithms. To evaluate the performance of our proposed adaptive load-balanced task scheduling (ALTS) algorithm, we considered three algorithms related to task scheduling concerning makespan, SLA violation, and resource utilization. For pragmatic result evaluations, we utilized a renowned cloud simulator named CloudSim. The experiments are performed on a machine equipped with Intel (R) Core (TM) i5-5300U CPU @ 2.30 GHz and 20 GBs of RAM. For the employed simulation experiments, configuration details are illustrated in Table 2. All the experiments are performed on 32 virtual machines (VMs) within a DC. Here, the complete computing power of all virtual machines (VMs) is measured in million instructions per second (MIPS).

**Table 2.** Configuration of simulation environment.

Parameter	Value
Simulation Tool	CloudSim Software Version 3.0.3
Computing Power of Host Machine	Intel (R) Core (TM) i5-5300U CPU @ 2.30 GHz
Host Machine Memory	20 GB
Total number of VMs	32
Total number of Cloudlets	1024

### 4.1. Simulation Results

For the simulation experiments, the HCSP benchmark dataset instances of c-hilo, i-hilo, c-lohi, and i-lohi are used to evaluate the performance of the proposed approach against the available contemporary approaches concerning the average resource utilization (ARUR) and makespan are calculated. The HCSP dataset are based on the Expected Time to Compute (ETC) model and are obtained in a way that represents the real behavior of a heterogeneous environment. The HCSP instances employs the C-THMH pattern, where the alphabet C represents the consistency level that can be consistent (c), inconsistent (i), or

semi-consistent (s). The *TH* corresponds to the heterogeneity of tasks while *MH* denotes the heterogeneity machine. The details related to the dataset instances are available in [52].

Results are compared with the algorithm ACO (Kumar et al., 2018), genetic algorithm (GA) (Safwat et al., 2016), and hybrid combination of ACO and GA (Sayantani et al., 2018). The comparison shows that the proposed adaptive load-balanced task scheduling (ALTS) algorithm outperforms other task scheduling algorithms in terms of resource utilization and overall makespan. Every experiment is executed multiple times, and the average values are obtained and plotted.

Figure 4 shows the makespan results, where the proposed ALTS algorithm consumes on the average 48, 44 and 21% reduced makespan for the c-hilo dataset instance as compared to ACO, GA, and GAACO task scheduling algorithms, respectively. For the c-lohi dataset, the ACO has shown poor performance against the other approaches. The GA and GAACO has shown almost similar behavior on the c-lohi dataset. Again the proposed approach dominated the other approaches by reducing the makespan by 66% against the ACO approach and 38% against the GA and GAACO approaches. However, for the i-lohi dataset a slight improvement in makespan is observed for the proposed approach against the compared scheduling approaches.

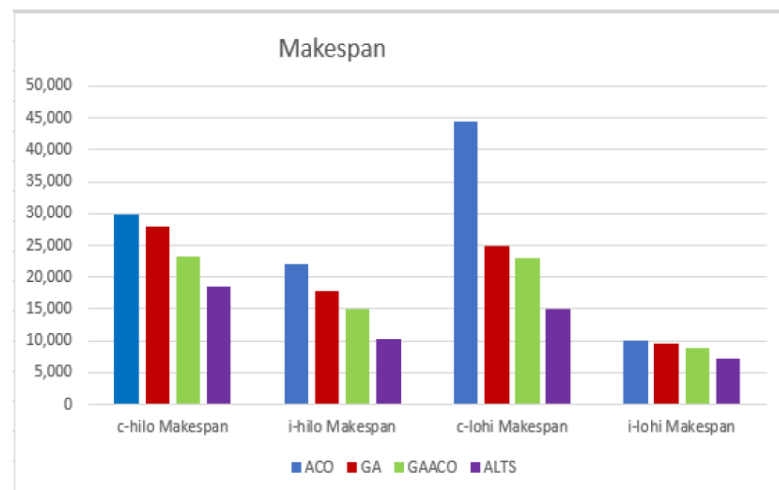


Figure 4. Makespan comparison.

