Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6125246, 22 pages
https://doi.org/10.1155/2022/6125246

Research Article

Load Balancing in Edge Computing Using Integer Linear Programming Based Genetic Algorithm and Multilevel Control Approach

Rui Zhang,[1] **Hong Shu**,[1] **and Yahya Dorostkar Navaei**[2]

[1]*Mathematics and Information Science College of Guiyang University, Guiyang 550005, China*
[2]*Department of Computer and Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

Correspondence should be addressed to Hong Shu; yrj999999@sina.com and Yahya Dorostkar Navaei; y.dorostkar@qiau.ac.ir

Received 19 December 2021; Revised 22 April 2022; Accepted 16 June 2022; Published 30 June 2022

Academic Editor: Saba Bashir

Copyright © 2022 Rui Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Due to the proliferation of requests in heterogeneous resources in edge computing, the existence of a large number of tasks and workloads in virtual machines in the edge computing environment is inevitable. Thus, load balancing strives to facilitate an even distribution of workload across available resources. Its purpose is to provide continuous service and to ensure fair load distribution among resources. Load balancing, with the aim of minimizing response time for tasks and improving resource efficiency, tries to do the proper mapping of tasks among virtual machines at a lower cost. Flow scheduling, on the other hand, assigns a task (group of tasks) to computational resources by prioritizing tasks, so that the relationship between them is maintained. Therefore, in this research, a hierarchical control framework for load balancing and assignment of tasks in edge computing services is presented in order to create load balancing. In the proposed method, in the first level, the genetic algorithm receives a set of tasks in a workflow. Genetic algorithm prioritizes and assigns tasks to resources according to time constraints, resource processing power, resource availability, and task cost. For this purpose, the integer linear programming optimization in the evaluation function of the genetic algorithm will be utilized. In the second level, considering the past load distribution in edge resources, we estimate the probability of load distribution among sources according to hidden Markov model (HMM). Finally, in order to optimally map tasks to the virtual machines in each host, we will use game theory with service quality factors as an evaluation function. Previous methods have provided a hierarchical control framework that aims to achieve conflicting goals within a data center, but does not use linear programming. Considering the use of service quality criteria as evaluation function parameters in heuristic and optimization methods in this research, it is expected that the results of this research will improve compared to previous methods.

1. Introduction

An edge computing model is possibly the most efficient model if its resources are used efficiently, and this can be achieved by applying and maintaining proper management of edge resources [1]. Resource management is achieved by adopting strong resource scheduling, efficient allocations, and powerful scalability techniques. These resources are provided to customers through a process called virtualization of a software component, hardware, or both, in the form of virtual machines (VMs) [2]. The biggest advantage of edge computing is that a physical machine becomes a merely multipurpose virtual machine for a user [3–5]. The cloud service provider (CSP) plays an important role in providing services to users, and considering the availability of virtual resources, assigning a task becomes really complex. While submitting user requests, some VMs experience heavy traffic on user tasks, and some of them experience less traffic. As a result, the edge service provider has to deal with unbalanced machines with large differences in user tasks and resource usage [6–9]. The problem with load imbalance is an adverse event on the CSP side that degrades the performance and efficiency of computing resources along with the quality of service (QoS) assurance in the service level agreement (SLA) between the consumer and the edge service provider. In this situation, there is a need for load balancing that has become a strange yet interesting topic among researchers. Load balancing in edge computing can be used at the physical device level as well as the VM level [10–13]. Load balancing is a process of redistributing workload in a distributed system such as edge computing, which makes sure that no virtual machine is overloaded, while other virtual machines are idle or have less workload. Load balancing tries to accelerate various limiting parameters, such as response time, runtime, and system stability, in order to improve edge performance. This is an optimization method in which scheduling is an NP-hard problem. There are a number of load balancing approaches proposed by researchers, most of which focus on task scheduling, task allocation, resource planning, resource allocation, and resource management [6]. After distributing a balanced load among the hosts of the service provider in the edge environment, there is a need for proper mapping of tasks among virtual machines and scheduling of resources to input tasks in the current host according to different criteria. The process of workflow scheduling refers to the mapping of tasks (a group of tasks) to existing computational resources and the timing of their execution by observing the priorities among tasks so that the relationship between them is maintained. The structure of workflows is often defined as a graph without a circle, like a tree structure. Decisions to map tasks to a resource can be made based on the information contained in a scheduler according to the criteria of the service level agreement (SLA) and the QoS needs of the users [14–17].

Therefore, in order to overcome these challenges, a hierarchical control framework for load balancing and task allocation in edge computing services is presented in this study. In the proposed method, in the first level, a genetic algorithm (GA) has been employed to model the workload [18]. The input of the genetic algorithm in the proposed method includes a set of tasks. Depending on the priorities and the relationship between the tasks, we will model the load distribution among the existing hosts in order to optimize the evaluation criteria.

The evaluation criteria used in this method include time constraints, resource processing power, probability of access to the resource, and the cost of performing tasks in the resource. To optimize the load distribution among resources, due to the existing limitations, the integer linear programming optimization (ILP) in the evaluation function of the genetic algorithm will be used [19]. In the second level, considering the past load distribution in edge resources, we estimate the probability of load distribution among sources according to the hidden Markov model (HMM) [20]. Finally, in order to map the tasks to the virtual machines in each host, we will use the game theory with QoS factors as an evaluation function [21]. Considering the use of QoS criteria as evaluation function parameters in discovery and optimization methods in this research, it is expected that the results of this research will improve compared to previous methods.

The continuation of this article is as follows:

(i) Prioritizing and assigning tasks to resources using integer linear programming based genetic algorithm by considering time constraints, resource processing power, resource availability, and task cost

(ii) Estimating the probability of load distribution among sources according to hidden Markov model (HMM) by considering the past load distribution in edge resources

(iii) Mapping tasks to the virtual machines in each host in optimally manner using game theory with quality of service factors as an evaluation function

(iv) Evaluating makespan, cost, energy, performance, and reliability of proposed method

In the second part, related works will be reviewed. In the third section, the details of the proposed method will be explained. In the fourth section, the implementation and evaluation of the proposed method will be stated. In the fifth section, the conclusion of the article will be stated.

2. Related Works

With the increasing popularity and the advancement of edge computing technology, users are merely trying to access resources that are sufficient to perform the tasks they need and they are only willing to pay for the resources they need. In edge computing, tasks must be distributed in such a way that all available resources have approximately the same number of tasks to execute. One solution that can solve this problem effectively is to schedule tasks with control approaches in order to balance the load. Resource scheduling using a control approach for mapping tasks to virtual machines in the edge is one of the most important issues we face. This resource scheduling approach makes it possible to authorize the execution of a task in a virtual machine in the edge or to migrate a task from one virtual machine to another. The restrictions that the user has on performing tasks on virtual machines must be taken into account during the scheduling based on the desired service quality rules. For instance, tasks may have a specific execution sequence, or virtual machines may be assigned tasks exclusively, and only one task may be executed on the resource at a time, or a specific time limit may be set for the execution of tasks. The importance of load balancing has led to extensive research in this area [22–31]. The following are some of these studies, the main basis of which is timing.

In [32], a job allocation mechanism (JAM) has proposed to reduce battery consumption in the processing of large Internet-physical-social data in mobile edge computing (MCC). In this method battery consumption for processing work has displayed continuously with mobile devices, without an external edge server in a shared architecture MCC environment. In [33], the improved maximum minimum scheduling algorithm (IMMSA) that improves request completion time by using machine learning training as well as requesting size clustering and clustering the productivity percentage of virtual machines has been proposed. In [34], a genetic algorithm (GA-)-based optimization technique in the sensor mobile edge computing environment to discern the optimal solution has been proposed. In [35], an improved elitism genetic algorithm (IEGA) for overcoming the task scheduling problem for FC to enhance the quality of services to users of IoT devices has been suggested. In [36], the two-level load balancing approach in fog computing environment as a multiobjective optimization problem using the elitism-based genetic algorithm (EGA) for minimizing the service time, cost, and energy consumption and thus ensuring the QoS of IoT applications has been presented. In [37], a processing model for the load balancing problem using NSGAII algorithm has presented in which a trade-off between energy consumption and delay in processing workloads in fog has formulated. In [38], an incentive-based bargaining approach which encourages the fog nodes to cooperate among themselves by receiving incentives from the end users benefitting from the cooperation has been proposed. In [39], the hidden Markov chain learning method has been used to cope with this challenge in the IoT ecosystem integrated with the fog computing, to determine the probability of the need for each thing or resource in the near future with the aim of reducing latency and increasing the network use. In [40], a computational model as a game that considers energy consumption and transmission latency as decision parameters for task offloading of IoT applications has been proposed. In [41], an offload and migration-enabled smart gateway for Cloud of Things approach has proposed that employs noncooperative game theory to offload, schedule, and reschedule the computational tasks effectively in the fog-cloud environment.

3. Proposed Method

The proposed method aims to provide a hierarchical control framework for the optimal allocation of resources to tasks in order to balance the work load in edge computing center. In this method, the ultimate goal is to provide services that meet QoS needs only by using the necessary resources in the edge infrastructure. In the proposed method, in general, QoS requirements are determined based on resource efficiency, total time of tasks in the edge environment, and energy consumption of resources while performing tasks and cost of performing tasks on resources, which may be different according to task priorities. Therefore, allocating effective and efficient resources contributes to increase in productivity and reduction in completion time, energy, and costs. Edge computing is a solution recently adopted in order to provide services that host tasks in virtual environments. Physical resources are divided into several virtual machines, and these virtual machines are responsible for assigning tasks that work with parts of the capacity of the physical system. Automated management techniques are implemented by network controllers that can control the set of programs run by each server, the volume of requests on different servers, and the capacity allocated to run each program on each server. Therefore, in this research, a two-tier control architecture is presented that deals with the issue of load balance with optimal allocation of resources to tasks as well as controlling task acceptance according to QoS criteria. The general architecture of the proposed method is shown in Figure 1.

As shown in Figure 1, the proposed method provides a two-tier hierarchical control framework for load balancing and task allocation in edge computing services. In the first level of the proposed method, genetic algorithm has been employed to model the workload [18]. The input of the genetic algorithm in the proposed method includes a set of tasks; then, according to the priorities and the relationship between the tasks, we will model the load distribution among the existing hosts in order to optimize the evaluation criteria. The evaluation criteria used in this method include time constraints, resource processing power, probability of access to the resource, and cost of performing tasks in the resource. Due to the existing limitations, to optimize the load distribution among sources, the correct linear programming optimization in the evaluation function of the genetic algorithm will be employed [19]. In the second level, according to the history of load distribution in edge resources, we estimate the probability of load distribution among sources using hidden Markov model [20]. Finally, in order to map the tasks to the virtual machines in each host, we will use the game theory with QoS factors as an evaluation function [21]. In the continuation of this chapter, we will describe the different parts of the proposed method in more detail.

3.1. Genetic Algorithm Based on Integer Linear Programming Approach. In the proposed method, a genetic algorithm based on the integer linear programming (ILP) approach with respect to the dynamic nature of the edge environment is presented. The proposed genetic algorithm looks for an optimal (or near-optimal) solution that starts from the initial population and is generated using the selection of the most appropriate chromosomes based on fitting and mutation operators. In addition to proper encoding for chromosomes as well as proper design for fitting and mutation operators, taking the limitations of the problem that should be consistent with the nature of the optimization problem into account, the proposed genetic algorithm is equipped with a mechanism that examines the feasibility of chromosomes and if the chromosome does not meet the expectations, there is a penalty and the value of the fitness function reduces in order to reduce its selection and survival chances in the next generation. The proposed genetic algorithm aims to look for a solution for scheduling new tasks entering the edge environment using the previous solution for the problem, which allows it to work according to the dynamic scheduling technique.

