A Probabilistic Genetic Algorithm Approach to Efficient Task Scheduling in Cloud Environments

Kale Jyoti S*[1], Dr. Gokuldhev M[2]

[1]Research Scholar, Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Awadi, Chennai, Tamil Nadu, India

[2]Associate Professor, Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Awadi, Chennai, Tamil Nadu, India

*[1]Corresponding author mail ID: pavanjyoti2007@gmail.com

Abstract

Recently, cloud computing is an emerging as a Cloud computing is recently a profitable reality under the domain of information technology. Moreover, cloud computing signifies increment, consumption and distribution model in IT services, which are depends on internet on compensation for each usage basis. The cloud services are scheduled to the customers through service providers powered with the cost benefit of the computing model. Consequently, task prioritizing results in a blockage in these systems. Therefore, selecting tasks with the minimum execution time via robust algorithms is applicable. Hence, the genetic algorithm (GA) is among the evolutionary methods utilized to resolve difficulties rapidly. The traditional algorithms include the estimation-of-distribution algorithm (EDA), GA, and EDA-GA algorithms. These algorithms are used to optimize the resource utilization with less time complexity. However, the load balancing process that distributes the tasks over multiple computing resources and task scheduling within cloud computing increases the efficiency by optimizing the resource utilization, which ensures high scalability and availability. Hence, the probability of discovering a reorganizing-based GA has the intention to optimize the task allocated in the virtual machines (VMs), which reduces the completion time and execution cost when optimizing the resource utilization. By the impact of genetic operators such as selection, crossover and mutation algorithms effectively direct the solution space to determine optimal tasks. Furthermore, the performance of the algorithm is calculated via the Simpy toolkit, which indicates the capacity to outperform existing scheduling techniques. Moreover, the experimental results reveal that at 1000 iterations, the proposed model has a completion time of 10,841 ms compared with the other algorithms. Hence, it has a significant effect on efficacy and balances the competing time. This study provides functional effects to improve task scheduling approaches in cloud computing, providing a valuable structure for future research to optimize cloud resource management.

Keywords: Genetic Algorithm (GA), Virtual Machines (VM), Task Scheduling, Cloud Computing, Competing Time

1. Introduction

Recently, cloud computing has developed as a recent exploration issue, and it is generally used in broadcastings, the industrial sector, education and technical investigations [1, 2]. For example, clouds that are used for storage can [3]afford locked data storage and hold-up and record services, which offers high accessibility for customers. Moreover, the clouds that are used for educational purposes [4] can virtualize different kinds of hardware education resources and formerly transfer them to the internet system, presuming an appropriate statistical platform for education sectors, instructors and learners. With cloud computing, sources, namely, hardware, software and platforms, are given as services through the "pay-as-you-go" model. Furthermore, consumers have to pay only for the hardware structure and not for the hardware structure. The existing studies have focused mainly on virtualization, resource management, cloud safety, green computing, task scheduling, and so on. By means of cloud computing services, there is robust development for scheduling tasks effectively for virtual machines in accordance with the significantly increasing objectives. The objective of task scheduling predominantly comprises decreasing the task completion time with less energy consumption and refining the resource consumption with a load-balancing capability [5-7]. Through the electrical upsurge in the number of cloud users, decreasing the task

completion time remains beneficial for enhancing consumer involvement. Additionally, increasing load balancing can effectively employ VMs, which can avoid completing efficacy. This leads to a reduction in resource overload or waste generated by indolent resources [8-10]. For example, reducing the task completion time can easily schedule tasks. Moreover, resources with robust computing power can lead to load imbalance issues. However, it is challenging to structure and enhance the algorithms applied for the task scheduling process. In addition, these algorithms can balance the model with less completion time and enhance the load balancing capacity.

The complexity of task scheduling has been shown to be NP-complete; thus, the optimal solution cannot be acquired with a restricted time bound [11, 12]. For an issue, the solution can confirm polynomial time with the time taken to resolve the difficulties. These problems are measured with increasing problem size. Hence, recent computing methods are not utilized to determine precise results with the required time. For this particular instance, the issue can measure NP-complete [13], which suggests that the task scheduling problem remains NP-complete in the overall instances, along with certain limited cases, such as tasks scheduling through one or two intervals toward dual processors along with scheduling unit-time tasks toward an uninformed number of processors. [14] Hence, the problem has projected in almost the entire case that has been established as NP-complete where optimal solutions can be established when there is a comprehensive examination. Simultaneously, the intellectual model design of multifaceted systems is a main problem for organizational sensitivity to indecision. The task scheduling model in cloud computing is an intricate intellectual structure that comprises many tasks with diverse computing possessions [15].

Consequently, generative algorithms have resolved various scheduling and mapping complications. Moreover, the EDA remains a population-based development algorithm that has been established to be operative in resolving various optimization complications. Accordingly, the possibility model sharing of the resolutions in the search space is defined, and different resolutions are produced through the sample. EDA has a firm convergence speed to detect a good solution in minimal time. Conversely, the EDA can reduce within a local optimum. Furthermore, the GA is an algorithm that pretends at normal selection in addition to the genetic mechanism of genomic progression. The T algorithm is an equivalent and global search technique that offers an overall structure for resolving multifaceted structure optimization complications. Moreover, the selection, crossover and mutation functions can increase the search range of the results. The GA is considered a prodigious global search capability that efficiently rectifies during the insufficiency of the EDA; however, its convergence speed remains slow.

Motivated by the efficacious implementation of the EDA and GA and by the benefits and drawbacks, the proposed algorithm provides an effective approach for scheduling a task in a cloud system.

The key contributions of the study are as follows.

• The model concerns scheduling performance with time as the limit in scheduling and attains this limit by decreasing the task completion time and enhancing the load balancing capability.

• The study creatively relates EDA for task scheduling issues, and an integration of EDA and modified GA has been utilized to aid in detecting the optimal solution.

• The efficiency of the proposed algorithm is compared with that of the Simpy simulation experimental platform, and the experimental results indicate that the proposed algorithm is an effective task scheduling algorithm in a cloud system.

1.1. Paper Organization

The framework and structure of the paper are organized with a general introduction in Section 1. A review of several existing studies involved in analysing task scheduling via different algorithms is presented in Section 2, along with literature gaps. In Section 3, the proposed methodology shows that the entire process involved in the present model is deliberated. The results acquired by the proposed model and a discussion of the effects of the proposed model with other methods are provided in detail in Section 4, and the conclusions are summarized in Section 5.

2. Literature Review

Cloud computing has recently been utilized for numerous applications to solve workflow scheduling issues, which involve the allocation and scheduling of various resources within the cloud computing environment. Furthermore, to optimize the task scheduling problem, a multiobjective genetic algorithm (MOGA) has been implemented, and long common subsequence (LCS) is integrated with a genetic algorithm (GA), resulting in better performance [16]. In addition, random double adaptive whale optimization (RDWOA) is applied by enhancing the mutation operator of the bee algorithm, and performance is evaluated by using measures such as execution time,

computational cost and resource utilization time [17]. Similarly, the multiobjective cloud task scheduling method is used for solving problems where the G-MOTSA uses a modified GA to detect the optimal solution for the appropriate virtual machine (VM), and the stimulated results exhibit better performance [18]. However, min-min and max-min algorithms are introduced to create an initialization population, and task completion time and load balancing are chosen as dual fitness functions, which enhance the quality of population initialization and the search ability of the algorithm and obtain better results than a typical GA does [19]. An enhanced electrosearch algorithm using genetic operators is applied for effectual task allocation, and it is implemented on Cloudsim. Moreover, the benchmark problem is used for estimating the performance of the HESGA, and it yields better outcomes [20]. The MGGS is the combination of a GA with a greedy approach for maximizing task scheduling; hence, it is applied in cloud-centered environments and enhances the return of investment [21]. The balanced genetic algorithm (BGA) is implemented to enhance the make span along with load balancing. Additionally, it results in a workload with twisted, normal and uniform distributions with various batch sizes and yields better outcomes [22]. The MapReduce framework is utilized to decrease the execution time, where the GA is combined with heuristic approaches and has a better execution time [23]. In addition, a combination of a GA with thermodynamic simulated annealing (TSA) is implemented, and it is used to schedule a communication-intensive workflow [24]. Similarly, differential evolution (DE) with SA and GA-SA is incorporated into the greedy method, and better results are obtained when the DESA algorithm is used [25].

Correspondingly, the nondominated sorted genetic algorithm (NSGA-III) is used to schedule the tasks for the group of VMs in the cloud-based MOAF to reduce the power consumption, runtime and cost, and it has achieved better results than NSGA-II [26]. In hybrid lion-based GA, the lion optimizer is used to balance the load by enhancing the selection of optimal parameters for VMs, and it attains the global search criteria with better outcomes [27]. In addition, a multiverse optimizer with a GA is used to optimize the task scheduling for a large set of tasks, and it is simulated in a cloud environment via the MATLAB distrust system [28]. Moreover, the multipurpose weighted GA is used to signify the fitness function to mitigate the time and cost parameters, and the statistical results have reached 94% [29]. The double-layer resource management system GA with edge computing is used to schedule the task dynamically for various scheduling tasks, and it has good results in terms of energy consumption, load balancing and time delay [30]. A new adaptive GA (NAGA) is subsequently applied to enhance the crossover mutation genetic operator, which greatly reduces the probability of the algorithm falling towards local optimal resolution [31]. The combination of a GA with Particle Swarm Optimization (PSO) implemented self-cognition with social perception of PSO to confirm power utilization, and it has acquired 23.0 to 33.2%, with resolutions ranging from 27.9 to 43.7% [32]. The incorporation of a GA and energy-conscious scheduling for better scheduling tasks for processors results in minimum makespan and energy consumption, which are estimated via MATLAB [33]. In addition, an HGA method was implemented to manage task scheduling in the cloud, and it achieved an execution rate of 32.57 ms [34]. The hybrid Max-Min GA for each VM is applied for high load balancing, and because of its lower time complexity and cost binary, the use of PSO yields better results for five VMs [35].