Figure 5 plots the ARUR results of all the available approaches. The proposed ALTS approach has obtained 142%, 87%, and 25% higher resource utilization as compared to the ACO, GA, and GAACO task scheduling algorithms, respectively, for the c-hilo dataset instance. The ACO approach has again shown poor ARUR performance on the c-lohi dataset instance by attaining only 0.1 ARUR value. The GAACO has shown improved performance on the c-lohi dataset instances. Again the proposed approach dominated the contemporary approaches with an improvement of 25–145% higher ARUR.

Another important metric that is used to evaluate the performance of any task scheduling algorithm for cloud computing is SLA violation. The SLA violation is important parameter to ensure the quality of service which is an important requirement for the cloud service provisioning. One objective of this work is to minimize the SLA violation. The results concerning the SLA violation for the compared approaches are plotted in Figure 6. Once again the proposed approach has outperformed the compared approaches with the lowest SLA violation of 0.41. The proposed ALTS approach has reduced the SLA violation by 21% against the GAACO approach and 45% against the ACO approach. The GAACO approach has shown second best improved results concerning the SLA violation. The obtained results confirms the efficiency of the proposed approach and is suitable for workload with various heterogeneity level.

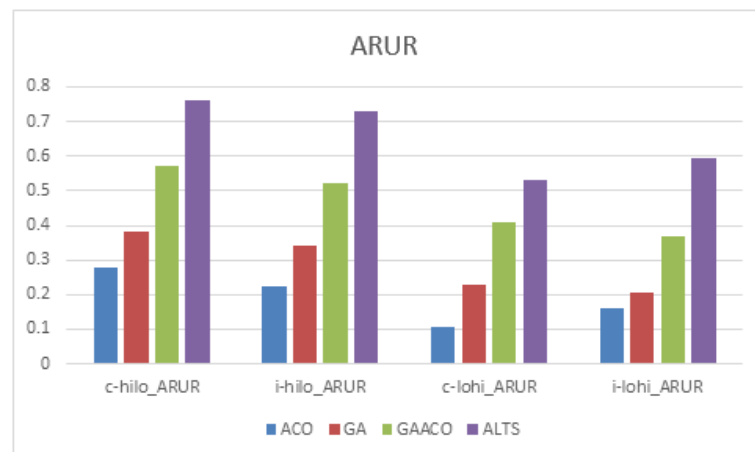


Figure 5. ARUR comparison.

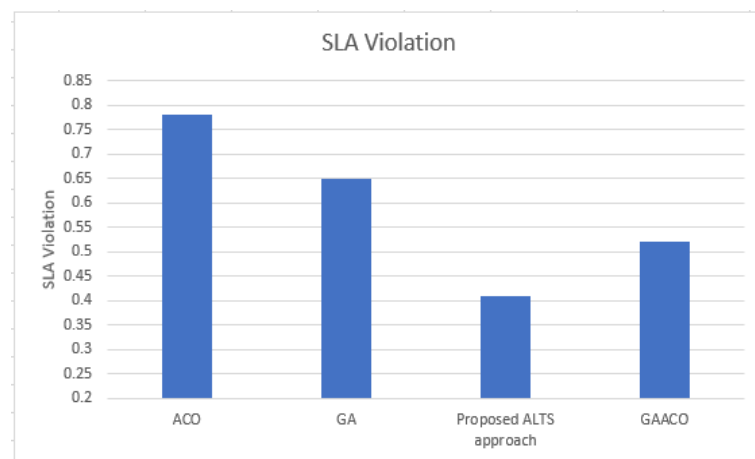


Figure 6. Average SLA violation comparison.

## 5. Conclusions and Future Work

In this work, an optimal task scheduling solution has been designed and implemented. The proposed approach has been evaluated and compared with ACO, GA, and existing GAACO approaches and the obtained results confirms the improvement concerning the makespan, resource utilization and SLA violation. In the proposed approach, the ACO is used to optimize and improve the solutions of Genetic algorithm. The proposed ALTS algorithm has shown improvement in the task scheduling concerning the makespan and resource utilization by the provisioning of load balanced task scheduling. Furthermore, the proposed approach minimizes the SLA violation to improve the quality of service. The obtained results confirms the effectiveness of the proposed ALTS approach.