3.1.1. Chromosome Encoding. The proposed genetic algorithm finds an initial set of tasks as input and begins to generate the initial population of chromosomes, each of which represents a possible solution to the optimization problem. Thus, a chromosome is a vector for a gene whose numbers within each gene indicate the source number and possible resource allocation to tasks and a potential scheduling. Figure 2 shows an example of the chromosomes presented in the proposed method.

As shown in Figure 2, the chromosomes in the proposed method include vector genes, and the number of these genes is equal to the number of tasks in the edge environment. In the proposed method, considering the fact that the genetic algorithm uses the previous schedules to present new chromosomes, the number of received tasks and resources are

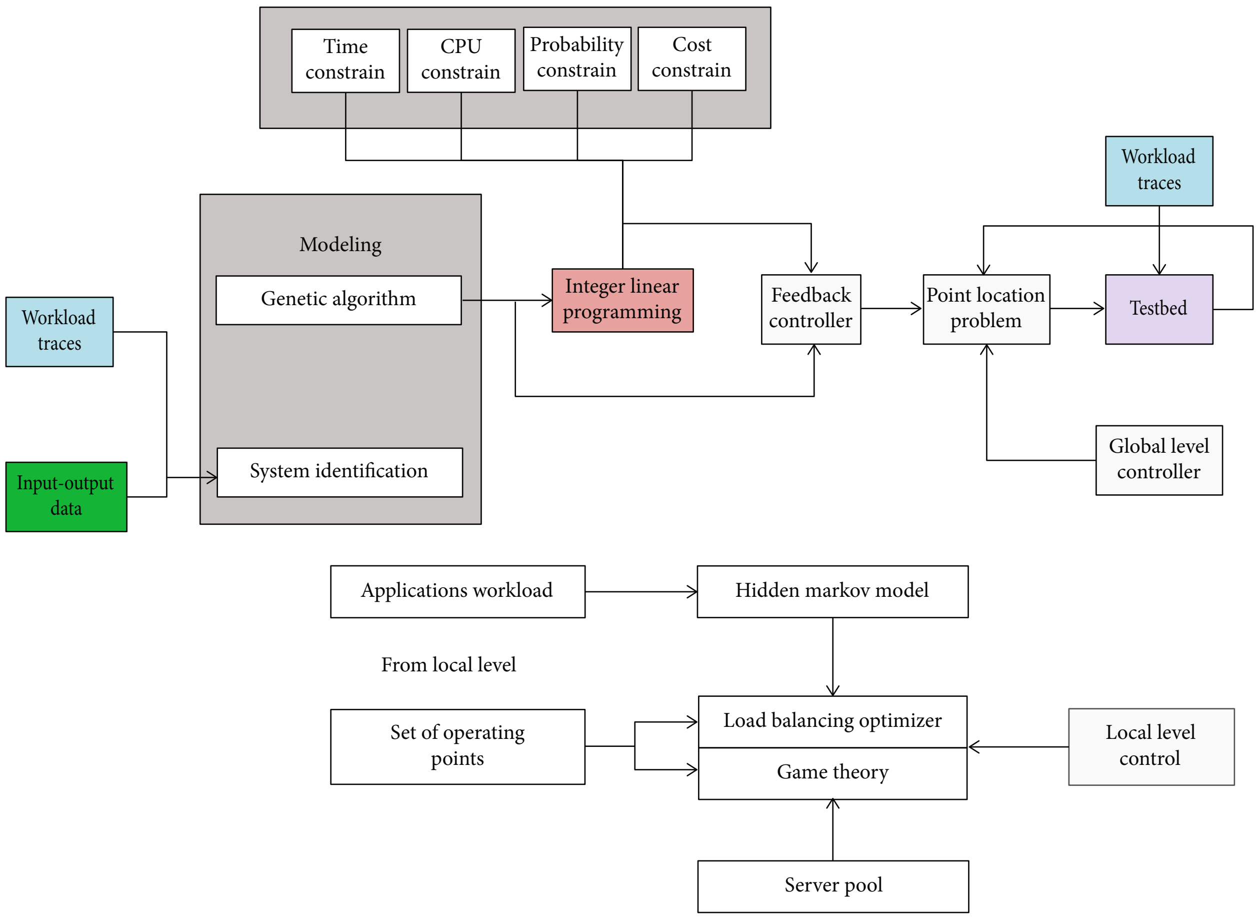


Figure 1: General architecture of the proposed method.

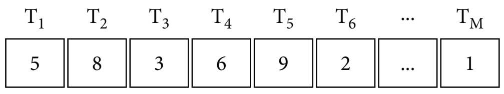


Figure 2: Representation of a chromosome in the proposed method.

the same so that in case of any problem for any of the resources, it will not violate error tolerance in the method. The numbers listed in each of the genes serve as a possible source for the task at hand. Initially, each chromosome is considered as a probabilistic scheduling that changes with the application of the fitness function and mutation operators, and finally, the chromosome with the highest amount of proportion is selected as the near-optimal scheduling.

3.1.2. Fitness Function. The fitness function for each chromosome is determined according to the objective function, which in the proposed method is a combination of four parameters: completion time, cost, resource efficiency, and probability of allocation. For each possible scheduling (chromosome) of the population, the fitness function is calculated while performing timed tasks on the chromosome at a particular time. The fit function represents the desired parameters such as:

(i) the amount of time required to complete tasks in resources

(ii) the cost required to perform tasks

(iii) the resource efficiency rate

(iv) the probability of assigning tasks to resources

These parameters are calculated in order not to accumulate charge in a source when performing tasks scheduled on the chromosome at a specific time. Table 1 illustrates the notation for the fit function in the proposed method.

The fitness function for calculating the given chromosomes is calculated based on Equation (1) using the objective function of the proposed model, which is optimized according to the integer linear programming.

$$\min \left(\sum_{i=1}^{M} \sum_{q=1}^{Q} X_{E_{q i}} P_i \left(C_{E_{R_i^N R_q^N}} + C_{E_{R_i^S R_q^s}} + C_{E_{R_i^C R_q^C}} \right) T_E \right.$$
$$\left. + \sum_{j=1}^{D} \sum_{f=1}^{F} X_{Mig_{q i}} (1 - P_i) \left(C_{T_{R_j^B R_f^B}} + C_{D_{R_j^B R_f^B}} \right) T_T \right)$$

Table 1: Notations of the proposed model.

Notation	Explanation
C_E	Cost of each task performance
C_T	Cost of each task migration
C_D	Cost of data transfer
T_E	Task execution time
T_T	Transfer time and task migration
T_I	System idle time
T_D	Duty deadline
X_E	Task performance decision
X_{Mig}	Task migration decision
D	The amount of data transferred between virtual machines
Q	Request (task)
F	Requests for data transfer (migration)
M	Virtual machines
P	Probability of allocation
R_N	Cost according to the number of source cores
R_S	Cost according to processing speed
R_C	Cost according to memory capacity (cache)
R_B	Cost according to network bandwidth

Subject to

$$\begin{gathered}
\sum_{q=1}^{Q} X_{E_q}\left(C_{E_{R_q^N}}\right) \le C_{E_N} \\
\sum_{q=1}^{Q} X_{E_q}\left(C_{E_{R_q^S}}\right) \le C_{E_S} \\
\sum_{q=1}^{Q} X_{E_q}\left(C_{E_{R_q^C}}\right) \le C_{E_C} \\
\sum_{q=1}^{Q} X_{Mig_q}\left(C_{E_{R_q^C}}\right) \le C_{E_B} \\
\sum_{q=1}^{Q} X_{Mig_q}\left(C_{D_{R_q^B}}\right) \le C_{D_B} \\
\sum_{i=1}^{M} T_{E_i} + T_{T_i} + T_{I_i} \le T_{D_i} \\
\sum_{q=1}^{Q} X_{E_q} \le M \\
\sum_{i=1}^{F} Q_i(1-P_i) \le D_i \\
\sum_{q=1}^{Q} P_i = 1 \\
X_{E_q}, X_{Mig_q} \in \{0,1\}, q = 1, 2, \cdots, Q \qquad (1)
\end{gathered}$$

According to the fitness function presented in Equation (1), the appropriate chromosomes are selected from the original population, and the rest of the chromosomes are sent to the fitting and mutation operator in order to diversify the population and produce new superior chromosomes. Each chromosome in the new offspring population is examined to determine whether it is a possible solution to the problem; that is, it minimizes the fitness function and meets the given constraint demands. Impossible chromosomes that violate existing constraints are fined according to the value of the fitness function, so they are less likely to be chosen to reproduce and become new chromosomes. The most appropriate chromosome, which represents a near-optimal scheduling solution, is maintained after each replication step and then sorted by optimality. This process is repeated until the termination condition is met.

3.1.3. Crossover Operator. The crossover operator plays an important role for the genetic algorithm to diversify the population and produce new chromosomes. To increase the scope of the search and consequently obtain more possible public solutions, it is necessary for the genetic algorithm to perform the fitting process between two chromosomes (parents) and produce new offspring as a new population. The crossover operator is a random replacement of a number of genes on the first and second chromosomes. The parameter that is important in the crossover operator is called the fitting probability or P-crossover, and regarding the random fitting operator, it can be defined as follows:

$$\mathrm{P-crossover} = \mathrm{round}(k * (G_{\max} - G_{\min})), k \text{ is } a \text{ rand in } (0, 1), \quad (2)$$

where the $G_{\max}$ parameter is the maximum number of genes on the chromosome and the $G_{\min}$ parameter is the minimum number of genes on the chromosome. The parameter k is considered as a random value in the range of zero and one. The value of the P-crossover parameter is considered as the part of the chromosome that must be exchanged between the first chromosome and the lower chromosome, and the initial and final intersection of this part is called the fitting point. The fitting point may first be selected from the beginning of the chromosome or from any other desired location on it, and the last fitting point is added to the value of the P-crossover parameter. Figure 2 shows the fitting operator.

As shown in Figure 3, during crossover operation, the genes present in P-crossover are swapped on two chromosomes. The part of the genes of the first chromosome that is in the P-crossover is transferred with the same number of genes in the second chromosome so that the new chromosomes have more variety than the previous chromosomes. Chromosome switching is end-to-end so that gene i from the first chromosome crossed with gene i from the second chromosome. Thus, in the new generation, n new chromosomes are produced which are added to the previous chromosomes, and the population after the crossover will be equal to 2n. In this paper, each of the Ti genes represents a task, and Ri represents the value of the gene that indicates the source number for each of the solutions.

3.1.4. Mutation Operator. The mutation operator plays a key role in generating new populations and diversifying chromosomes and is as important as the fitting operator. This operator, like the mutation operator, is able to increase the scope of search as well as introducing possible solutions and producing new children as a new population. In this operator, the probability parameter is of great importance, which is considered as the point of the chromosome that must mutate. The P-mutate parameter or the probability of mutation can be calculated as follows:

$$\mathrm{P-mutate} = \mathrm{round}(k * (G_{\max} - G_{\min})), k \text{ is } a \text{ rand in } [0, 1], \quad (3)$$

where the P-mutate parameter indicates the probability of mutation and the k parameter can be used as a value between zero and one, and even zero or one as the first and last gene on the chromosome. The difference between the fitting and the mutation operator is that the fitting operator replaces several genes on one chromosome with genes on another chromosome, but the mutation operator changes the value of one (or more) genes to produce a new chromosome. Figure 3 shows the mutation operator.

As shown in Figure 4, there are two chromosomes with different genes that change separately at the T_2 and T_6 locations. At T_2, the value of gene has changed from R_{12} to R_{15}. Similarly, the value of gene has mutated from R_{16} to R_{13} at T_6. Gene mutations may have good results. Occasionally it can have bad results. Nevertheless, gene mutations are essential for maintaining population diversity.