In the study, [36] benchmark datasets are compared with the typical GA, PSO, where the existing algorithm provides the best solution for every parameter and gives the greatest solution for all parameters. Moreover, the task scheduling problem is formulated by the M/M/n queuing model and uses a sequential manner with a precise number of tasks. The performance is compared with that of BATS, the IDEA and BATS+BAR, and it attains better accuracy [37]. The semidynamic real-world task scheduling algorithm is implemented for the bag-of-task in a cloud environment, and it has attained better results first and best fit and provides a balance among the makespan and execution cost with minimal task failures [38]. The scheduling algorithm is based on a GA with a flower pollination centred algorithm (FPA), which has revealed optimized resource utilization with minimum energy consumption at low completion times [39]. In [40], the task includes interdependent subtasks that are scheduled to minimize the offloading latency with the failure probability and attain the optimal solution.

2.1. Problem identification

- The traditional scheduling algorithm does not minimize the execution time of the allocated task; hence, it leads to an increase in waiting time and delay during the completion of tasks [15].
- Complex complexities in task scheduling occur because of improper scheduling, which leads to resource overload and effectively affects system performance [30].

Conversely, in traditional GA this methods includes probabilistic functions, which dynamically adjust with the progressing cloud environment, may impact on additional optimized scheduling solutions. The key contribution rely in the capability of balanced assessment and utilisation more effectively, results in enhanced resource utilization with reduced task completion times. This probabilistic improvement differentiates the proposed method

against existing algorithms by providing a more adaptable and effective scheduling mechanism structured precisely for the dynamic and heterogeneous feature of cloud computing backgrounds.

3. Proposed Methodology

Cloud computing allows for the sharing of scalable computing resources and fast, convenient network access. These assets can be quickly established and produced with less supervision or participation from service providers. Cloud computing is a process that is used for calculating computing resources through the internet whenever needed. In contrast to the typical internet, cloud computing provides better reliability, scalability, cost effectiveness, and various other advantages. These features allow the cloud platform to reduce expenses, improve adaptability, speed up time to reach value, scale more effectively, and address performance decline in extensive information systems caused by insufficient computing and management capabilities. The combination of task scheduling along with load balancing can enhance performance and high reliability with better resource handling. Moreover, task scheduling comprises assigning tasks to obtainable resources to optimize the performance and utilization of resources. It considers task priority and the availability of resources, and the scheduling algorithm can reduce completion and waiting times. However, load balancing combines with task scheduling, enables dynamic scheduling and reduces bottleneck problems. Hence, the proposed model uses the EDA algorithm with a modified GA, as shown in Figure 1.

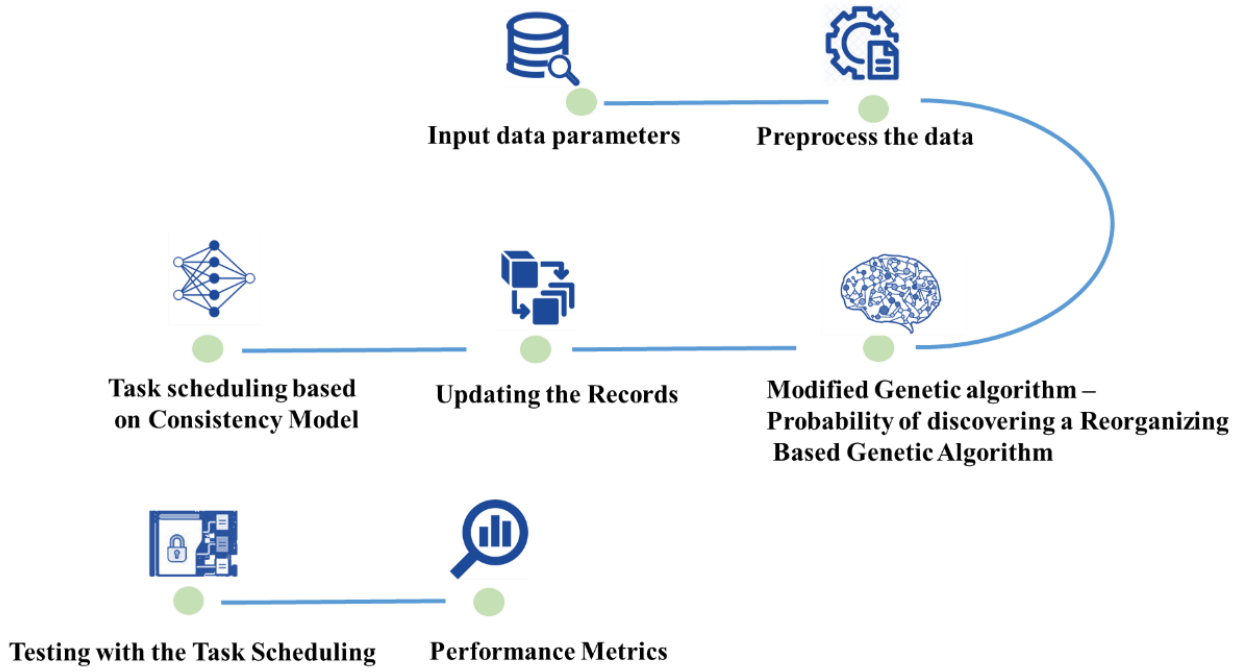

Figure 1. Overall Flow of the Proposed Model

The proposed model focuses on optimizing the distribution of tasks toward resources in a cloud background. The method applies a GA that simulates the procedure of normal selection toward iteratively progressing solutions that aim to reduce the time required to complete a set of tasks, although it improves the utilization of resources. Moreover, significant components of the technique comprise a representation of a task, where tasks described as nodules also illustrate the dependencies between tasks. The GA structure includes various phases, namely, the initialization of the population, where possible solutions are produced. The other is the evaluation of fitness, which measures every solution's efficiency with decreasing time and exploiting resource usage. The best-execution solutions for reproduction are subsequently selected. The implementation of crossover with mutation methods searches the solution space. Then, a probability-based device is combined to robustly regulate selection and mutation procedures, enabling variation towards changing workloads along with resource obtainability. Moreover, resource allocation is accompanied by allocating tasks toward VMs consistent with the optimized schedules, confirming that dependent tasks are implemented in the accurate order with efficient resource utilization. Moreover, the benefits of the proposed method include scalability, which enables it to achieve a wide range of scheduling tasks; flexibility, which enables variations towards accommodating several limitations; and performance upgrading, as the algorithm can determine optimal schedules that exceed typical systems. Hence, the proposed method improves task execution efficacy in cloud computing through perceptively distributing resources depending on evolutionary values, efficiently managing the complications of active and mixed cloud backgrounds.

3.1. Dataset Description

In cloud computing, task scheduling is accomplished via a probability-based GA, whereas various key input data parameters are necessary for optimizing performance. The parameters include the number of tasks to be scheduled, along with related attributes such as task length, limits, and significance. Furthermore, the features of obtainable VMs perform a main role that contains the VMs, with the processing speeds and the cost for every unit time. The GA involves parameters such as population size, number of generations, and crossover possibility, with mutation possibility. The objective is to reduce the completion time with execution costs while exploiting resource utilization, confirming effective task allocation to VMs, and sustaining system performance above changeable loads and resource accessibility. Moreover, the parameters collectively contribute to the efficiency of the scheduling algorithm in attaining optimal task distribution with implementation in a cloud background. Moreover, as part of the input and pre-processing phases, create randomized parameters for the virtual machines (VMs) in this dataset. Variable population sizes (10 and 100 VMs) should be taken into consideration by the parameters. This procedure ensures that the data is relevant and applicable to cloud computing contexts by simulating those environments.

3.2. Preprocessing

In the preprocessing section, the parameters stimulate the balance among the search and utilization in the search of optimal task schedules, predominantly affecting the efficacy and efficiency of the scheduling process in cloud systems. Moreover, parameters such as the population size, elite population size and learning rate are determined. In addition, the population size and elite size control the search–deployment determination, whereas the learning rate affects the convergence speed.

Task Scheduling Mechanism

During task scheduling, users provide tasks toward the cloud structure, which comprises three elements—task and resource managers—along with the scheduler. Moreover, the cloud system directs tasks towards the task manager such that procedures tasks within batch mode also achieve data such as the task size. Moreover, the resource manager regularly achieves the VM and acquires data such as the calculation speed of the VM. The obtained data, such as the task size, are subsequently acquiesced through the task manager along with the calculation speeds of VMs, which are capitulated via the resource manager, and the scheduler subsequently develops its function. Moreover, the scheduler is the essential element and accounts for tasks allocated to VMs via the EDA-GA algorithm proposed in the study. Moreover, Figure 2 shows the framework of task scheduling in cloud computing.

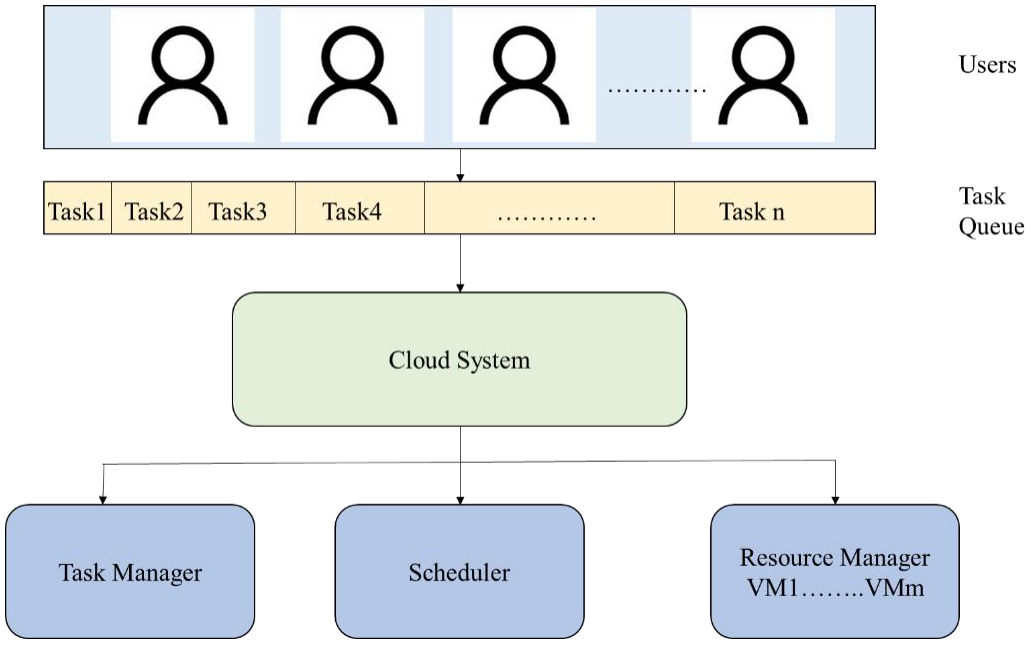

Figure 2. Task Scheduling Mechanism

3.3. EDA Algorithm

Estimation-of-distribution (EDA) is a common metaheuristic that is utilized for optimizing evolutionary algorithms. Moreover, EDAs do not directly evolve the population of the search spaces; however, they construct probabilistic models for providing solutions through iterative sampling along with selecting points in the search space. The simple EDA includes a description of benchmark functions and is implemented for analysis. Figure 3 shows the structure of the EDA algorithm, which is depicted below.