In the last few years, energy aware strategies have shown huge attention from the research community. In future, an energy-aware task scheduling will be designed and implemented in real cloud environment.

**Author Contributions:** Data curation, A.M.; Funding acquisition, M.B. and H.H.; Project administration, M.I. and O.C.; Resources, H.H. and O.C.; Supervision, M.I. and N.B.; Visualization, M.B.; Writing—original draft, A.M. and M.I.; Writing—review & editing, M.I., A.M. and N.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** N/A.

**Informed Consent Statement:** N/A.

**Data Availability Statement:** N/A.

**Acknowledgments:** The authors thank Taif University Research Support Project number (TURSP-2020/239), Taif University, Taif, Saudi Arabia.

**Conflicts of Interest:** The authors declare that they have no conflict of interest.

## References

1. Ibrahim, M. SIM-Cumulus: A Large-Scale Network-Simulation-as-a-Service. Ph.D. Thesis, Capital University of Science and Technology, Islamabad, Pakistan, 2019.
2. Ibrahim, M.; Iqbal, M.A.; Aleem, M.; Islam, M.A.; Vo, N.S. MAHA: Migration-based Adaptive Heuristic Algorithm for Large-scale Network Simulations. *Clust. Comput.* **2020**, *23*, 1251–1266. [[CrossRef](#)]
3. Iturriaga, S.; Dorransoro, B.; Neschachnow, S. Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters. *Int. Trans. Oper. Res.* **2017**, *24*, 199–228. [[CrossRef](#)]
4. Ibrahim, M.; Nabi, S.; Baz, A.; Naveed, N.; Alhakami, H. Toward a Task and Resource Aware Task Scheduling in Cloud Computing: An Experimental Comparative Evaluation. *Int. J. Netw. Distrib. Comput.* **2020**, *8*, 131–138. [[CrossRef](#)]
5. Ibrahim, M.; Nabi, S.; Baz, A.; Alhakami, H.; Raza, M.S.; Hussain, A.; Djemame, K. An In-Depth Empirical Investigation of State-of-the-Art Scheduling Approaches for Cloud Computing. *IEEE Access* **2020**, *8*, 128282–128294. [[CrossRef](#)]
6. Ibrahim, M.; Nabi, S.; Hussain, R.; Raza, M.S.; Imran, M.; Kazmi, S.A.; Hussain, F. A Comparative Analysis of Task Scheduling Approaches in Cloud Computing. In Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Melbourne, VIC, Australia, 11–14 May 2020; pp. 681–684.
7. Selvakumar, A.; Gunasekaran, G. A novel approach of load balancing and task scheduling using ant colony optimization algorithm. *Int. J. Softw. Innov.* **2019**, *7*, 9–20.
8. Hayyolalam, V.; Kazem, A.A.P. A systematic literature review on QoS-aware service composition and selection in cloud environment. *J. Netw. Comput. Appl.* **2018**, *110*, 52–74. [[CrossRef](#)]
9. Basu, S.; Karupiah, M.; Selvakumar, K.; Li, K.C.; Islam, S.H.; Hassan, M.M.; Bhuiyan, M.Z.A. An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment. *Future Gener. Comput. Syst.* **2018**, *88*, 254–261. [[CrossRef](#)]
10. Lakra, A.V.; Yadav, D.K. Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. *Procedia Comput. Sci.* **2015**, *48*, 107–113. [[CrossRef](#)]