3.1.5. Selection Operator. After the fitting and mutation operators perform their tasks, among the new population and the chromosomes produced as the next generation, the selection operator selects the chromosomes that have the highest proportionality function or, in other words, have a near-optimal solution to the scheduling problem in order to balance the load in the edge. In this case, the tasks assigned to virtual machines are examined with the maximum total execution time, the cost of performing tasks in resources and resource efficiency, and the possibility of optimal allocation of tasks to resources, in order to balance the load in the edge environment, and in case tasks in the optimal solution need to be migrated, they are transferred from one virtual machine to another considering optimal evaluation criteria. Therefore, optimizing the performance of the task scheduling algorithm in order to balance the load in the edge environment and transferring some tasks to another virtual machine will also help balance the workload.

3.2. Markov Hidden Model. As mentioned, in the proposed method, in order to create a load balance, examining the tasks that have already been assigned to each resource and calculating the probability of assigning tasks to the resources have been determined as one of the evaluation parameters for the fit function in the proposed linear programming model. Resources that have already received more tasks have more potential to upset the balance of load distribution in the edge environment. Therefore, identifying such resources can greatly help to balance the load in the edge environment. In this method, in order to determine the probability of resource distribution, a possible hidden Markov model is used which is described below.

To clarify the general concept of the Markov model process, it can be said that if we divide the time in the Markov model into three periods (past, present, and future), the future time of the system depends not only on its present state but also on this model and its process of determining the probability of the system on the path it has taken in the past. In other words, if the state of the system is known at moments such as $t_1, t_2, \cdots, t_n$, it can be said that to predict the state of the system in the next moment, $t_n + 1$, the state of the system from moment $t1$ to moment tn must be examined [42].

Markov hidden models can be defined as probabilistic models in which a sequence of probabilities is created by two random processes:

The process of moving between states and the process of propagating an output sequence are characterized by Markov property and output independence. The first model is a Markov model created by a finite set of states, which creates a sequence of states of variables and is provided by the initial state probabilities and the probability of transition between state variables. The second model is characterized by the release of a character from the alphabet specified in each mode, with a probability distribution that depends only on the mode. Transfer sequence mode is a hidden process. This means that variable modes cannot be viewed directly but can

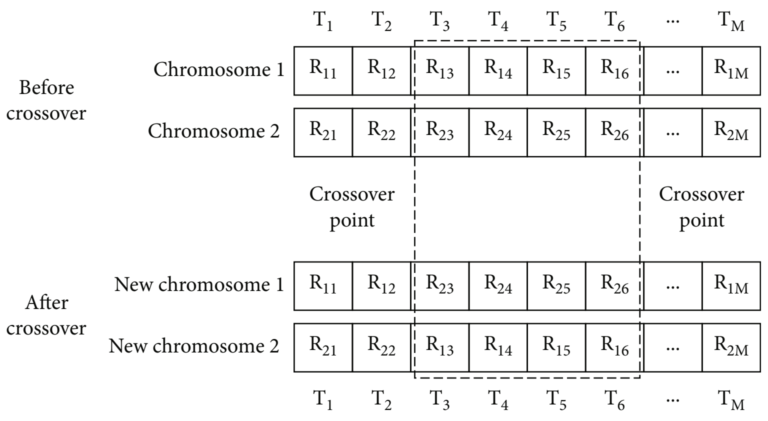


Figure 3: An example of a fitting operator on two chromosomes.

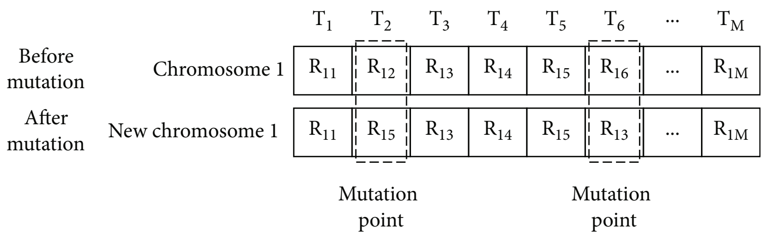


Figure 4: An example of population diversity on chromosomes.

be seen through a sequence of published symbols and that is the reason it is named Markov hidden model. Thus, a hidden Markov model is defined by different states, state probabilities, transition probability between states, propagation probabilities, and initial probabilities, and all these make up the architecture of Markov hidden models. The formal definition of hidden Markov models for the proposed method is based on specified pairs (S, V, p, A, B) with the following elements:

(i) $S = \{S_1, S_2, \cdots, S_N\}$ is a set of states, where N is the number of states. The triple sequence pairs (S, p, A) represent the Markov chain in which the states are hidden and we never see them directly. In the proposed method, the virtual machines inside the hosts are considered as states in the hidden Markov model where the status of each virtual machine is considered hidden, and communication with the hosts containing the virtual machines is established. There may be several virtual machines in each host that provide services independently, but they are generally part of one host, and we can directly observe host states

(ii) $V = \{v_1, V_2, V_{\mathrm{VM}}\}$ are words or a set of symbols that may be published. In the proposed method, tasks are set of symbols that may be propagated and transferred between states. Transferring and migrating tasks between modes are considered as possibilities

(iii) $\pi : S \longrightarrow [0.1] = \{\pi_1, \pi_2, \cdots, \pi_N\}$ is the initial probability distribution in the states. This indicates the probability of beginning the process in a mode. In the proposed method, for each resource, this probability is shown as the number of tasks in the queue for one resource compared to the total number of tasks in the edge environment and is considered as the initial probability distribution in the proposed method. It is therefore expected that:

$$\sum_{s \in S} \pi(s) = \sum_{i=1}^{N} \pi_i = 1. \quad (4)$$

(iv) $A = (a_{ij})_{i \in S, j \in S}$ is the probability of transfer and motion between the S_i and S_j modes. It is expected that for each S_i and S_j‘ $a_{ij} \in [0, 1]$ and for each S_j‘ $\sum_{i \in S} a_{ij} = 1$.

In the proposed method, the migration of tasks between two sources i and j is shown with a_{ij}.

(v) $B = (b_{ij})_{i \in V, j \in S}$ is the probability of propagation if the symbol vi is seen in the state S_j. In the proposed method, the presence of task i in source j is denoted by b_{ij}

Markov hidden models are of great help if you need to model a process in which there is no direct knowledge of the state of the system. The main idea is that Markov hidden model is a “productive” sequel. In general, in this study, we talk about assigning tasks to observable resources since we can use the hidden Markov model as a generating model that could be used to generate observational sequences. Algorithmically, a sequence of assigning tasks to resources $O = o_1, \cdots, o_T$, with $o_t \in V$ can be generated by the hidden Markov model. Two assumptions are made by the model. The first assumption is called Markov and indicates the model memory. It means that the current state depends only on the previous state, and therefore, we have:

$$P\left(q_t | q_1^{t-1}\right) = P(q_t | q_{t-1}). \tag{5}$$

The second assumption is the independence of assigning tasks to resources; i.e., the observation of output at time t depends only on the current state and is independent of previous observations, and we have:

$$P\left(o_t | o_1^{t-1}, q_1^t\right) = P(o_t | q_t). \tag{6}$$

The Markov property of a process can also be expressed in mathematical language. Consider a set of random variables $[X\,(S), S \geq 0]$ and a set of system states at moments $t1$, $t2, \cdots, tn$. If $X\,(t)$ follows the Markov process, for all values $x1, x2, \cdots, xn$, the following relation holds:

$$X(t_n + 1) = \begin{cases} \dfrac{x_1 + \cdots + x_k}{x_n}, & 1 < k \leq n | F_1, \cdots, F_k \subseteq t_1 \\ \dfrac{x_1 + \cdots + x_k}{x_n}, & 1 < k \leq n | F_1, \cdots, F_k \subseteq t_2 \ , \\ \dfrac{x_1 + \cdots + x_k}{x_n}, & 1 < k \leq n | F_1, \cdots, F_k \subseteq t_n \end{cases}$$

$$P[X(t_n + 1) \leq x | X(t_n) = x_n, \cdots, X(t_2) = x_2, X(t_1) = x_1] = P[X(t_n + a) \leq x | (t_n) = x_n]. \tag{7}$$

According to this relation, it can be said that the past states of the system can play a part in determining the next state of the system. In Markov Models, the states of system are denoted by t that can be continuous or discrete. The fact that t is discrete and can be interpreted in this way:

The behavior of the system is studied only at certain points in time. If t is discrete, $X(t)$ is replaced by random variables in the form of $X_1, X_2 \cdots$, and X_n. The set of values that $X(t)$ can choose, by definition, is called the system state. System mode can also be discrete or continuous. Given that the requests sent to the edge environment might be related to each other, the first assumption based on nonindependence is true in the present case, and the past state of the system is examined to determine the possibility of assigning tasks to resources. In the proposed method, the value of the Markov hidden process at $t + 1$ can be considered as the probability of assigning a new task to a resource with respect to the assignment of previous tasks to that resource. In this case, the system is the resources to which tasks are assigned, and then, the system state changes to a stable state. In the proposed method, according to the assignment of previous tasks in the edge environment, the possibility of assigning tasks in a limited time can be considered for each resource. Given that our tasks are removed from the edge environment by running on resources, the number of these tasks in the resource queue is reduced, and then, the possibility of allocating resources at a particular time should be considered. This probability should be updated sequentially with the arrival of the new task, and the new tasks should be assigned to the source with the highest probability.

3.3. Game Theory. As illustrated in Figure 1, the proposed method uses game theory as a strategy in order to assign tasks to resources. In the previous steps, a near-optimal solution based on a genetic algorithm using a fit function based on integer linear programming was proposed. In this method, the proportionality function considers the constraints related to cost, execution time, and resource efficiency. Output solutions of genetic algorithms based on time, cost, and resource efficiency optimization are presented. However, in these solutions, the status of assigning previous tasks to resources and controlling the load balance between resources has not been considered. Therefore, in the proposed method, the hidden Markov model is used to investigate the distribution of tasks among resources according to the distribution of previous tasks. According to this model, based on the current status of the resource and the status of the assignment of previous tasks, allocation probability is assigned to each resource. Therefore, tasks that are less likely to be distributed should be removed from the generated solutions and then replaced with another solution that is out of the genetic algorithm based on its ranking. Furthermore, the proposed method uses game theory considering that the edge environment is a dynamic multifactor environment and tasks compete to get access to resources, in order to dynamically allocate and control the load balance and service quality parameters. Game theory is responsible for step-by-step review and control of the overall state of the edge system with respect to assigning tasks to resources according to the proposed solution.

Game theory is a mathematical approach for decision making that analyzes competitive situations to determine the optimal actions. In recent years, game theory has been widely used to deal with various problems related to managerial optimization. In particular, the issue of multifactor scheduling with game theory has attracted considerable attention from researchers [43]. In a work using the game-based two-layer scheduling method, with the aim of reducing the completion time of tasks in resources, balancing the total load of machines and energy consumption to achieve real-time data-based optimization for the edge development environment is of great importance [44]. In another work, in order to solve the problem of scheduling edge services by several factors, a model is proposed based on game theory to coordinate the relationships between diverse parameters, according to different strategies of service providers [45].