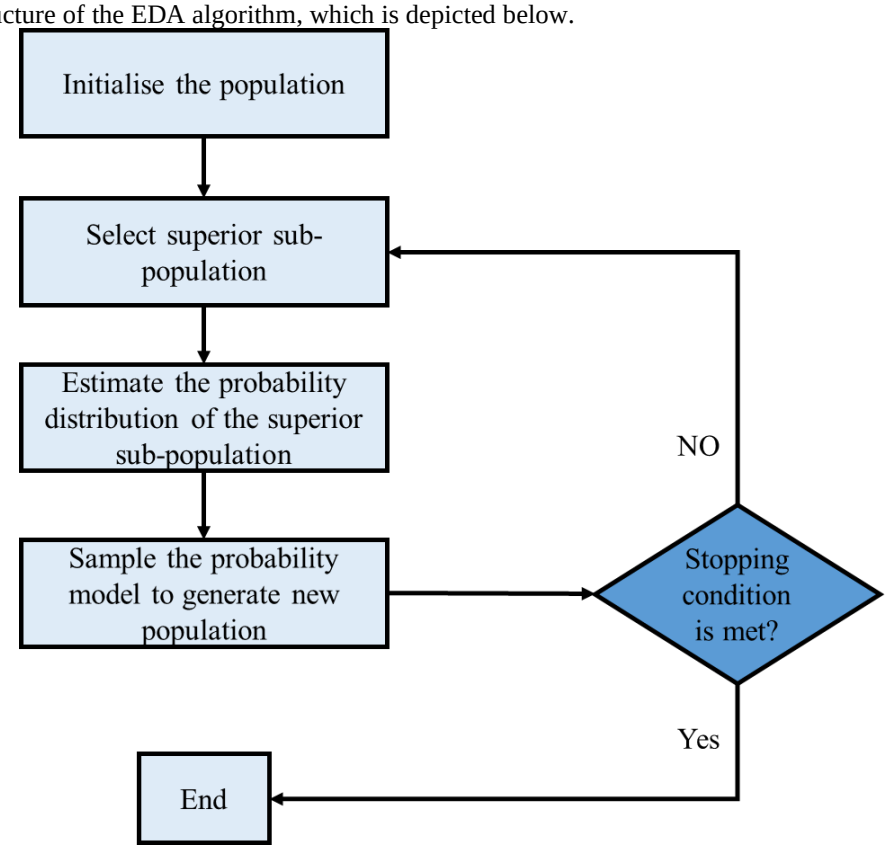

Figure 3. Structure of the EDA Algorithm

EDAs are speculative optimization algorithms that investigate the candidate solution space through sampling a definite stochastic model made from optimistic solutions that are estimated accordingly. Moreover, EDAs normally work via candidate population solutions to these issues; initially, the population is generated in accordance with a regular distribution of acceptable results, and then, the population is attained via a fitness function. Moreover, the fitness function contributes a statistical position for every sequence, where the string shows better results with more numbers. Through the classified population, a subdivision of the furthermost optimistic solutions is designated through the selection operator. Furthermore, the algorithm formerly makes a stochastic model that endeavours to estimate the probability distribution for the selected solutions. After the construction of the model, different solutions are produced through sampling, and then, diffusion is set by the system. These solutions are then combined as the former population, perhaps changing it completely. The procedure is repeated until certain dissolution measures are encountered, with every repetition of the process typically referred to as a unique generation of the EDA. The EDA process is defined in Algorithm 1.

Algorithm-I EDA

$t < -0$
produce first population $P(0)$
while (not)do
Choose population of solution $a(t)$from $P(t)$
Construct probabilistic model $M(t)$from $a(t)$
insatnce $M(t)$to produce new candidate solutions $O(t)$
combine $O(t)$in $P(t)$
$t < - t + 1$
end while

The algorithm shows the structure of the EDA, which is a generative algorithm that starts with the initialization of an early population of solutions and is defined as P(0). The algorithm subsequently passes into a main loop that remains until a stated termination form is encountered, such as attaining a definite number of repetitions or defining an acceptable resolution. In every iteration, a solution subset is designated from the recent population P(t) depending on the fitness, and it is denoted as a(t). However, an uncertain model M(t) is constructed by these selected solutions, taking the connections and patterns intrinsic to them. The model is subsequently sampled to produce distinct candidate solutions O(t) that are predicted to possess enhanced features related to the predecessors. The recently produced solutions are combined with the existing population P(t), interchanging certain or entire prevailing solutions, thus presenting variety and investigating distinct spaces for the search space. The repetition counter t is increased by 1, and then the procedure is reiterated until the stoppage state is fulfilled. EDAs are mostly beneficial in confronting optimization complications considered through difficult or high-dimensional search spaces, whereas classical evolutionary algorithms can compete to competently recognize optimal solutions. By exploiting a stochastic model, EDAs effectively capture and achieve the essential framework of the problem, managing the search to optimistic areas of the search space.

3.4. GA Algorithm

GA is a technique for searching a wide-range solution space via the concepts of evolution and genetics. With the GA, every solution in space is signified by an individual named a chromosome. However, every chromosome can be estimated by a fitness value that is connected to the cost function of interest.

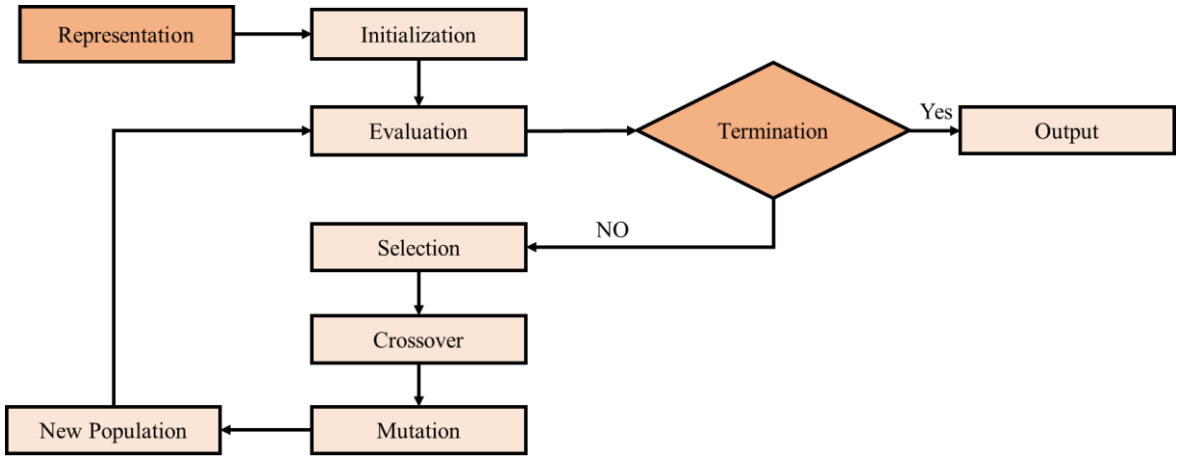

Figure 4. Structure of the GA Algorithm

Figure 4 shows that GAs are optimization methods that are stimulated through the concepts of regular selection and genetics, which are considered to perceive optimal solutions toward difficult complications. The process begins with a randomly generated population of potential solutions, each represented as a chromosome. Moreover, every individual is estimated on the basis of a fitness function that computes its efficiency compared with the accessible difficulty. The algorithm formerly determines that the fittest individuals can function as parents for the succeeding generation, relating genetic operators, namely, crossover, which is used for combining parental chromosomes. Mutation introduces random variations for creating offspring. Therefore, the sequence of selection, reproduction, and mutation continues for several generations to progress to a population that gradually estimates the optimal solution. Furthermore, GAs are predominantly effectual in a wide range of search spaces where typical optimization approaches can compete, such as the range over which various solutions can compete concurrently and control chronological data to show the search procedure regarding better solutions above time.

Algorithm-II - GA

Begin GA Algorithm
While Not Stop DO
Initialize the population at random;
Begin
Select i, by a t process for mating;
Apply a crossover operator to create the new i;
Apply mutation operator on the new i created;
Compute the f of the i;

Fix the easy constraint with the repair operator;
Insert this i into the population, and remove the i with the lowest f;
Check if stopping criteria are satisfied;
End
Output best i found
End GA Algorithm;

From Algorithm-I, it is inferred that the GA initializes by a population that includes individuals, signifying the primary solution to the complication typically in the arrangement of a bit vector, which is denoted as {P0, P1, …, Pmax}, whereas every population includes m entries and is given as {R1, R2, …, Rm}. Additionally, every entry has Ri, which is set to {rk, s, t, $1 \leq k \leq n$ and rk = {0, 1}}. Whereas {r1, r2, …, rn } signifies the set of available distinct resources. Moreover, the probability selection is given by 0 and 1, whereas the particular resources are denoted by the value 1; then, the denied resources are signified with the value 0. The chromosome fitness values for a certain population are calculated on the basis of the computational competence, otherwise referred to as processing power, which is related to the precise resource. Consequently, the fitness function fitness(R*) is defined, and it is utilized to calculate and save the best-fit chromosome produced by a specified population. With respect to the GA, the defined solutions have the potential where the lowest attribute measurements are required for every resource allocation. Moreover, the solutions are produced by best-fit searching, which originates at the bin-packing difficulty. Furthermore, the heuristic is firmly effective. It begins with the consideration of the initial resource to find regardless of whether it can be distributed to the user job and, diversely, transfers to the resource and so on. To produce more different solutions, the crossover and mutation rates and the resource selection structure are improved by exploiting a random permutation. In addition, population flooding can be avoided by the scale-down operator. It is presented as a measure of the genetic operator used to control the number of individual copies. In the subsequent initialization stage, a repetitive procedure involves Algorithm II. Moreover, the principle of presenting the repair operator is to confirm that all bits of the chromosomes do not overdo the value 1 once the fitness surpasses the selected capability. The repair operator attains the results by constructing the bit of the chromosome, which is equal to 0 at the point, whereas the fitness operator outstrips the preferred measurements. These stages are tunable through certain parameters, resulting in diverse instances of the identical GA.