11. Priya, V.; Babu, C.N.K. Moving average fuzzy resource scheduling for virtualized cloud data services. *Comput. Stand. Interfaces* **2017**, *50*, 251–257.
12. Shukla, S.; Gupta, A.K.; Saxena, S.; Kumar, S. An evolutionary study of multi-objective workflow scheduling in cloud computing. *Int. J. Comput. Appl.* **2016**, *133*, 14–18. [[CrossRef](#)]
13. Nabi, S.; Ibrahim, M.; Jimenez, J.M. DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing. *IEEE Access* **2021**, *9*, 61283–61297. [[CrossRef](#)]
14. Iqbal, M.A.; Aleem, M.; Ibrahim, M.; Anwar, S.; Islam, M.A. Amazon cloud computing platform EC2 and VANET simulations. *Int. J. Ad Hoc Ubiquitous Comput.* **2019**, *30*, 127–136. [[CrossRef](#)]
15. Naik, K.; Gandhi, G.M.; Patil, S.H. Multiobjective virtual machine selection for task scheduling in cloud computing. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*; Springer: Singapore, 2019; pp. 319–331.
16. Waheed, M.; Javaid, N.; Fatima, A.; Nazar, T.; Tehreem, K.; Ansar, K. Shortest job first load balancing algorithm for efficient resource management in cloud. In *International Conference on Broadband and Wireless Computing, Communication and Applications*; Springer: Cham, Switzerland, 2018; pp. 49–62.
17. Bhushan, K. Load Balancing in Cloud Through Task Scheduling. In *Recent Trends in Communication and Intelligent Systems*; Springer: Singapore, 2020; pp. 195–204.
18. Panda, S.K.; Jana, P.K. A multi-objective task scheduling algorithm for heterogeneous multi-cloud environment. In Proceedings of the 2015 International Conference on Electronic Design, Computer Networks and Automated Verification (EDCAV), Shillong, India, 29–30 January 2015; pp. 82–87.
19. Zuo, L.; Shu, L.; Dong, S.; Zhu, C.; Hara, T. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access* **2015**, *3*, 2687–2699. [[CrossRef](#)]
20. Ping, G.; Chunbo, X.; Yi, C.; Jing, L.; Yanqing, L. Adaptive ant colony optimization algorithm. In Proceedings of the 2014 International Conference on Mechatronics and Control (ICMC), Jinzhou, China, 3–5 July 2014; pp. 95–98.
21. Mishra, S.K.; Sahoo, B.; Manikyam, P.S. Adaptive scheduling of cloud tasks using ant colony optimization. In Proceedings of the 3rd International Conference on Communication and Information Processing, Tokyo, Japan, 24–26 November 2017; pp. 202–208.
22. Saleh, H.; Nashaat, H.; Saber, W.; Harb, H.M. IPSO task scheduling algorithm for large scale data in cloud computing environment. *IEEE Access* **2018**, *7*, 5412–5420. [[CrossRef](#)]
23. Dordaie, N.; Navimipour, N.J. A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. *ICT Express* **2018**, *4*, 199–202. [[CrossRef](#)]
24. Alkayal, E.S.; Jennings, N.R.; Abulkhair, M.F. Efficient task scheduling multi-objective particle swarm optimization in cloud computing. In Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), Dubai, United Arab Emirates, 7–10 November 2016; pp. 17–24.
25. Verma, A.; Kaushal, S. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Comput.* **2017**, *62*, 1–19. [[CrossRef](#)]