3.3.1. Game-Based Scheduling Formulation. In the proposed method, the scheduling problem can be considered as a noncooperative game between N players with complete information in which each machine as a player decides on the next

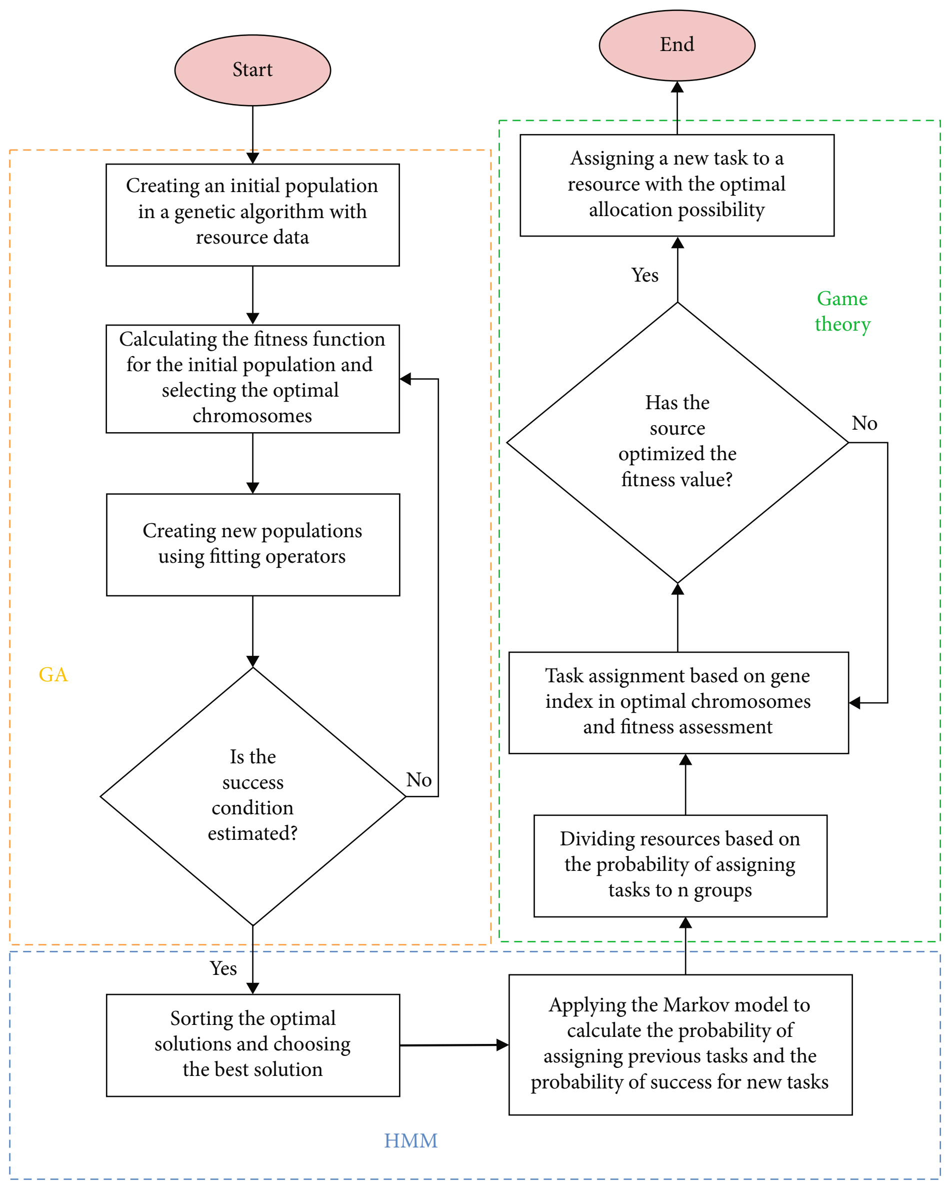

Start
Creating an initial population in a genetic algorithm with resource data
Calculating the fitness function for the initial population and selecting the optimal chromosomes
Creating new populations using fitting operators
Is the success condition estimated?
No
Yes
GA
Sorting the optimal solutions and choosing the best solution
Applying the Markov model to calculate the probability of assigning previous tasks and the probability of success for new tasks
HMM
Dividing resources based on the probability of assigning tasks to n groups
Task assignment based on gene index in optimal chromosomes and fitness assessment
Has the source optimized the fitness value?
No
Yes
Game theory
Assigning a new task to a resource with the optimal allocation possibility
End

Figure 5: Flowchart of the proposed method.

move (performing task k). And game strategies (i.e., selecting an appropriate method to process tasks) are selected to achieve their goal (optimization of service quality parameters). This game is defined as a timing game in the edge environment in a triple pair consisting components $G = \{N, S, U \cdots\}$. To apply game theory to task scheduling and its allocation to resources, we must first define the game to meet the requirements. The game scheduling task variables are defined in detail as follows:

(i) Players: In the game of task scheduling to resources in the proposed method, virtual machines act as players in a dynamic edge environment. All virtual machines are divided into n groups according to their assignment probability and the machines in each group act as potential players for a game. Given that the Mk machine, which belongs to the gt group machine from the edge environment, is a suitable option for allocation based on the cost solution of the genetic algorithm stage and considering the probability of allocations, the virtual machines in the gt group of players are candidates for the next stage of the game. They can be candidates because the probability of distribution of this group of virtual machines is high, and if the genetic algorithm is in the proposed solution, it can be selected as the next stage player

(ii) Strategies: In edge environment scheduling, task priorities are determined by the control modules in the edge environment according to whether the task should be performed or transferred in order to balance the edge environment. The overall

Table 2: Complexity of each step in proposed method.

Operator	Complexity
1-scheduling step {	
1-1 Genetic algorithm {	(it = n)
1-1-1 Generate randomly initial population	(c = constant)
1-1-2 Evaluating the initial N chromosome	(n)
1-1-3 Crossover the initial population	($n/2$)
1-1-4 Mutation the population	(n)
1-1-5 Select the optimal chromosome}}	($\log(n)$)
2-Load balancing step {	
2-1 Hidden Markov Model {	
2-1-1 Calculate the probability of load in VMs}	(m)
2-2 Game theory {	
2-2-2 Calculate of the objective in each VMs vs Scheduled solutions}}	($m * n$)

strategy in the edge environment is to improve runtime, execution cost, and resource efficiency based on load balancing with each player trying to maximize their fitness by choosing the right processes. According to the solution chosen by the genetic algorithm and also according to the number of groups created in the game, a set of n tasks is created that corresponds to n groups of virtual machines. The first task in the edge environment is examined according to the near-optimal solution for the virtual machine in group i which is located in the first gene of the superior chromosome. If other machines in group i have the ability to increase the improvement of the proportion of the first task, the index of the virtual machine in the first gene will be replaced with a more efficient virtual machine in group i. In the edge environment scheduling game, each machine tries to organize its processing queue in a reasonable way so that it can maximize the efficiency of the resource and reduce the time and cost of the task and then by adopting appropriate strategies, tries to minimize the efficiency of other machines in a group that are considered as rivals. Therefore, it should be noted that the fitness of a machine is influenced not only by its chosen strategies but also by the strategies chosen by other machines in the game based on load balancing

(iii) Fitness function: Fitness function in the game indicates the strength of the players at different stages. Fitness acts as an indicator for machines to choose their strategies. In the proposed method, the repayment of a machine is related to the load index and the amount of productivity of resources and the time and cost of completing the task. As mentioned in the previous section, proportion is directly related to productivity and inversely related to time, cost, and load balance. Therefore, the proportion of the Mk virtual machine in the gt group at stage h of the game is defined in

$$U_k^h(M) = \frac{\eta_k^h}{TC_k^h}. \tag{8}$$

In particular, when a processing task is taken from the Mk machine and assigned to another virtual machine with the same or a higher probability allocation, the Mk permissible machine is defined as zero so that it can participate in the rest of the game. Figure 5 illustrates the proposed method flowchart.

3.4. Complexity of the Proposed Method. In order to calculate the complexity of the proposed method, all operators in the proposed method must be considered for the tasks and virtual machines used. The proposed method has two main steps, including scheduling and load balancing. Also, n tasks are assigned to m virtual machines during *it* iteration of the genetic algorithm. In Table 2, we examine the operators and the complexity of each step in detail, and then, the order of the complexity of the proposed method is given in

$$\begin{gathered} T(n) = n \times \left(n + \frac{n}{2} + n + \log(n)\right) + (m + (m \times n)) + C, \\ T(n) = 2n^2 + \frac{n^2}{2} + (n \times \log(n)) + m + (m \times n) + C, \\ T(n) = \frac{7n^2}{2} + (n \times \log(n)) + m + C, \\ \mathrm{T}(n) \in \mathrm{O}\left(n^2\right). \end{gathered} \tag{9}$$

As shown in Equation (9), complexity order of proposed method is $O(n^2)$ that is directly related to the number of input tasks.

4. Proposed Method Implementation

In order to implement the proposed method, random scenarios with 50, 60, 70, 80, 90, and 100 services in the form

Table 3: An example of the initial population in the proposed genetic algorithm.

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
2	5	5	5	5	6	2	3	3	6	4	3	2	6	2	6	1	1	2	1
1	1	5	4	3	4	3	4	6	1	5	1	6	1	2	5	6	3	5	5
1	6	6	2	2	3	2	1	2	4	4	1	1	2	1	5	2	1	2	1
1	2	1	5	5	4	6	6	6	2	6	6	2	6	2	2	5	4	4	2
1	3	3	5	2	2	1	3	2	4	1	5	4	3	6	2	1	5	3	5
6	2	6	5	1	3	3	3	6	2	3	2	3	4	5	1	5	6	2	2
3	6	6	3	1	5	1	5	5	1	4	4	6	1	2	3	6	5	3	6
5	1	5	4	1	3	1	3	6	4	5	5	2	6	6	1	5	6	5	2
5	2	2	1	2	6	3	5	1	5	2	4	6	2	1	4	6	6	4	3
1	5	6	3	3	1	5	1	4	4	5	5	6	5	1	5	5	4	4	3

of a workflow and 10 virtual machines with different features to perform tasks in the edge environment are defined. The proposed method is simulated in MATLAB software version 2020. Services sent to the edge computing environment are workflows in which one service could be randomly dependent on the others. Therefore, considering the task interdependence, data transfer among services and distribution of services in different virtual machines in terms of geography and priority in providing services, the scheduling of tasks in order to maintain the quality of service and load balance in the edge computing environment is complicated. In the proposed method, when a workflow is introduced, the services are randomly assigned to virtual machines to form the chromosomes in the proposed genetic algorithm. In the proposed method, in the first stage, the genetic algorithm provides a random scheduling by assigning tasks to resources. In the next steps, based on the proposed control framework, in order to achieve the goals of increasing resource efficiency, reducing the total time of tasks in the edge environment, and reducing energy consumption of resources in performing tasks as well as reducing the cost of performing tasks on resources, based on the fitness and mutation operators in the proposed genetic algorithm, this schedule changes and tends towards optimal points. Table 3 shows an example of the initial population in the proposed method.

Accordingly, in the first step of the proposed method, we evaluate the initial population using the fitness function introduced in Equation (1). In this regard, each chromosome according to the priority of the service, the relationship between the services, the time of service based on the number of instructions in each service, the cost of service, and the energy required to run each service in the virtual machine distributed in the environment edge computing is subject to evaluation. Given that the objective function of the proposed genetic algorithm uses a linear optimization problem to improve service quality parameters, the best fitness is given to the chromosome that balances the execution of services in virtual machines. Hence, for each of the chromosomes created in the initial population, a fitness value is calculated, and the value of this fitness function is obtained from the optimality of each gene or the execution of a service in a virtual machine based on the characteristics of that virtual machine. In fact, the fitness of a chromosome is equal to the average fitness of each of the genes in the corresponding virtual machines. Table 4 shows the fitness values of the chromosomes in the initial population.

Table 4: Fitness values of the initial population.