3.5. EDA-GA Algorithm

The EDA-GA algorithm incorporates the robustness of evolutionary strategies to optimize complex problems. First, a population of potential solutions is produced, and every individual is assessed by a fitness function to define its quality. In contrast to typical GAs, which depend profoundly on crossover and mutation to create distinct solutions, GA-EDA uses a stochastic model that encapsulates the distribution of the best-accomplishing individuals. Then, the model is experimented with to generate distinct candidate solutions, efficiently investigating the search space centred on learned patterns of preceding generations. The algorithm repeatedly enhances the stochastic model by determining efficient individuals, relating genetic operations, and restoring the model to lead imminent generations. The hybrid technique improves search and utilization, supporting further effectual convergence for optimal solutions while continuing variety in the population.

Algorithm-III EDA-GA

Input: tasks set K, virtual machines set V
Output: A result for task schdeluing in VM
Set the possibility model
while iter $\leq$ itermax do
Sample the possibility model to generate the population
for every task in the population do
Evaluate GV alue with $time_j = \sum_{r=1}^{q} EKC(t_{r,} r_j)$
Arrange entire individuals in decresing order relevant to GV alue
Elect the top 50% of the effecient individuals, signifies as K EP
for every individual in K EP do
Execute crossover function
Execute mutation function
Evaluate the new population generated in the two steps as K NP
end for
for every individual in K EP and K NP do

Evaluate GV alue with $time_j = \sum_{r=1}^{q} EKC(t_{r,} r_j)$
Arrange entire individuals in decreasing order with GV alue
Elect the top p% of the effecient individuals to obtain the elite population
end for
end for
Utilize the elite population for updating the probability model iter + +
end while

The algorithm describes a scheduling method for allocating a group of tasks K through a set of virtual machines V by a stochastic model with evolutionary approaches. Initially, the algorithm assembles a possibility model to lead the scheduling procedure. In addition, the main loop repeats until it reaches a maximum number of iterations, and it is signified as itermax and the process until it is reached. In every iteration, the model is experimented with to produce a population of possible task projects. For every individual in the population, a GValue is designed, which depends on the overall execution time of tasks allocated to VMs and is represented as timej=∑r=1qEKC(t(r)rj. In addition, the individuals are arranged in descending order consistent with the GV values, and the top 50% of the best-performing individuals, denoted as KEP, are designated. Moreover, crossover and mutation operations are performed on the selected individuals to generate a distinct population, which is represented as KNP. Consequently, the GV values of complete individuals in both KEPand KNPare evaluated, and the top p% of individuals are designated to configure an elite population. This elite population is utilized to improve the possibility model, and the repetition counter is incremented. This procedure continues until the determined number of repetitions is extended, eventually allowing an enhanced solution for scheduling tasks through VMs.

3.6. Probability of discovering a reorganizing-based genetic algorithm via task scheduling in cloud computing

The proposed GA is used for scheduling tasks in cloud computing, which includes optimizing the allocation of tasks for VMs through evolutionary strategies. In this system, a population of potential solutions is initialized to evaluate the fitness used to minimize the make-span and optimize the resource usages. Through the integration of reorganizing-based pairing (RBP) in the fitness function calculation, added complexity along with structure is introduced into the GA, as illustrated in Figure 5.

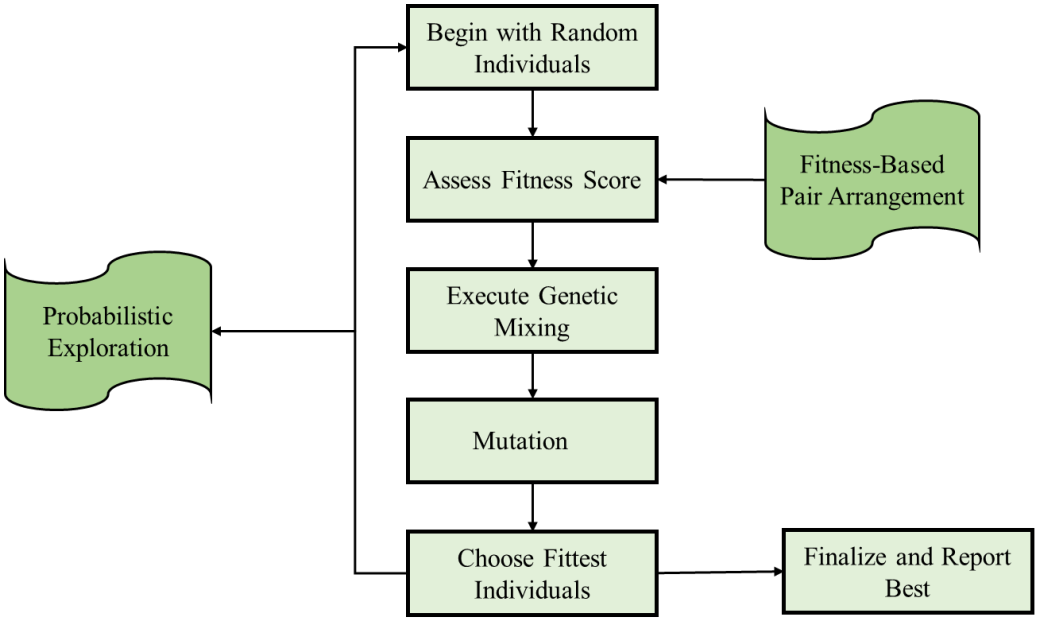

Figure 5. Architecture of the proposed model

In the proposed RBP calculation, fitness functions are evaluated, and the pairing calculation involves grouping task-scheduling elements into pairs and reshuffling them via a designated reorganization strategy. Hence, the

reorganization-based pairing calculation method enhances the fitness evaluation by allowing the fitness function to be applied to the rearranged pairs, leading to a more thorough assessment of their performance. Various approaches, such as rearranging pairs, can be employed during the reorganization phase to accurately portray the relationships between paired components. Ultimately, this approach aims to increase the exploration of the solution space and enhance the genetic algorithm's performance through leveraging the connections between paired elements, which is particularly advantageous in complex optimization tasks.

The use of a probabilistic discovery approach significantly increases the likelihood of discovering effective strategies in scheduling tasks. One illustration is a method for arranging tasks in grid environments aimed at minimizing average response times, demonstrating that incorporating probability can lead to better scheduling effectiveness. Additionally, research has shown that the proposed GAs effectively solve task scheduling problems in cloud computing by optimizing parameters to reduce completion time and improve resource allocation, underscoring the importance of probabilistic methods in enhancing task scheduling effectiveness. Typically, combining probabilistic models and adaptive algorithms increases the likelihood of finding optimal scheduling solutions in complex computing environments.

To develop the task rule, a distinct method is provided that does not depend on the task sequence. Thus, the task scheduling contains users alone. In addition, the decoding step is divided into two levels of user-only task scheduling. The initial level distributes user-node task scheduling into various groups, whereas the accumulation of every set's package weight does not outweigh the optimal loading ability of the united tasks. In addition, the space between every route does not exceed the determined track ranges between all the tasks. Formerly, a route set is generated by a series of task schedules along with a consistent travel time devoid of considering that the task can proceed precisely. On the other hand, the generated routes are allocated to tasks through creating a subproblem through the distribution time for the generated routes and the capacity of various kinds of tasks as inputs.
Moreover, the subproblem reduces the reciprocal of the GA's fitness function and is given by

$$fitness_i = \frac{1}{max\{\sum_p time_{\rho,k}, \forall k\}} \quad (1)$$

In contrast, $time_{\rho,k}$ defines the essential time for task k to attend the customers in a group ρ. To compute the optimum fitness value, a minor integer programming model can be built for ideally configuring distribution paths for the tasks.

Here, an index of the task is used, and it is denoted as i for every task category; every I has its ni tasks. Moreover, every i task has an identical capacity, speed Vi and optimum range Ri, and j represents the route index.

$$T \geq \sum_j \frac{X_{ijk} * r_j}{V_i} + (\sum_j X_{i,j,k} - 1) * L\_i, \forall i, k \quad (2)$$

However, $Xijk$ is represented as a decision variable, signifying that route j is distributed towards the kth task of the i type. The balance of the model is based on the number of routes with the number of tasks.

$$\omega_j * X_{ijk} \leq W_i, \qquad \forall i, j, k \quad (3)$$

$$r_j * X_{ijk} \leq R_i, \qquad \forall i, j, k \quad (4)$$

$$\sum_i \sum_k X_{ijk} = 1, \qquad \forall j \quad (5)$$

Furthermore, the travel distance for route j is expressed as r_j. Furthermore, every route can guide packages to certain users, and the load of packages is defined as follows: and its resolution is to reduce the complete time T.

Equation (2) expresses that entire tasks can complete the task at T time. Furthermore, equation (3) requires that the package weight of route j not outstrip the ability constraint of the determined task. Furthermore, equation (4) creates a definite flight range of the designed task that can protect the total distance of the trip. However, equation (5) ensures that every route is allocated to a single task. All the variables in the model are binary, and it is easy to solve when it is compared with the actual model. Since a state has I tasks along with J routes, the typical model of the subproblem remains reduced in scale relative to the typical model. Through $I * J$ variables along with I 2 $*$ J limits. Through various perceptions, the typical model projected above fundamentally resolves scheduling difficulties in tasks. The optimal objective with the resolution of the scheduling issue is combined into the proposed GA over the fitness function along with task scheduling.