26. Shishido, H.Y.; Estrella, J.C.; Toledo, C.F.M.; Arantes, M.S. Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds. *Comput. Electr. Eng.* **2018**, *69*, 378–394. [CrossRef]
27. Liu, J.; Luo, X.G.; Zhang, X.M.; Zhang, F.; Li, B.N. Job scheduling model for cloud computing based on multi-objective genetic algorithm. *Int. J. Comput. Sci. Issues* **2013**, *10*, 134.
28. Wang, T.; Liu, Z.; Chen, Y.; Xu, Y.; Dai, X. Load balancing task scheduling based on genetic algorithm in cloud computing. In Proceedings of the 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, Dalian, China, 24–27 August 2014; pp. 146–152.
29. Kaur, S.; Verma, A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment. *Int. J. Inf. Technol. Comput. Sci.* **2012**, *4*, 74. [CrossRef]
30. Hamad, S.A.; Omara, F.A. Genetic-based task scheduling algorithm in cloud computing environment. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 550–556.
31. Basu, S.; Kannayaram, G.; Ramasubbareddy, S.; Venkatasubbaiah, C. Improved genetic algorithm for monitoring of virtual machines in cloud environment. In *Smart Intelligent Computing and Applications*; Springer: Singapore, 2019; pp. 319–326.
32. Farhadian, F.; Kashani, M.M.R.; Rezaazadeh, J.; Farahbakhsh, R.; Sandrasegaran, K. WITHDRAWN: An efficient IoT cloud energy consumption based on genetic algorithm. *Digit. Commun. Netw.* **2019**. [CrossRef]
33. Liu, C.Y.; Zou, C.M.; Wu, P. A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. In Proceedings of the 2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Xi'an, China, 24–27 November 2014; pp. 68–72.
34. Kumar, A.S.; Venkatesan, M. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment. *Wirel. Pers. Commun.* **2019**, *107*, 1835–1848. [CrossRef]
35. Abdalkafor, A.S.; Alheeti, K.M.A. A hybrid approach for scheduling applications in cloud computing environment. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 1387–1397. [CrossRef]
36. Ibrahim, M.; Imran, M.; Jamil, F.; Lee, Y.J.; Kim, D.H. EAMA: Efficient adaptive migration algorithm for cloud data centers (CDCs). *Symmetry* **2021**, *13*, 690. [CrossRef]
37. Krishna, A.V.; Ramasubbareddy, S.; Govinda, K. Task scheduling based on hybrid algorithm for cloud computing. In *International Conference on Intelligent Computing and Smart Communication 2019*; Springer: Singapore, 2020; pp. 415–421.
38. Shanthan, B.H.; Arockiam, L.; Donald, A.C.; Kumar, A.D.V.; Stephen, R. Priority Intensed Meta Task Scheduling Algorithm for Multi Cloud Environment (PIMTSA). *J. Phys. Conf. Ser.* **2020**, *1427*, 012007. [CrossRef]
39. Arunarani, A.R.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [CrossRef]
40. Wang, Y.; Zuo, X. An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1079–1094. [CrossRef]
41. Keshanchi, B.; Souri, A.; Navimipour, N.J. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. *J. Syst. Softw.* **2017**, *124*, 1–21. [CrossRef]
42. Raju, R.; Amudhavel, J.; Kannan, N.; Monisha, M. A bio inspired Energy-Aware Multi objective Chiropteran Algorithm (EAMOCA) for hybrid cloud computing environment. In Proceedings of the 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCCE), Coimbatore, India, 6–8 March 2014; pp. 1–5.
43. Zhang, P.; Zhou, M.; Wang, X. An intelligent optimization method for optimal virtual machine allocation in cloud data centers. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1725–1735. [CrossRef]
44. Li, W.; Xia, Y.; Zhou, M.; Sun, X.; Zhu, Q. Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds. *IEEE Access* **2018**, *6*, 61488–61502. [CrossRef]
45. Zhang, X.; Wu, T.; Chen, M.; Wei, T.; Zhou, J.; Hu, S.; Buyya, R. Energy-aware virtual machine allocation for cloud with resource reservation. *J. Syst. Softw.* **2019**, *147*, 147–161. [CrossRef]
46. Babukartik, R.G.; Dhavachelvan, P. Hybrid Algorithm using the advantage of ACO and Cuckoo Search for Job Scheduling. *Int. J. Inf. Technol. Converg. Serv.* **2012**, *2*, 25. [CrossRef]
47. Torabi, S.; Safi-Esfahani, F. A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. *J. Supercomput.* **2018**, *74*, 2581–2626. [CrossRef]
48. Zahid, M.; Javaid, N.; Ansar, K.; Hassan, K.; Khan, M.K.; Waqas, M. Hill climbing load balancing algorithm on fog computing. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*; Springer: Cham, Switzerland, 2018; pp. 238–251.
49. Zhu, Q.H.; Tang, H.; Huang, J.J.; Hou, Y. Task Scheduling for Multi-Cloud Computing Subject to Security and Reliability Constraints. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 848–865. [CrossRef]
50. Zhang, P.; Zhou, M. Dynamic cloud task scheduling based on a two-stage strategy. *IEEE Trans. Autom. Sci. Eng.* **2017**, *15*, 772–783. [CrossRef]
51. Yuan, H.; Bi, J.; Zhou, M. Spatial task scheduling for cost minimization in distributed green cloud data centers. *IEEE Trans. Autom. Sci. Eng.* **2018**, *16*, 729–740. [CrossRef]
52. Cloud Task Scheduling Dataset. Available online: <https://iee-dataport.org/documents/dataset-task-scheduling-cloud-using-cloudsim> (accessed on 11 August 2021).