Chromosome index	The value of the fitness function
1	0.8215
2	0.8557
3	0.7521
4	0.8492
5	0.7416
6	0.7554
7	0.8153
8	0.8062
9	0.8221
10	0.8036
11	0.7565
12	0.8050
13	0.8715
14	0.8123
15	0.7669
16	0.7583
17	0.8257
18	0.8040
19	0.8565
20	0.8084
21	0.8420
22	0.8075
23	0.7664
24	0.8113
25	0.7921
26	0.7627
27	0.7550
28	0.8111
29	0.7945
30	0.7315

Table 5: Fitness values for population after crossover.

Chromosome index	The value of the fitness function
1	0.8406
2	0.8467
3	0.8189
4	0.8510
5	0.8070
6	0.8215
7	0.8119
8	0.7967
9	0.8213
10	0.8609
11	0.8094
12	0.8244
13	0.7785
14	0.8749
15	0.8133
16	0.8355
17	0.8162
18	0.8049
19	0.8537
20	0.8144
21	0.8082
22	0.8084
23	0.7967
24	0.8119
25	0.8018
26	0.8738
27	0.7912
28	0.8659
29	0.8376
30	0.7991

Table 6: Fitness values for population after mutation.

Chromosome index	The value of the fitness function
1	0.8211
2	0.8486
3	0.8112
4	0.8125
5	0.8040
6	0.8084
7	0.8039
8	0.8111
9	0.8221
10	0.8113
11	0.8113
12	0.8123
13	0.8153
14	0.8050
15	0.8186
16	0.8266
17	0.8112
18	0.8558
19	0.8125
20	0.8030
21	0.8065
22	0.8075
23	0.8078
24	0.8123
25	0.8541
26	0.8123
27	0.8110
28	0.8040
29	0.8053
30	0.8153

As shown in Table 4, the value of the fitness function for each of the chromosomes in the initial population is calculated according to the average fitness for each of the chromosomes. The fitting operator is applied on the initial population in order to change the new population in order to improve the optimization and increase the value of the fitness function and to diversify the new population compared to the initial population. In the proposed method, the fitness probability value is considered 0.6, which is selected according to the previous methods and can be changed. Given that the chromosomes in the initial population have 50 genes, 30 genes from chromosome i and the remaining 20 genes from chromosome $i+1$ are selected, and a new population is created based on this crossover operator. Now, in this step, the evaluation is done using the proposed fitness function on the new population. Table 5 shows the values of the fitness function for the chromosomes in the new population.

As shown in Table 5, the values of the chromosome after crossover operator are calculated according to the proposed fitness function. It can be seen that about 75% of the chromosomes, after crossover, have improved compared to the initial population. In the next step, in order to create diversity and improvement in the new population, the mutation operator has been applied. The mutation operator randomly selects one or more genes on some chromosomes in a population and replaces their values with allowable values. In fact, the mutation operator takes one or more services from the virtual machines to which they are assigned and then assigns them to other virtual machines. In this case, the chromosomes will change, and of course, there will be a change in the fitness of chromosomes. Table 6 shows fitness values for mutant chromosomes.

As shown in Table 6, the values of the fitness function for the mutant population are calculated. By looking closely at the values of the mutant population fitness function, it can be seen that approximately 50% of the chromosomes that were mutated showed improvement and the remaining 50% had the opposite results. Figure 6 shows the fitness

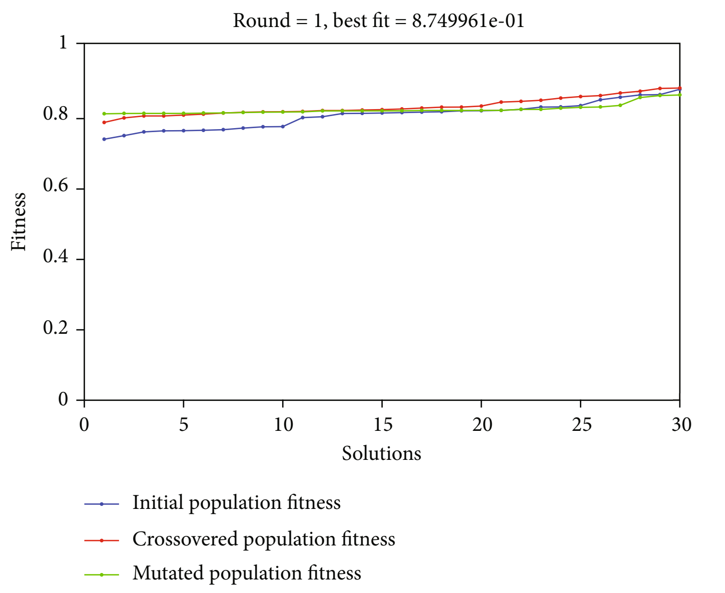


Figure 6: Fitness function values in the first stage of the genetic algorithm.

improvement graphs for the initial population, the new population after crossover, and the mutation in the first stage of the genetic algorithm.

As shown in Figure 6, the values of the fitness function for the initial population are shown in blue, the crossovered population in red, and the mutated population in green. The crossovered population is generally higher than the other two populations, and this indicates load balancing and service quality factors in scheduling solutions in a crossovered population. The values of the fitted and mutated populations are mostly oriented towards improving the fitness of the original population which indicates the optimization of scheduling based on the proposed method. In the next step, according to the selection operator, those chromosomes that are more fitness than the average proportion of the total chromosomes are selected as the expert population and the rest of the chromosomes are removed from the initial population. Fitness and mutation operators are applied to the new expert population, respectively, in order to find the best population among the scheduling solutions. The final population is the most optimal scheduling solution that, in addition to load balancing, takes service quality factors into account. Figure 7 shows examples of the replication process of a genetic algorithm.

As shown in Figure 7, the values of the fitness function are gradually optimized in order to get scheduling solutions, and finally, the best population with the fitness function of 97.4 is selected as the proposed scheduling based on the genetic algorithm. After this stage, according to the proposed method, control solutions for load balancing and service quality factors are presented in two steps, Markov model and game theory.

4.1. Implementing the Control Framework. After assigning services to virtual machines, the queue of these virtual machines can be more crowded than the others considering that powerful virtual machines may receive more tasks due to the fit function, and obviously, it will improve the balance of load distribution among virtual machines and require more time to perform all services as well which could affect service quality factors. Therefore, in order to overcome this challenge, the hidden Markov model has been applied. The proposed hidden Markov model, which is part of the proposed control framework, calculates the probability of allocations according to the allocation of services for each of the virtual machines. Based on this allocation probability, the new service will be allocated with respect to the fitness function in the genetic algorithm, the number of pending services and the number of services assigned to the genetic algorithm output population, the performance of the virtual machine in the data set, and the allocation probability of the Markov model to the most suitable virtual machine. Table 7 shows the probability of each virtual machine when allocating new services based on the Markov model.

As presented in Table 7, the Markov model calculates an allocation probability for each virtual machine, and based on this allocation probability, it is possible to understand how the previous services were distributed and how full the queue of each processor is. During each round of service allocation to virtual machines, the allocation probability is updated, and each service is assigned to the virtual machine with the highest allocation probability. Therefore, the new services are directed toward the most appropriate virtual machines considering the allocation probability and the population provided by the genetic algorithm. In order to determine the status of service quality factors during each step of assigning the service to a virtual machine in the proposed method, the game pattern is used as another component of the hierarchical control framework. Due to the fact that a task completion deadline is defined for each service and computing task in the edge computing environment, in order to help increase the reliability of the scheduling

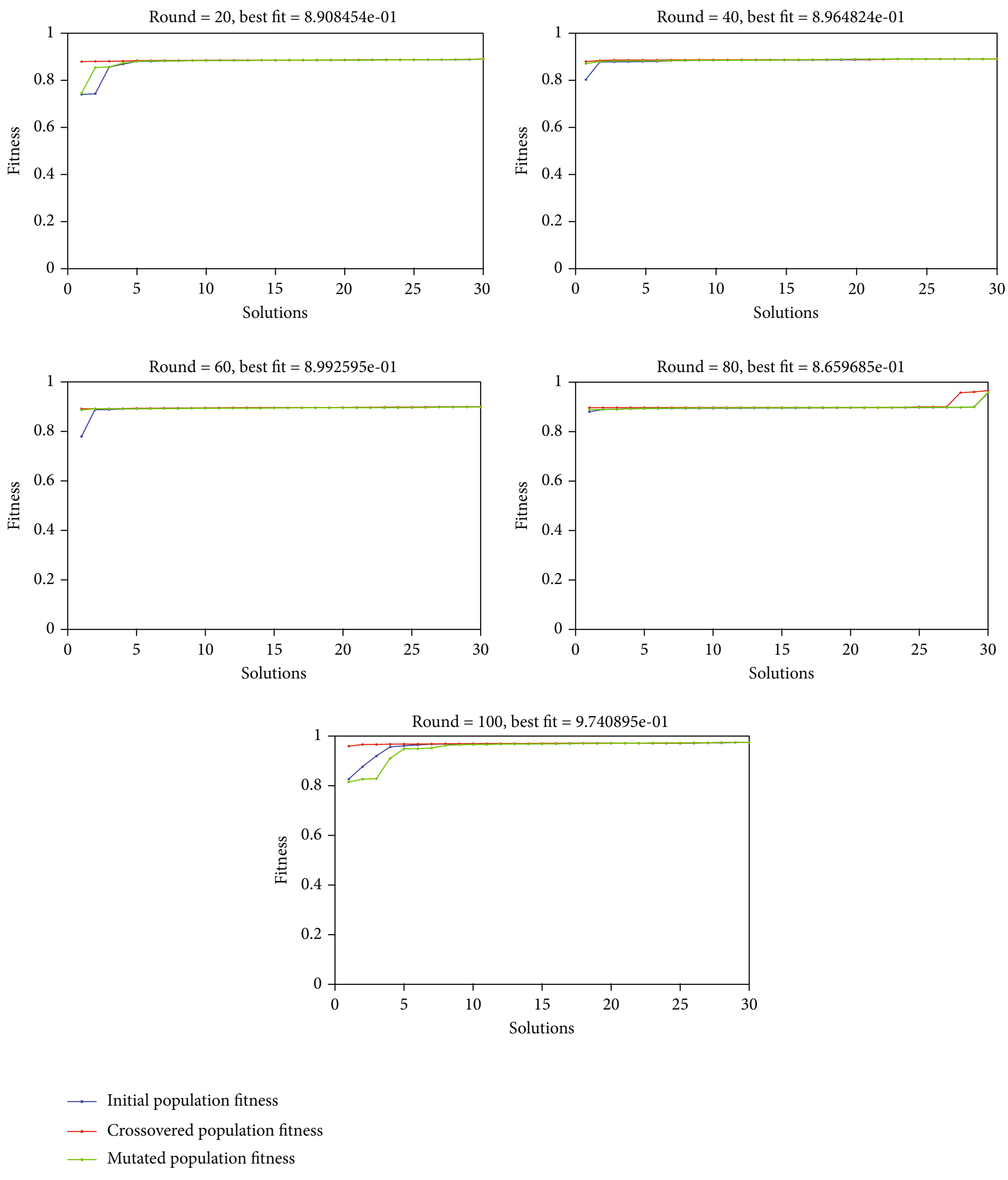


Figure 7: Process of optimization of the fitness function in the genetic algorithm.

method in the edge environment, in addition to calculations related to completion time, cost, and energy consumption in the edge computing environment, the possibility of performing a service in a virtual machine is considered to prevent the expiration of the deadline for service. In case the virtual machines are not able to run the service within the specified time, the scheduling operation is performed again, and the service is assigned to another virtual machine. Table 8 shows the game process for a round of service allocation to virtual machines.

As shown in Table 8, the allocation of services to virtual machines is presented according to the fitness function for the genetic algorithm, the probability of allocations, and the optimal amount of each virtual machine in the multifactor game. Services that cannot be processed by any virtual machine are also identified. In the following, we will evaluate the proposed method.

Table 7: Probability of allocation to virtual machines.