The parameter P_a denotes the probability of searching for the nodes in the search algorithm, and it is modified robustly and expressed in equation (6).

$$P_a = P_{\text{amax}} - \frac{P_{\text{amax}} - P_{\text{amax}}}{iter_{\text{amax}}} * \text{iter} \quad (6)$$

where P_{amax} represents the optimistic probability of the search, which is the highest probability in the search algorithm, and where P_{amin} represents the lowest probability in the search algorithm. Therefore, the subproblem does not include and affects further functions in the GA, namely, crossover or mutation, and the proposed GA is shown in Algorithm III.

Algorithm-III Probability of discovering a Reorganizing Based Genetic Algorithm using Task scheduling in cloud computing

for all task − scheduling v ∈ Pg do
Choose task d1 , remaining capacity $x1$, traveling distance $q1$, working time $\epsilon 1 = 0$
for all Customer j ∈ i do
if Customer j can be served by task dk then
Update xk , qk and ϵk
else
Record the type of dk as n
Record task dk 's traveling distance as rn , its packages weight Wn
if dk is the last task in the queue then
Choose task d1
Update $x1$, $q1$ and $\epsilon 1$
else
Choose task dk + 1
Update xk + 1, qk + 1 and ϵk + 1
end if
Update ϵk after returning to the hub
end if
end for
$P_a = P_{\text{amax}} - \frac{P_{\text{amax}} - P_{\text{amax}}}{iter_{\text{amax}}} * \text{iter}$
Input all rn , Wn , P_a and task data into the IP model
Compute the optimal time Ti by allocating all routes rn to all task
end for

The algorithm provides a method for enhancing task scheduling in cloud computing, possibly in a situation where delivery or service tasks must be allocated to resources effectively. The algorithm is repeated for every task scheduling v in population Pg, which represents probable solutions. For every task scheduling, the process starts by selecting an initial assignment d1 and initializing parameters such as obtainable capacity x1, travel distance q1, and function time ϵ1 to zero. In addition, the algorithm measures each user j related to the recent task dk. If task dk can assist customer j, the algorithm can consequently regulate the enduring capacity, travel distance, and working time. Otherwise, it records the task group with a consistent travel distance with the weight. Moreover, dk remains the absolute task in the record, and the algorithm returns to the initial task d1 and consequently regulates the parameters. Otherwise, it transfers on to the ensuing task dk+1 in addition to revising the essential parameters for the task. When users function, the algorithm regulates its working time ϵk while recurring towards the hub. The probability Pa is improved to be consistent with the maximum probability Pamax and the recent repetition count and affects the selection procedure in imminent repetitions. Eventually, the detailed travel distances, weights of the packages, and adjusted possibilities are stored in an IP model to regulate the ideal time Ti for allocating entire routes for the tasks. The precise method promises an efficient scheduling process that considers various limits and enhances resource allocation in the cloud environment.

4. Results and Discussion

This section describes the results obtained by the proposed model along with the various algorithms and is discussed in detail.

4.1 EDA (Exploratory Data Analysis)

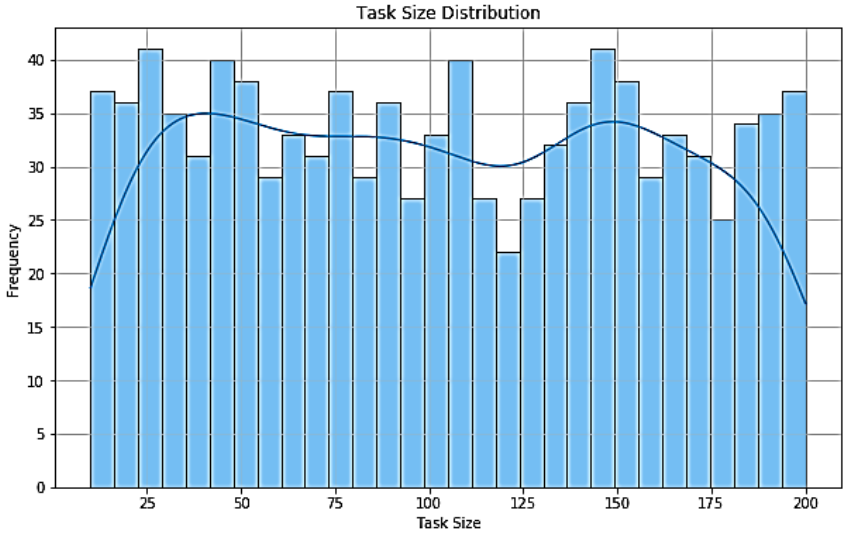

Figure 6. Histogram of Task Size Distribution

The distribution of task sizes in a simulated cloud computing environment is graphically depicted by the histogram in Figure 6. The y-axis shows the frequency of each task size, which ranges from 0 to just over 40, while the x-axis shows the task size range, which runs from 0 to 200. The histogram's bars are visually distinct and simple to understand because they are light blue with black edges. The semi-transparent bars make it easy to see the smooth blue line that represents the overlaid Kernel Density Estimation (KDE) curve. A smoothed approximation of the underlying data distribution is provided by this KDE curve, which highlights patterns that might not be immediately apparent from the bars alone. The comparison between the histogram's frequency and KDE curve's generalized probability density, which provides a complete insights of task size distribution, emphasizing both absolute observed frequencies in bins and the entire continuous shape and possible patterns within the data, like peaks signifies higher task concentrations at some sizes along with complete spread.

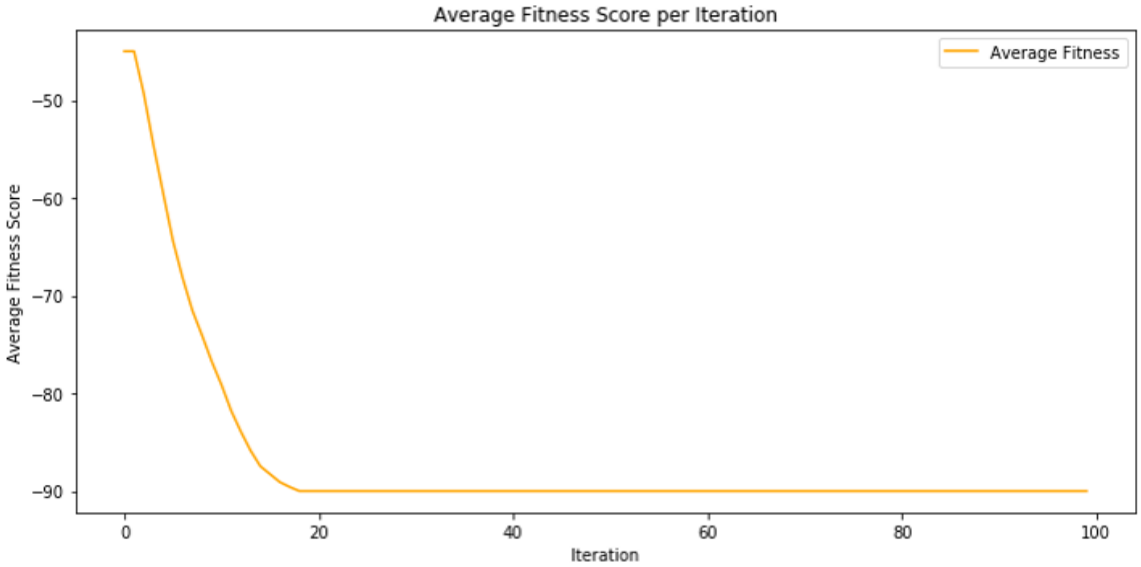

Figure 7. Graphical representation of the average fitness score per iteration

Figure 7 shows that the average fitness score is a mean of the fitness value over the population. Every generation can change its population to attain a distinct average population fitness. Here, when the average fitness score is high at the initial point, it gradually decreases, and at 20 iterations, it remains constant. In addition, the figure shows the best fitness score, which is depicted.

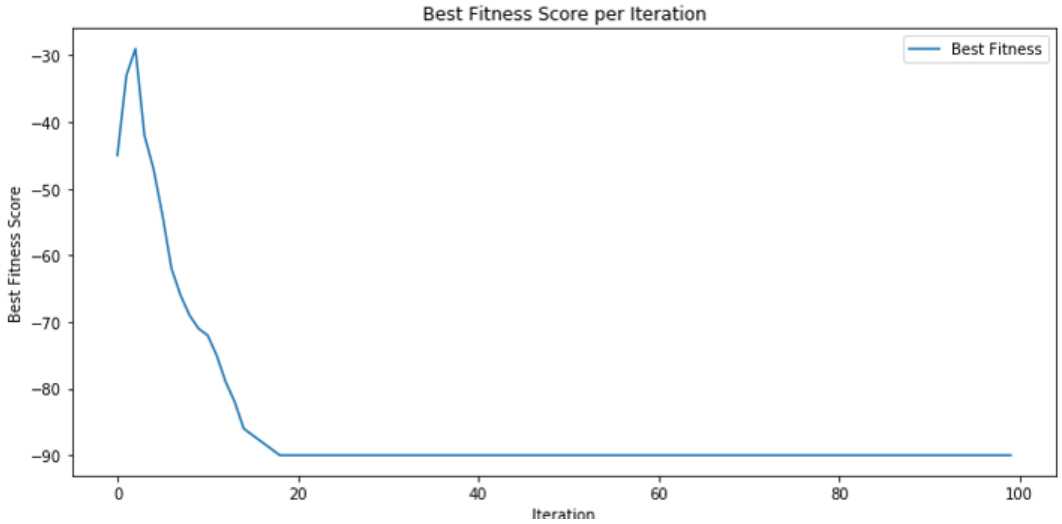

Figure 8. Illustration of the best fitness score per iteration

Figure 8 shows an illustration of the best fitness, which is referred to as the fitness of the best individual in the recent population. Moreover, Figure 8 defines the relationships among the best fitness scores for 100 iterations. Initially, the iteration starts with the high point and decreases correspondingly with an increase in the number of iterations. In addition, Figure 9 defines the population diversity for the iterations.

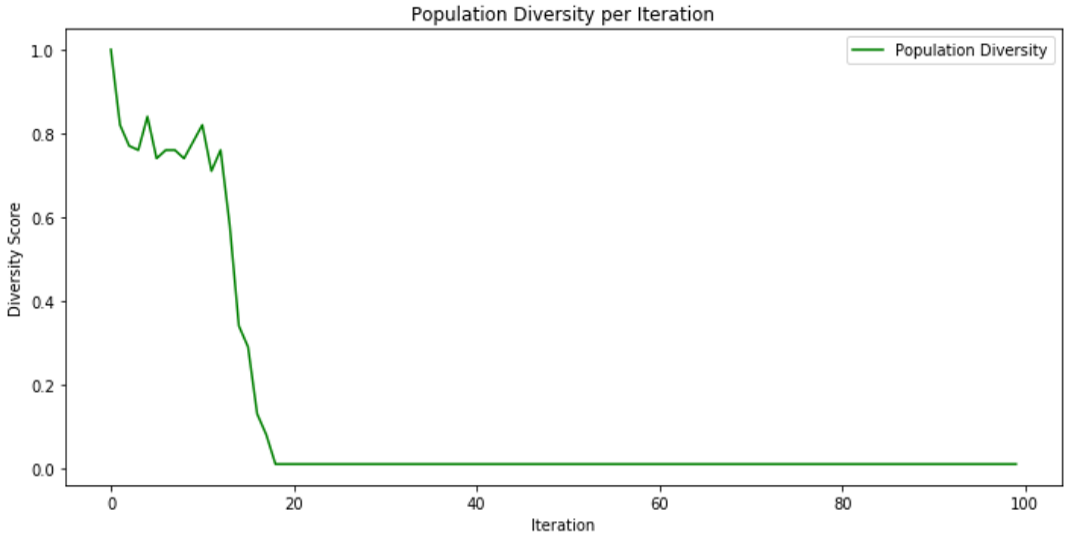

Figure 9. Representation of population diversity per iteration

Figure 9 shows the population diversity per iteration. The diversity of the GA is determined by the performance of the GA. In addition, the diversity is defined as follows: if the average distance among the individuals remains high, then the diversity is high. Similarly, if the average distance remains small, then there is a reduction in diversity. Hence, the diversity score decreases gradually from 1 to 0 at the 20th iteration, and it remains constant up to the 100th iteration.

4.2 Performance Metrics

Each individual within the population produced by the probability of discovering a reorganizing-based genetic algorithm represents a feasible solution to the problem. The fitness evaluation function is used to measure the efficacy of solutions. Preventing getting stuck at a less than ideal solution is important for achieving the best result. Different requirements can lead to the creation of multiple fitness functions. This paper examines the total time taken to complete the task as well as the extent of the load distribution. Equation (6) and formula (7) define the fitness function.

$$GV\ alue = \omega_1 * \frac{1}{completion\ time} + \omega_2 * DBL \quad (6)$$

$$\omega_1 + \omega_2 = 1(0 \leq \omega_1, \omega_2 \leq 1) \quad (7)$$

where $\omega_1\ and\ \omega_2$ are the weight coefficients and DBL is the load degree of load balancing.

4.3 Performance analysis

This section explains the performance attained by the proposed model with the various algorithms in detail.

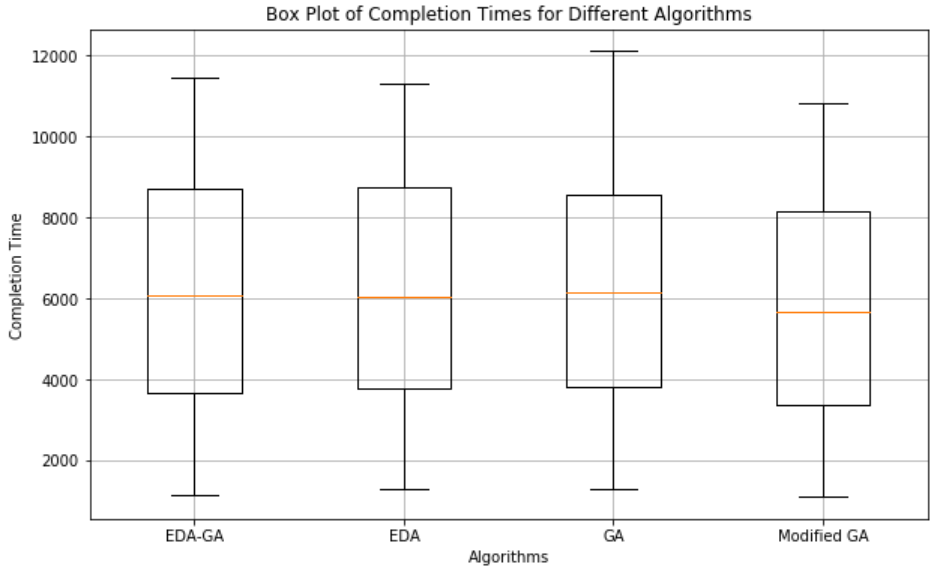

Figure 10. Box plot of the completion times of various algorithms

Figure 10 shows box plots for different algorithms, such as the EDA-GA, EDA, GA and modified GA. Moreover, the modified GA has attained less completion compared with other algorithms. From this, it is inferred that the proposed modified GA can provide effective results because it consumes less completion time.

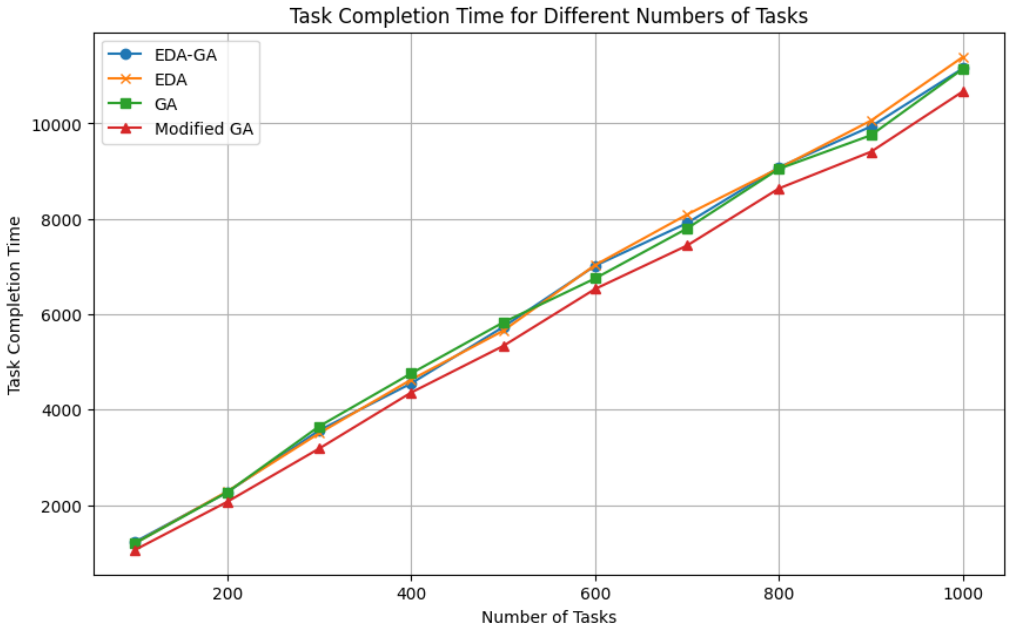

Figure 11. Graphical representation of task completion times for various algorithms

Figure 11 shows the relationships between the modified GA and the other algorithms. Thus, the completion time of the modified GA is shorter than that of the other algorithms. For 1000 tasks, 10841 tasks are obtained, which is less than the number of tasks considered. Here, it is observed that Modified GA has shown a better performance in task sizes, constantly obtaining the lowest completion times. This is because of the enhancement in genetic operations (selection, crossover, mutation) or through problem -specific optimizations, which made it more stable and effective. Further, the GA shows the moderate performance, while it lacks in probabilistic guidance of EDA and impacts in slower convergence with greater completion time, particularly at increase in tasks. Moreover, the EDA shows the least performance, although it has utilized the probabilistic models for generating results, the computational cost of these model can results in increase in task size, which may cause high completion time. Finally, EDA-GA shows the better performance than GA but not with Modified GA. Where, the combination of EDA and GA demonstrates the probabilistic guidance to search and for increasing the speed. But, the complexity

increase certain overhead and making its performance less that Modified GA. From this, it is demonstrated that the Modified GA has attained less completion time. The completion times taken by the EDA and proposed model are tabulated below.

Table 1. Completion Times Taken by the EDA and the Proposed Model

No. Of. Tasks	EDA Completion Time	Proposed Completion Time
100	1282	1108
200	2188	1976
300	3589	3159
400	4367	4050
500	5557	5218
600	6541	6139
700	7933	7531
800	9025	8346
900	1012	9790
1000	11306	10841

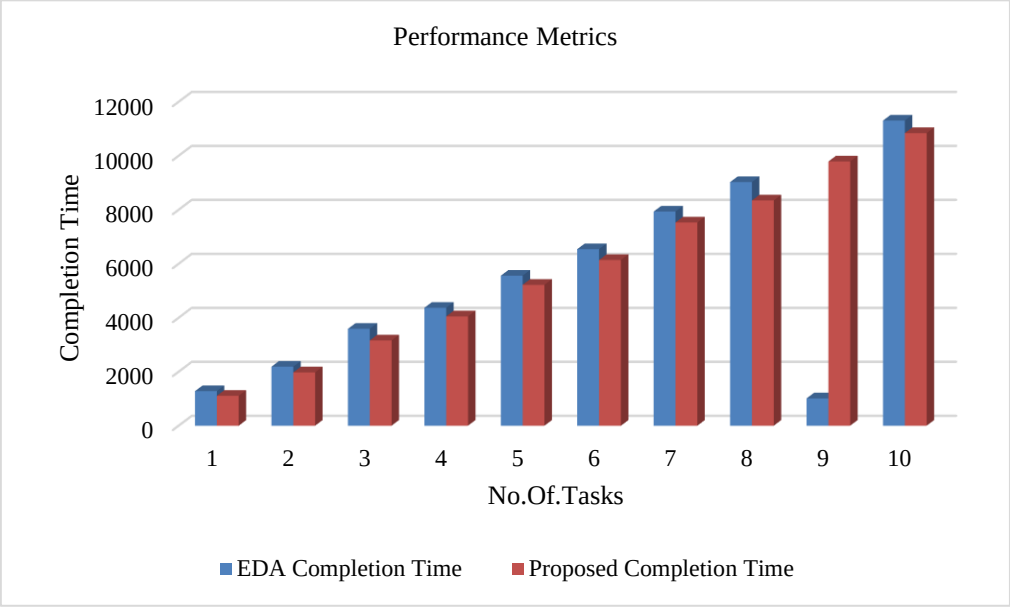

Figure 12. Illustration of the completion time of the EDA with the proposed model