Virtual machine	Allocation probability
1	0.5227
2	0.1250
3	0.4053
4	0.1439
5	0.1098
6	0.1856
7	0.2689
8	0.1780
9	0.4696
10	0.8534

Table 8: Game theory strategy in the proposed hierarchical control framework.

Virtual machine number	Game theory results		Out of deadline services index
	Strategy A	Strategy B	
1	0.374	0.626	3
2	0.749	0.251	2, 7, 25
3	0.6	0.4	3, 2, 7
4	0.571	0.429	—
5	0.444	0.556	25
6	0.455	0.545	7, 25
7	0.462	0.538	10
8	0467	0.533	—
9	0.471	0.529	10
10	0.474	0.526	3

4.2. Performance Evaluation for the Proposed Method. After implementing the proposed method in the form of a hierarchical control framework consisting of linear optimization, in order to create scheduling plans based on time, cost, and energy factors using genetic algorithms and calculating the probability of allocating services to virtual machines using hidden Markov model and load balance control and service quality factors as well as feasibility of performing each task and providing services in virtual machines based on deadline by dynamic multifactor theory of the game, we evaluate the performance of the proposed method in order to provide the improvement obtained from the combination of the above methods, and then, we evaluate the quality of the proposed method. Due to the importance of load balancing and optimal scheduling in performing services and computational tasks for users in the edge computing environment and in order to comply with service quality factors, various evaluation criteria have been introduced to measure the improvement of scheduling methods in publications. Considering the mentioned goals for the research, the proposed method has been investigated in terms of service completion time in virtual machines (makespan), the total cost of services in virtual machines, the energy required to run all services in the computing center and data storage in edge computing environment, the performance efficiency of virtual machines in edge computing environment, and the reliability of the proposed scheduling approach.

One of the main criteria for evaluating scheduling approaches in the edge computing environment is the time to complete services in virtual machines, known as makespan. Calculating this criterion requires calculating time from the moment of entering a task to the moment of completing it which consists of waiting time in order to access the virtual machine, the time required in order to transfer data in the network in case of need for transferring data from another service, the execution time, and the time required to perform a computational task or to provide services. Figure 8 shows the service completion time graph in virtual machines in the scenarios of proposed method.

As shown in Figure 8, the time required to complete services in virtual machines increases with the number of services. What can be seen is that increasing the number of services does not have much effect on the completion time of all tasks in virtual machines. Due to the near-optimal scheduling in the proposed method, the time required to complete the services with increasing their number increases appropriately, which indicates the efficiency of the proposed method in controlling the load and scheduling of tasks. Also, this indicates that there is not a long waiting time to get to the virtual machine. We can also observe that long services require a minimum runtime due to the fact that powerful virtual machines are responsible for long services.

Another criterion that has been evaluated in the proposed method is the cost of running services in virtual machines. The cost of performing services includes the cost of transferring information between two tasks in virtual machine network and the cost of performing a task or service on a virtual machine. Therefore, it is obvious that if one service is dependent on another service, the cost of performing it will be higher than other tasks. Figure 9 shows the cost of performing tasks in scenarios in virtual machines.

As shown in Figure 9, the cost of tasks and services entering the edge environment varies depending on the relationship between the services and some tasks, and services that require the transfer of information between virtual machines in the edge, via the network, will cost much more. Now, according to the proposed scheduling method, the total cost of performing tasks in scenarios in virtual machines is shown in Figure 10.

As shown in Figure 10, in the proposed method, the cost for executing and transferring data between virtual machines has increased via increasing the number of tasks. To understand the cost reduction in the proposed method, we can consider that the cost of 100 tasks using a random scheduling method such as the initial population of the genetic algorithm would be around 9000$, while in the proposed method, this cost is at most of $ 2,500, which shows a significant improvement. The proposed method has scheduled tasks in near optimally manner among virtual machines, and the cost reduction is achieved by using cheaper virtual machines for tasks that require communication.

Another criterion used to evaluate the hierarchical control framework mentioned in the proposed method is the energy consumption to perform computational tasks and

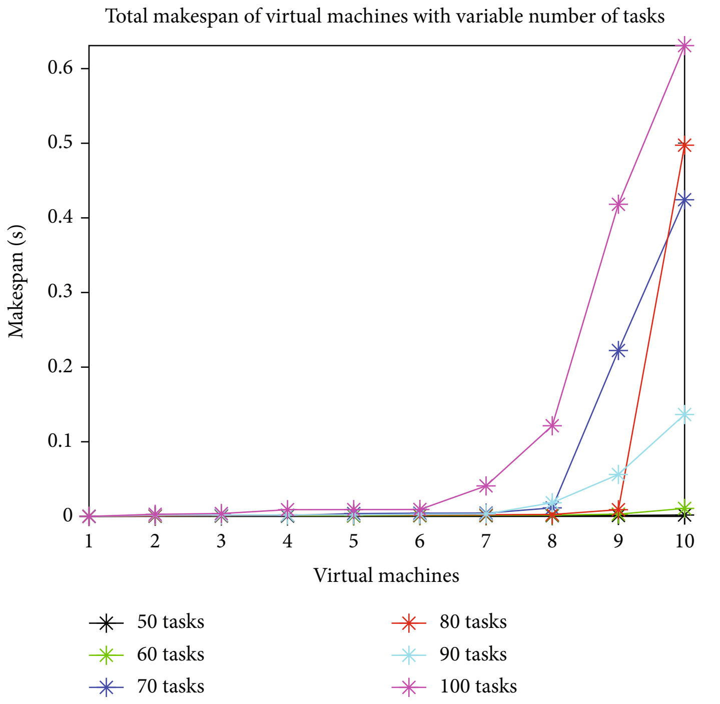


Figure 8: Service completion time in scenarios in virtual machines.

provide services in virtual machines. In order to perform tasks and provide services as well as computing and transferring data over a network between virtual machines, we need energy, and naturally, the presence of more communication in the workflow requires more energy. Figure 11 shows the energy consumption in scenarios in the proposed method.

As shown in Figure 11, the energy required to perform tasks in virtual machines increases with the number of tasks and the communication between tasks. The same procedure is used in the proposed method, but it can be seen that increasing the number of tasks does not impose unreasonable energy consumption on data centers.

Another criterion that has been evaluated in the proposed method is the performance of virtual machines during edge tasks execution. Virtual machine performance means the maximum use of virtual machines in performing computational tasks and providing services. Figure 12 represents the performance for virtual machines according to the scenarios in the proposed method.

As shown in Figure 12, in the proposed method, most virtual machines have high performance that indicates the distribution of uniform loads among the virtual machines in the edge computing environment. According to the Figure 12, the performance of virtual machines for different scenarios is almost close to each other and near to optimal. Proper scheduling of tasks to virtual machines with fair distribution of load among virtual machines has prevented virtual machines from being idle and losing cycles.

The last criterion considered in the proposed method is reliability. The proposed method relies on predicting the load probability in virtual machines based on the Markov model and avoids sending tasks to inappropriate virtual machines. It can be prevent the loss of deadline to perform tasks. This allows virtual machines to receive tasks that they are able to perform on their deadline. Therefore, the reliability of the proposed method will be high, and the tasks will be performed on the deadline. Figure 13 shows the reliability of the proposed method in the scenarios.

As can be seen from Figure 13, the reliability of the proposed method is close to each other in different scenarios. Proper distribution of load among virtual machines and considering the deadline for performing tasks based on the probability of load distribution in virtual machines has resulted in the proposed method of executing tasks with high reliability.

4.3. Comparison of the Proposed Method with Previous Methods. After implementing and evaluating the proposed method in order to measure the validity of its performance, we compare it with previous methods in the field of load balancing in the edge environment. Due to the importance of load balancing and creating appropriate schedules in order to benefit from service quality factors in edge computing, many articles have been presented, and many researchers have shown interest in this field. Therefore, the proposed method can be compared with the previous methods [14, 18, 32] based on the main criterion in scheduling methods, namely, the completion time of tasks in edge environment resources, which is considered and evaluated in many articles. Figure 14 illustrates a comparison of the proposed method with the methods available in the publications from the makespan criterion point of view.

As shown in Figure 14, the proposed method has performed better than other methods in publications in

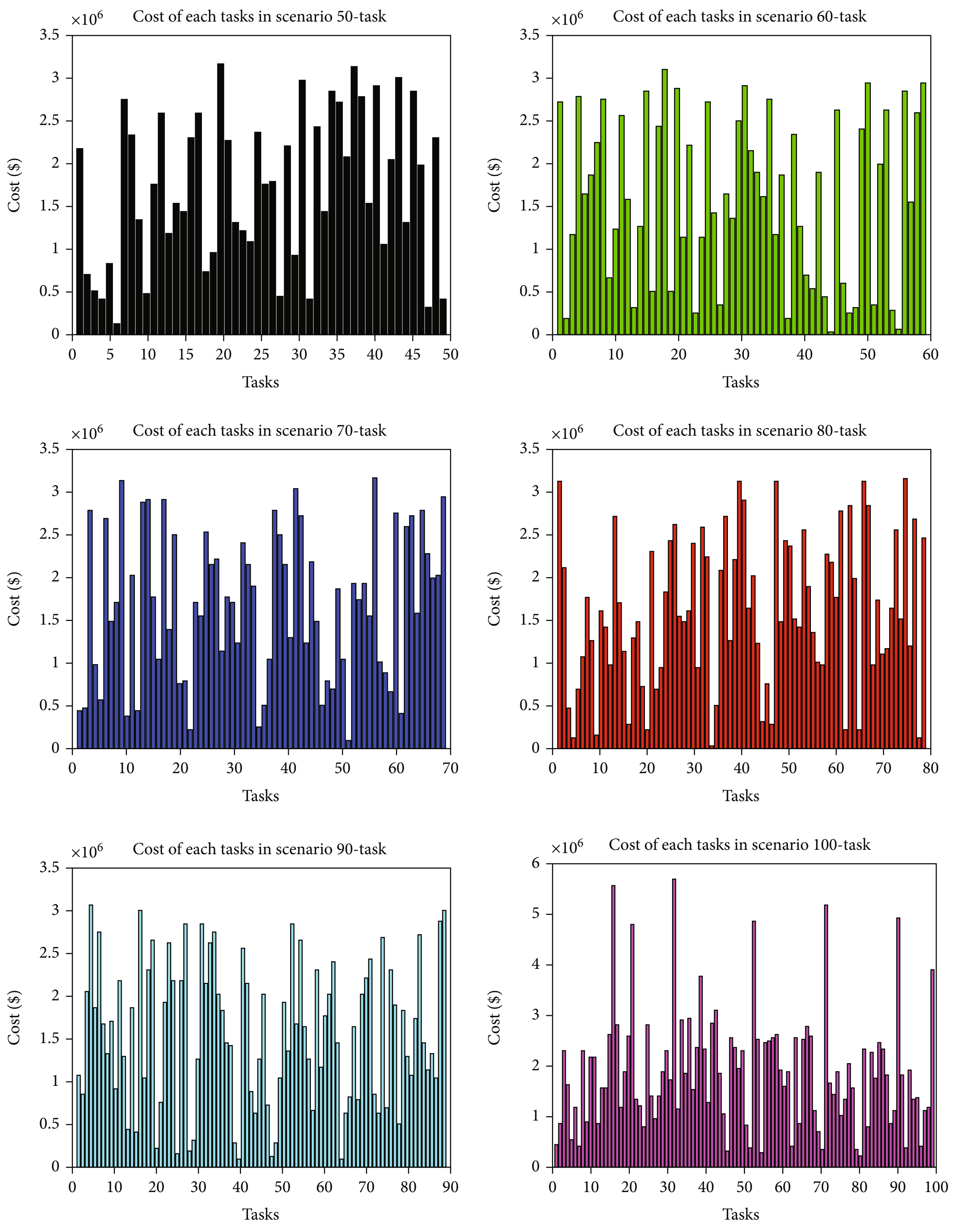


Figure 9: Cost of each tasks in scenarios.

allocating services to virtual machines. It has also reduced the service completion time in virtual machines. Therefore, the proposed method has a lower value for the makespan criterion compared to the previous methods and has well complied with service quality factors.