Figure 12 and Table 1 show the completion times of the EDA and the proposed model, which reveal that if the number of tasks increases, the completion time also increases for both algorithms. However, the proposed model results in a shorter completion time for more tasks. Hence, the proposed model is more efficient than the existing EDA model. Therefore, the proposed Modified GA determines a high performance for tasks from 1 to 8 that ranges from smaller to moderate sizes, because of the optimization over its effective heuristic improvement. It reduces the computational complexity and the acceleration process. While EDA shows a continuous increase in the completion time with increasing tasks because of computational overhead related with probabilistic models. Moreover, task 9 and 10 also shows that the proposed model has performed well when handling larger task size. This demonstrates that the proposed algorithm exhibits a sable scalability.

The tabulation shows the completion time for the GA.

Table 2. Completion Times of the Proposed Model with the GA

No . Of . Tasks	GA Completion Time	Proposed Completion Time
100	1309	1108
200	2270	1976
300	3531	3159
400	4719	4050
500	5521	5218
600	6776	6139
700	8106	7531
800	8711	8346
900	10818	9790
1000	12110	10841

Figure 13. Graphical representation of the GA with the proposed completion time

Table 2 and Figure 13 describe the efficiency of the proposed model in terms of the results obtained via the GA and the proposed model. Here, the completion time decreases if the number of tasks increases and the number of iterations decreases for the proposed and GA models. Moreover, GA has performed moderately, because of its less efficacy without heuristic configurations and involves more generations or with large population size for finding the solution. It is observed that the scalability is limited, as computational cost rises more rapidly for large tasks. It is because of lessen guided search process with greater costs that are related with the computation of each generation.

Table 3. Completion Times of the EDA-GA with the Proposed Model

No. Of. Tasks	EDA-GA Completion Time	Proposed Completion Time
100	1154	1108

200	2154	1976
300	3448	3159
400	4270	4050
500	5639	5218
600	6515	6139
700	8138	7531
800	8898	8346
900	10179	9790
1000	11445	10841

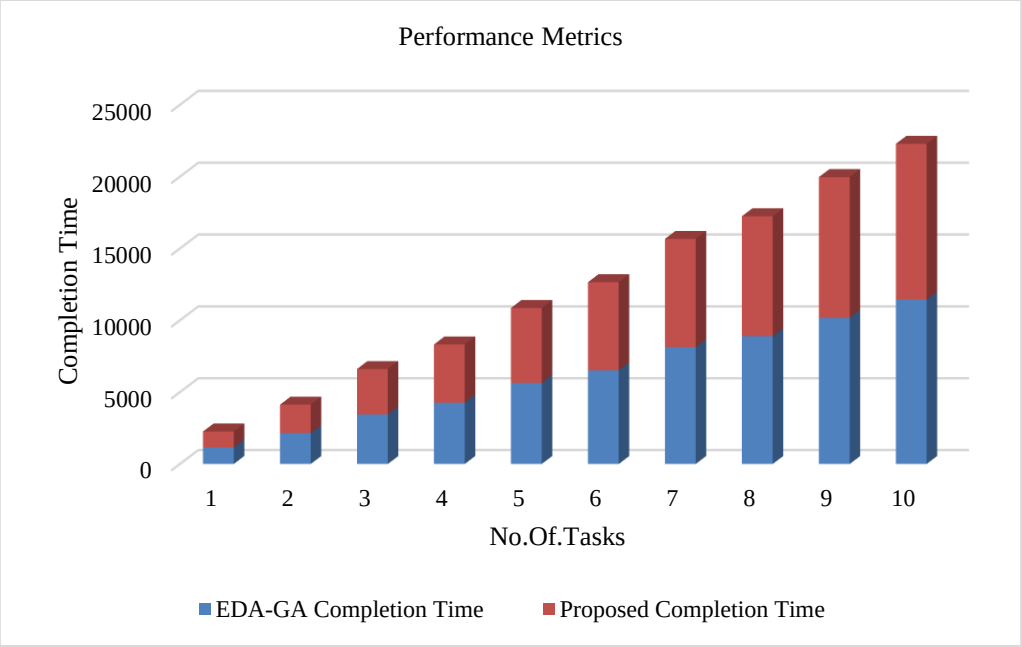

Figure 14. Completion Times of the EDA-GA with the Proposed Model

Table 3 and Figure 14 show the completion times of the EDA-GA and the proposed model. In addition, the results reveal that the proposed model has attained less completion time for more tasks. Hence, the proposed model has provided a betterment in balancing load along with the optimization of resource allocation, assist in mitigating the effect of high complexity under cloud computing specifically. Conversely, conventional methods such as GA or EDA may struggle with these circumstances because of slower convergence rates or a tendency to confine the local optima. Therefore, through large problem sizes or high dynamic conditions, these algorithms provides more iterations for finding optimal solutions that increases the completion times.

4.4 Comparative Analysis

This section compares the completion times of the EDA, GA, EDA-GA and proposed model, and the results are tabulated below.

Table 4. Comparison of the completion times of the proposed model with those of other algorithms

No. Of. Tasks	EDA Completion Time	GA Completion Time	EDA-GA Completion Time	Proposed Completion Time
100	1282	1309	1154	1108
200	2188	2270	2154	1976

300	3589	3531	3448	3159
400	4367	4719	4270	4050
500	5557	5521	5639	5218
600	6541	6776	6515	6139
700	7933	8106	8138	7531
800	9025	8711	8898	8346
900	1012	10818	10179	9790
1000	11306	12110	11445	10841

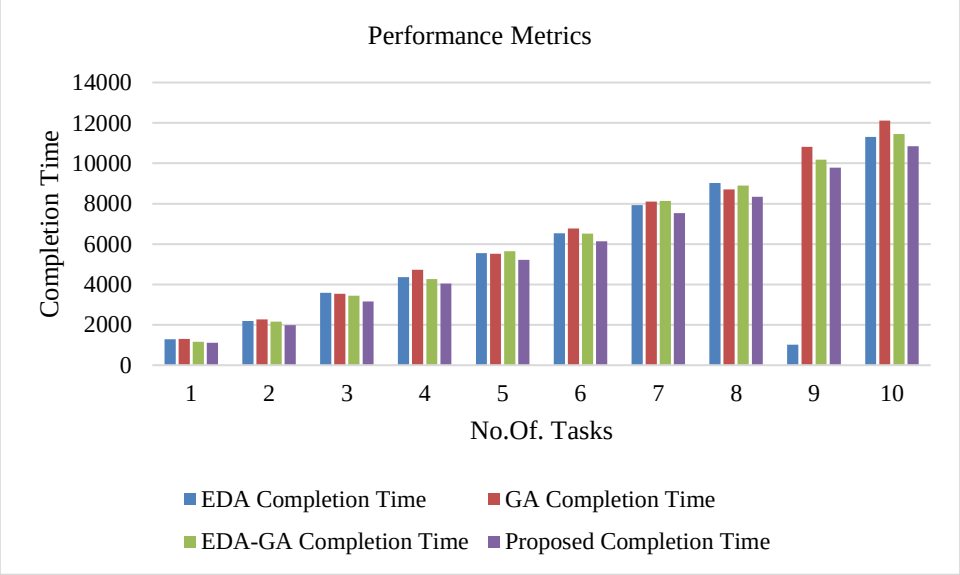

Figure 15. Comparison of the completion times of different algorithms with those of the proposed algorithm

Figure 15 and Table 4 explain the completion times of the EDA, GA, EDA-GA and proposed model. Here, the same number of tasks is taken by all the algorithms and executed. However, the results indicate that the proposed model requires less completion time than the other algorithms do. Moreover, the statistical test P values are evaluated for 100, 500 and 1000 tasks and it is given in table.5.

Statistical tests P values

Table 5. Performance of the Proposed Model for Statistical tests P values

Metric	Value (100 Tasks)	Value (500 Tasks)	Value (1000 Tasks)
Number of Tasks	100	500	1000
ANOVA Test p-value	**0.00033**	**0.00024**	**0.00288**
Significant Difference	Yes	Yes	Yes

Table.5 demonstrates that all three scenarios (100, 500, and 1000 tasks) show statistically significant differences between the groups, a difference clearly indicated by the ANOVA test, because each p-value is much lower than the cut-off level of 0.05. The tasks ordered 100, 500, and 1000, the p-values show a very low chance of finding such differences by chance, which means the null hypothesis, which says that all group means are the same, is rejected. While the smaller p-values of 100 and 500 tasks suggest stronger evidence against the null hypothesis in comparison to 1000 tasks, it might be an artifact of different variances within the groups, and not necessarily a smaller effect size, as a larger sample size can decrease the p-value even for a small effect. Although the ANOVA test shows that group means differ, it does not indicate which groups specifically differ, and further post-hoc analysis would be required to identify such differences.

Ethical Limitations

- *Objectivity and Bias*: For optimizing the particular objectives might unintentionally acquaint with bias. However, the fitness function is heavily highlights some characteristics of task or VM types can cause partial resource allocation, possibly producing longer delays or high costs for further tasks. Confirming that “fairness” over different user and tasks particularly under multi-tenant cloud, needs a structure for fitness function and possible multi-objective optimization, which openly incorporates fairness metrics.

- *Transparency with Explainabilty:* Because Genetic Algorithms are probabilistic metaheuristics, it can be difficult to provide a complete explanation for a given scheduling decision. In situations that call for accountability or auditing, where comprehending the reasoning behind resource allocation is essential for compliance or troubleshooting, this "black box" feature may raise ethical questions.

- *Environmental Impact (Energy Consumption):* Although optimizing resource utilization is mentioned in the paper, the current fitness function does not have a direct focus on minimizing energy consumption as one of its main goals. If completion time and load balancing are the only factors taken into account, this could result in schedules that, although effective in execution, could increase the carbon footprint of cloud operations by keeping more resources active or requiring frequent task migrations.

Ensuring data confidentiality, increasing algorithm efficiency, and putting in place real-time adaptive mechanisms to preserve performance and security are all necessary to address these.

5. Conclusion

In cloud computing, the use of a probabilistic GA for task scheduling has proven to be an efficient approach for resource management under dynamic conditions. In addition, the method uses GA concepts, such as selection, crossover, and mutation, to enhance task assignment to virtual machines, ultimately minimizing time and expense while maximizing resource utilization. The algorithm experienced extensive challenges with the Simpy toolkit, which is a reliable simulation framework for cloud environments. Research indicates that the updated scheduling method outperforms traditional practices by offering improved efficiency and flexibility in handling various workloads and increasing resource availability. Additionally, this research highlights the importance of achieving an equilibrium among different objectives, such as cost, time, and resource effectiveness, throughout the scheduling process. By effectively examining potential solutions, the GA using probabilities not only improves task completion but also ensures the best use of cloud resources, ultimately increasing the efficiency of cloud computing services. Future research potential includes exploring the integration of further optimization methods and modifying the algorithm to accommodate emerging cloud technologies and user requirements, enhancing its applicability and efficacy in real-world scenarios. Furthermore, more parameters are included to enhance the task scheduling algorithm in the presented model. Moreover, Deep Reinforcement Learning agents will be used for robust task scheduler for green cloud computing, where it learns the optimal policies of edge and fog environments. It includes energy consumption and carbon footprint, combined with standard metrics for multi-objective optimization and its performance is evaluated by real-time workloads and large-scale simulations. Hence, the proposed study greatly improves the realm of cloud computing by paving the way for more sophisticated and efficient task scheduling techniques.

Declaration

Conflict of Interest

There is no conflict of interest.

Funding Support

There is no funding support for this study.

Data Availability Statement

There is no research data outside the submitted manuscript file.

References

[1] A. Nag *et al.*, "Exploring the applications and security threats of Internet of Thing in the cloud computing paradigm: A comprehensive study on the cloud of things," *Transactions on Emerging Telecommunications Technologies*, vol. 35, no. 4, p. e4897, 2024.

[2] Y. Li and F. Wu, "Design and Application Research of Embedded Voice Teaching System Based on Cloud Computing," *Wireless Communications and Mobile Computing*, vol. 2023, no. 1, p. 7873715, 2023.
[3] E. Priyanka, S. Thangavel, and X.-Z. Gao, "Review analysis on cloud computing based smart grid technology in the oil pipeline sensor network system," *Petroleum Research*, vol. 6, no. 1, pp. 77-90, 2021.
[4] J. Radianti, T. A. Majchrzak, J. Fromm, and I. Wohlgenannt, "A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda," *Computers & education*, vol. 147, p. 103778, 2020.
[5] A. Ali *et al.*, "An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing," *Sensors*, vol. 21, no. 13, p. 4527, 2021.
[6] H. Yuan, J. Bi, and M. Zhou, "Energy-efficient and QoS-optimized adaptive task scheduling and management in clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1233-1244, 2020.
[7] R. Chen, X. Chen, and C. Yang, "Using a task dependency job-scheduling method to make energy savings in a cloud computing environment," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 4550-4573, 2022.
[8] B. Kruekaew and W. Kimpan, "Multiobjective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803-17818, 2022.
[9] F. Ebadifard and S. M. Babamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment," *Cluster Computing*, vol. 24, pp. 1075-1101, 2021.
[10] A. Semmoud, M. Hakem, B. Benmammar, and J. C. Charr, "Load balancing in cloud computing environments based on adaptive starvation threshold," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 11, p. e5652, 2020.
[11] M. Sulaiman, Z. Halim, M. Lebbah, M. Waqas, and S. Tu, "An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment," *Journal of Grid Computing*, vol. 19, pp. 1-31, 2021.
[12] M. H. Kashani, "Task assignment in distributed systems based on PSO approach," *arXiv preprint arXiv:2112.00053*, 2021.
[13] S. K. Roy, R. Devaraj, A. Sarkar, and D. Senapati, "SLAQA: Quality-level aware scheduling of task graphs on heterogeneous distributed systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5, pp. 1-31, 2021.
[14] K. Salimifard and S. Bigharaz, "The multicommodity network flow problem: state of the art classification, applications, and solution methods," *Operational Research*, vol. 22, no. 1, pp. 1-47, 2022.
[15] S. Pal, N. Jhanjhi, A. S. Abdulbaqi, D. Akila, F. S. Alsubaei, and A. A. Almazroi, "An intelligent task scheduling model for hybrid internet of things and cloud environment for big data applications," *Sustainability*, vol. 15, no. 6, p. 5104, 2023.
[16] X. Xia, H. Qiu, X. Xu, and Y. Zhang, "Multiobjective workflow scheduling based on genetic algorithm in cloud environment," *Information Sciences*, vol. 606, pp. 38-59, 2022.
[17] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Computer Communications*, vol. 187, pp. 35-44, 2022.
[18] F. A. Emara, A. Gad-Elrab, A. Sobhi, and K. Raslan, "Genetic-Based Multiobjective Task Scheduling Algorithm in Cloud Computing Environment," *International Journal of Intelligent Engineering & Systems*, vol. 14, no. 5, 2021.
[19] G. Weiqing and C. Yanru, "Task-scheduling algorithm based on improved genetic algorithm in cloud computing environment," *Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering)*, vol. 14, no. 1, pp. 13-19, 2021.
[20] S. Velliangiri, P. Karthikeyan, V. A. Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 631-639, 2021.
[21] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Computing and Applications*, vol. 32, pp. 1531-1541, 2020.
[22] R. Gulbaz, A. B. Siddiqui, N. Anjum, A. A. Alotaibi, T. Althobaiti, and N. Ramzan, "Balancer genetic algorithm—A novel task scheduling optimization approach in cloud computing," *Applied Sciences*, vol. 11, no. 14, p. 6244, 2021.

[23] Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, "Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework," *Mathematical Problems in Engineering*, vol. 2022, no. 1, p. 4290382, 2022.
[24] M. Tanha, M. Hosseini Shirvani, and A. M. Rahmani, "A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments," *Neural Computing and Applications*, vol. 33, pp. 16951-16984, 2021.
[25] M. N. Aktan and H. Bulut, "Metaheuristic task scheduling algorithms for cloud computing environments," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 9, p. e6513, 2022.
[26] L. Imene, S. Sihem, K. Okba, and B. Mohamed, "A third generation genetic algorithm NSGAIII for task scheduling in cloud computing," *Journal of King Saud university-computer and information sciences*, vol. 34, no. 9, pp. 7515-7529, 2022.
[27] K. Malathi and K. Priyadarsini, "Hybrid lion–GA optimization algorithm-based task scheduling approach in cloud computing," *Applied Nanoscience*, vol. 13, no. 3, pp. 2601-2610, 2023.
[28] L. Abualigah and M. Alkhrabsheh, "Amended hybrid multiverse optimizer with genetic algorithm for solving task scheduling problem in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 740-765, 2022.
[29] N. S. Soulegan, B. Barekatain, and B. S. Neysiani, "MTC: minimizing time and cost of cloud task scheduling based on customers and providers needs using genetic algorithm," *International Journal of Intelligent Systems and Applications*, vol. 15, no. 2, p. 38, 2021.
[30] Z. Nan, L. Wenjing, L. Zhu, L. Zhi, L. Yumin, and N. Nahar, "A New Task Scheduling Scheme Based on Genetic Algorithm for Edge Computing," *Computers, Materials & Continua*, vol. 71, no. 1, 2022.
[31] S. Liu and N. Wang, "Collaborative optimization scheduling of cloud service resources based on improved genetic algorithm," *IEEE Access*, vol. 8, pp. 150878-150890, 2020.
[32] K. Shao, H. Fu, and B. Wang, "An efficient combination of genetic algorithm and particle swarm optimization for scheduling data-intensive tasks in heterogeneous cloud computing," *Electronics*, vol. 12, no. 16, p. 3450, 2023.
[33] G. Agarwal, S. Gupta, R. Ahuja, and A. K. Rai, "Multiprocessor task scheduling using multiobjective hybrid genetic Algorithm in Fog–cloud computing," *Knowledge-Based Systems*, vol. 272, p. 110563, 2023.
[34] A. A. Hussain and F. Al-Turjman, "Hybrid Genetic Algorithm for IOMT-Cloud Task Scheduling," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 6604286, 2022.
[35] S. Kodli and S. Terdal, "Hybrid Max-Min Genetic Algorithm for Load Balancing and Task Scheduling in Cloud Environment," *International Journal of Intelligent Engineering & Systems*, vol. 14, no. 1, 2021.
[36] M. Sardaraz and M. Tahir, "A parallel multiobjective genetic algorithm for scheduling scientific workflows in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 16, no. 8, p. 1550147720949142, 2020.
[37] S. Lipsa, R. K. Dash, N. Ivković, and K. Cengiz, "Task scheduling in cloud computing: A priority-based heuristic approach," *IEEE access*, vol. 11, pp. 27111-27126, 2023.
[38] A. S. Abohamama, A. El-Ghamry, and E. Hamouda, "Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment," *Journal of Network and Systems Management*, vol. 30, no. 4, p. 54, 2022.
[39] N. K. Walia *et al.*, "An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments," *IEEE Access*, vol. 9, pp. 117325-117337, 2021.
[40] A. A. Al-Habob, O. A. Dobre, A. G. Armada, and S. Muhaidat, "Task scheduling for mobile edge computing using genetic algorithm and conflict graphs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8805-8819, 2020.