Another criterion added in this paper to compare with previous methods is the performance of the scheduling approach in the fog environment. The proposed method achieves good performance by assigning optimal tasks to virtual machines suitable for different scenarios (as shown in

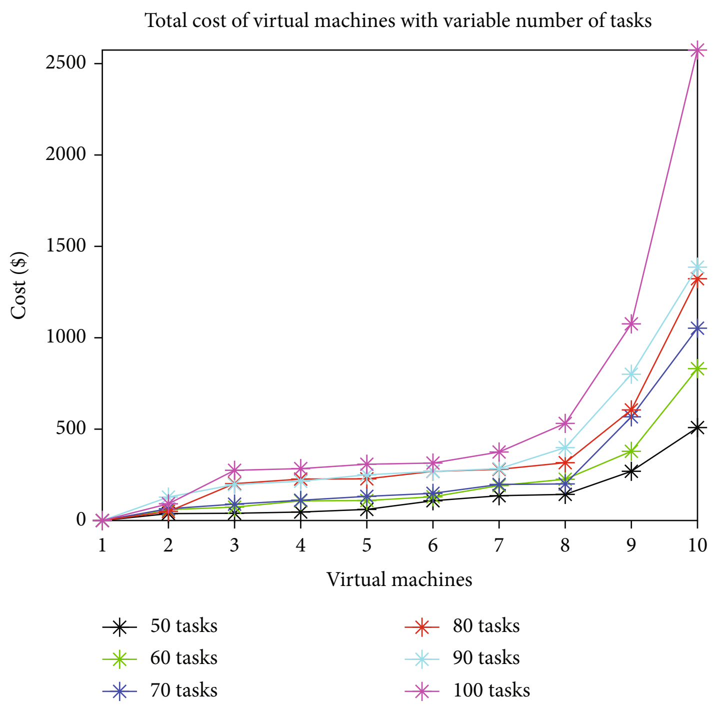


FIGURE 10: Total cost of performing tasks in the proposed method.

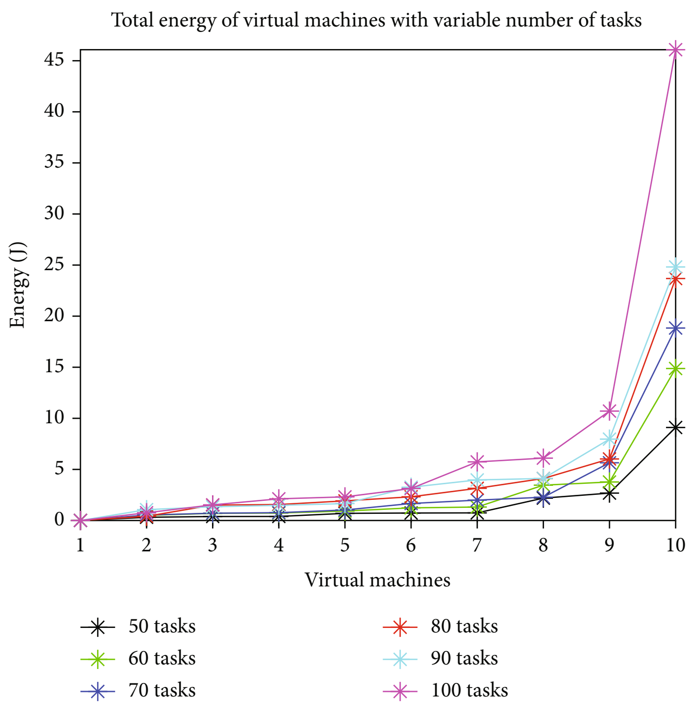


FIGURE 11: Energy consumption in scenarios in the proposed method.

Figure 12). Figure 15 shows a comparison of the performance of the proposed method with previous methods [14, 32, 46].

As shown in Figure 15, the performance of the proposed method is better than other existing methods. The proposed method by applying the control step to adjust and balance the load in the cloud environment and the use of linear programming in the genetic algorithm has been able to assign optimal tasks to the resources so that the performance of resources in the fog-cloud environment is near-optimal.

Another important issue in load balancing in a cloud environment is the cost of performing tasks on virtual

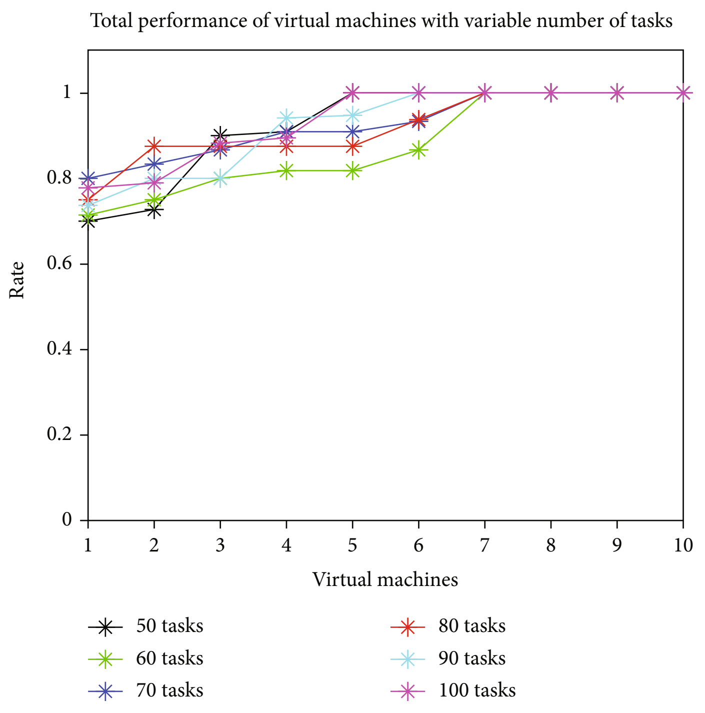


Figure 12: Virtual machine efficiency in the proposed method.

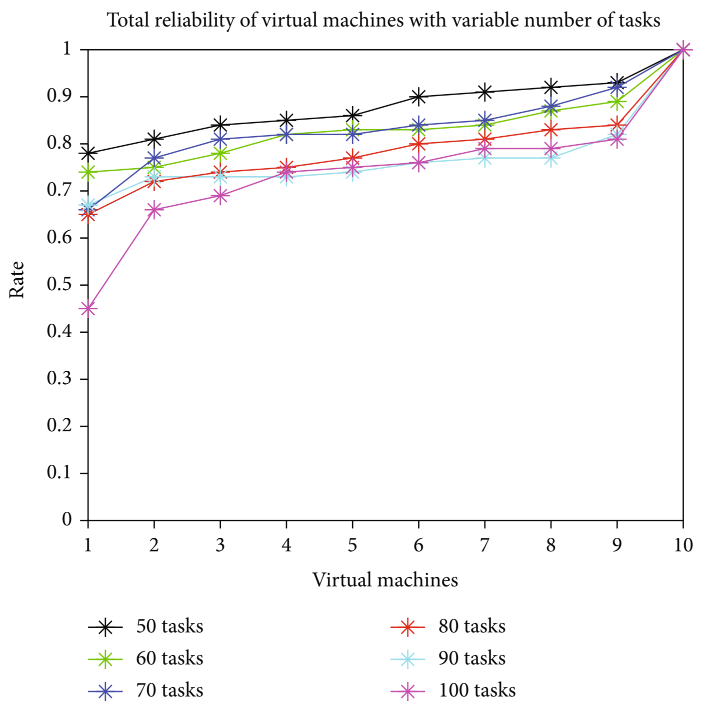
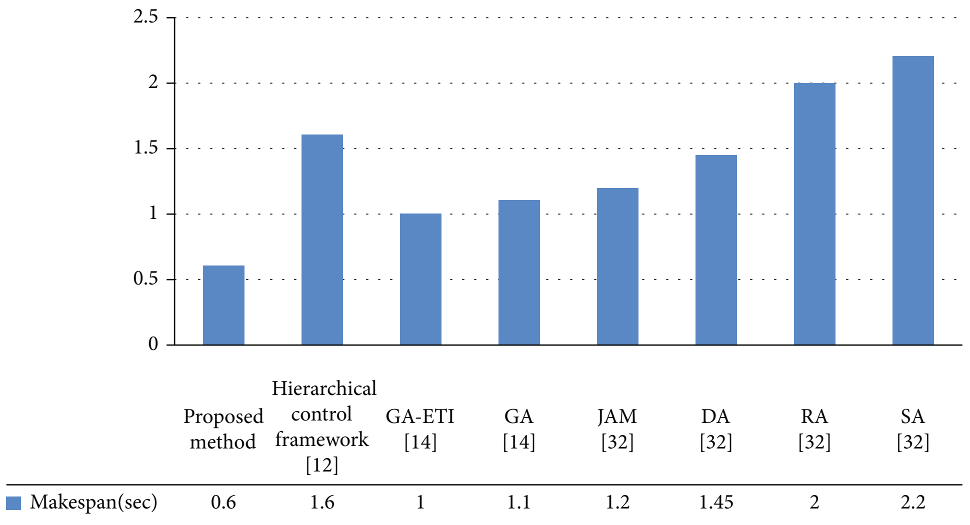


Figure 13: Reliability value of task executing in scenarios in proposed method.

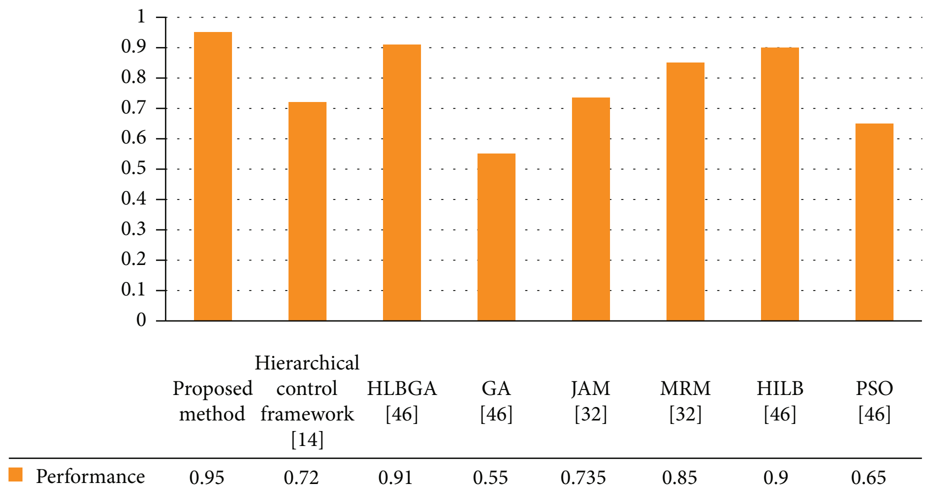
machines. In this paper, in order to show the improvement of the proposed method in reducing the costs associated with performing tasks in virtual machines, we compare the results of the proposed method with other existing methods [47, 48]. Figure 16 shows a comparison of the cost of performing tasks in virtual machines.

According to Figure 16, it can be seen that the proposed method has a lower cost of performing tasks than other existing methods. Considering that in the proposed method, one of the main parameters in the fitness function of the genetic algorithm has been the cost of performing tasks, the proposed method by optimizing this parameter has



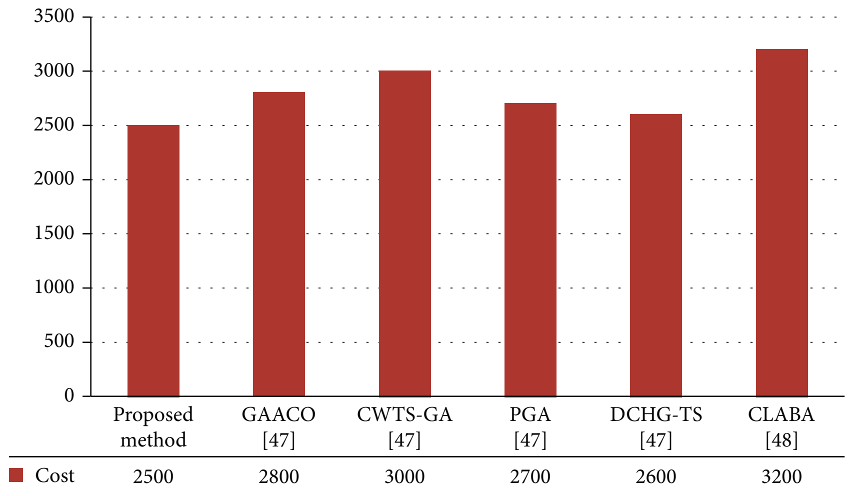
	Proposed method	Hierarchical control framework [12]	GA-ETI [14]	GA [14]	JAM [32]	DA [32]	RA [32]	SA [32]
🟦 Makespan(sec)	0.6	1.6	1	1.1	1.2	1.45	2	2.2

FIGURE 14: Comparison of the makespan in proposed method with previous methods.



	Proposed method	Hierarchical control framework [14]	HLBGA [46]	GA [46]	JAM [32]	MRM [32]	HILB [46]	PSO [46]
🟧 Performance	0.95	0.72	0.91	0.55	0.735	0.85	0.9	0.65

FIGURE 15: Comparison of the performance in proposed method with previous methods.



	Proposed method	GAACO [47]	CWTS-GA [47]	PGA [47]	DCHG-TS [47]	CLABA [48]
🟥 Cost	2500	2800	3000	2700	2600	3200

FIGURE 16: Comparison of the performance in proposed method with previous methods.

allocated tasks to resources at a reasonable cost. For this reason, the overall cost of performing the tasks is optimized compared to previous methods.

5. Conclusion

In this research, in order to balance the load in the edge computing environment, a hierarchical control framework for assigning tasks in edge computing services is presented. In the proposed method in the first stage, genetic algorithm is used to model the workload. The input of the genetic algorithm in the proposed method includes a set of tasks that are evaluated according to the priorities and the relationship between the tasks in order to model the load distribution among the existing hosts as well as optimizing the criteria. The evaluation criteria applied in this method include time constraints, resource processing power, probability of access to the resource, and the cost of performing tasks in the resource. In order to optimize the load distribution among sources, considering the existing limitations, the integer linear programming optimization in the evaluation function of the genetic algorithm has been used. In the second stage, according to the past load distribution in edge resources, the probability of load distribution among sources is calculated according to the hidden Markov model. Finally, in order to map the tasks to the virtual machines in each host, the game theory with QoS factors has been used as an evaluation function. The results of the experiments show that the proposed method, in comparison with other methods available in publications, in addition to providing balanced service allocations to virtual machines, has reduced the service completion time as well. Therefore, the proposed method has minimized the makespan compared to the previous methods and has well complied with service quality factors.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] V. Lander, H. Sastry, S. Katti, S. V. Vepa, and S. V. Shenoy, "Identity cloud service authorization model," US Patent 10454940B2, 2019.

[2] J. B. Rajkumar Buyya and A. Goscinski, *Cloud computing principles and paradigms*, John Wiley & Sons, 2019.

[3] M. Alouane and H. El Bakkali, "Virtualization in cloud computing: NoHype vs HyperWall new approach," in *2016 International Conference on Electrical and Information Technologies (ICEIT)*, pp. 49–54, Tangiers, Morocco, 2016.

[4] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: a survey," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–35, 2019.

[5] S. Einy, C. Oz, and Y. D. Navaei, "The anomaly- and signature-based IDS for network security using hybrid inference systems," *Mathematical Problems in Engineering*, vol. 2021, Article ID 6639714, 10 pages, 2021.

[6] S. Afzal and G. Kavitha, "Load balancing in cloud computing–a hierarchical taxonomical classification," *Journal of Cloud Computing*, vol. 8, no. 1, p. 22, 2019.

[7] M. Chiregi and N. J. Navimipour, "A new method for trust and reputation evaluation in the cloud environments using the recommendations of opinion leaders' entities and removing the effect of troll entities," *Computers in Human Behavior*, vol. 60, pp. 280–292, 2016.

[8] Y. D. Navaei and M. Afzali, "A survey on product recommendation system in e-commerce," *International Journal of Computer & Information Technologies*, vol. 2014, 2014.

[9] Y. D. Navaei and M. Afzali, "Dihedral product recommendation system for E-commerce using data mining applications," *International Journal of Computer & Information Technologies*, vol. 3, pp. 610–631, 2015.

[10] S. K. Mishra, B. Sahoo, and P. P. Parida, "*Load balancing in cloud computing: a big picture*," Journal of King Saud University-Computer and Information Sciences, 2018.

[11] Y. Nanehkaran, Z. Licai, J. Chen et al., "Anomaly detection in heart disease using a density-based unsupervised approach," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 6913043, 14 pages, 2022.

[12] Z. Peng, M. S. Jabloo, Y. D. Navaei et al., "An improved energy-aware routing protocol using multiobjective particular swarm optimization algorithm," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6677961, 16 pages, 2021.

[13] S. Einy, C. Oz, and Y. D. Navaei, "Network intrusion detection system based on the combination of multiobjective particle swarm algorithm-based feature selection and fast-learning network," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6648351, 12 pages, 2021.

[14] N. Leontiou, D. Dechouniotis, S. Denazis, and S. Papavassiliou, "A hierarchical control framework of load balancing and resource allocation of cloud computing services," *Computers & Electrical Engineering*, vol. 67, pp. 235–251, 2018.

[15] Z. Peng, M. Rastgari, Y. D. Navaei et al., "TCDABCF: a trust-based community detection using artificial bee colony by feature fusion," *Mathematical Problems in Engineering*, vol. 2021, Article ID 6675759, 19 pages, 2021.

[16] S. Einy, C. Oz, and Y. D. Navaei, "IoT cloud-based framework for face spoofing detection with deep multicolor feature learning model," *Journal of Sensors*, vol. 2021, Article ID 5047808, 18 pages, 2021.

[17] S. Jamali and Y. D. Navaei, "A two-level product recommender for E-commerce sites by using sequential pattern analysis," *International Journal of Integrated Engineering*, vol. 8, no. 1, 2016.

[18] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments," *Journal of Computational Science*, vol. 26, pp. 318–331, 2018.

[19] H. Ibrahim, R. O. Aburukba, and K. El-Fakih, "An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers," *Computers & Electrical Engineering*, vol. 67, pp. 551–565, 2018.

[20] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic Stackelberg game based resource allocation using hidden Markov for cloud computing," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 78–89, 2018.

[21] T. Halabi, M. Bellaiche, and A. Abusitta, "Toward secure resource allocation in mobile cloud computing: a matching game," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 370–374, Honolulu, HI, USA, 2019.

[22] O. Elzeki, M. Reshad, and M. Elsoud, "Improved max-min algorithm in cloud computing," *International Journal of Computer Applications*, vol. 50, no. 12, pp. 22–27, 2012.

[23] P. Samal and P. Mishra, "Analysis of variants in round robin algorithms for load balancing in cloud computing," *International Journal of computer science and Information Technologies*, vol. 4, no. 3, pp. 416–419, 2013.

[24] V. W. Thawari, S. D. Babar, and N. A. Dhawas, "An efficient data locality driven task scheduling algorithm for cloud computing," *International Journal in Multidisciplinary and Academic Research*, vol. 1, no. 3, 2012.

[25] S. Sharma, S. Singh, and M. Sharma, "Performance analysis of load balancing algorithms," *World Academy of Science, Engineering and Technology*, vol. 38, no. 3, pp. 269–272, 2008.

[26] D. Agarwal and S. Jain, "Efficient optimal algorithm of task scheduling in cloud computing environment," 2014, https://arxiv.org/abs/1404.2076.

[27] K. Mahajan, A. Makroo, and D. Dahiya, "Round robin with server affinity: a VM load balancing algorithm for cloud based infrastructure," *Journal of Information Processing Systems*, vol. 9, no. 3, pp. 379–394, 2013.

[28] G. Joshi and S. Verma, "Load balancing approach in cloud computing using improvised genetic algorithm: a soft computing approach," *International Journal of Computers and Applications*, vol. 122, no. 9, pp. 24–28, 2015.

[29] C. T. Joseph, K. Chandrasekaran, and R. Cyriac, "A novel family genetic approach for virtual machine allocation," *Procedia Computer Science*, vol. 46, pp. 558–565, 2015.

[30] S. A. Hamad and F. A. Omara, "Genetic-based task scheduling algorithm in cloud computing environment," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 4, pp. 550–556, 2016.

[31] T. Wang, Z. Liu, Y. Chen, Y. Xu, and X. Dai, "Load balancing task scheduling based on genetic algorithm in cloud computing," in *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, pp. 146–152, Dalian, China, 2014.

[32] G. Yi, H. W. Kim, J. H. Park, and Y. S. Jeong, "Job allocation mechanism for battery consumption minimization of cyber-physical-social big data processing based on mobile cloud computing," *IEEE Access*, vol. 6, pp. 21769–21777, 2018.

[33] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 275–295, 2015.

[34] S. Chakraborty and K. Mazumdar, "Sustainable task offloading decision using genetic algorithm in sensor mobile edge computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1552–1568, 2022.

[35] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, and M. J. Ryan, "IEGA: an improved elitism-based genetic algorithm for task scheduling problem in fog computing," *International Journal of Intelligent Systems*, vol. 36, no. 9, pp. 4592–4631, 2021.

[36] B. Natesha and R. M. R. Guddeti, "Adopting elitism-based genetic algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," *Journal of Network and Computer Applications*, vol. 178, article 102972, 2021.

[37] M. Abbasi, E. Mohammadi Pasand, and M. R. Khosravi, "Workload allocation in iot-fog-cloud architecture using a multi-objective genetic algorithm," *Journal of Grid Computing*, vol. 18, no. 1, pp. 43–56, 2020.

[38] S. Kashyap, S. K. Singh, A. Rouniyar, R. Saxena, and A. Kumar, "Load balancing and resource allocation in fog-assisted 5G networks: an incentive-based game theoretic approach," 2022, https://arxiv.org/abs/2202.05128.

[39] S. Kalantary, J. Akbari Torkestani, and A. Shahidinejad, "Resource discovery in the internet of things integrated with fog computing using Markov learning model," *The Journal of Supercomputing*, vol. 77, no. 12, pp. 13806–13827, 2021.

[40] A. Mebrek and A. Yassine, "Intelligent resource allocation and task offloading model for IoT applications in fog networks: a game-theoretic approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2021, pp. 1–15, 2021.

[41] S. Balasubramanian and T. Meyyappan, "Game theory based offload and migration-enabled smart gateway for cloud of things in fog computing," in *Computing in Engineering and Technology*, pp. 253–266, Springer, 2020.

[42] M. Franzese and A. Iuliano, "Hidden markov models," in *Encyclopedia of Bioinformatics and Computational Biology*, pp. 753–762, Academic Press, Oxford, UK, 2019.

[43] J. Xiao, W. Zhang, S. Zhang, and X. Zhuang, "Game theory–based multi-task scheduling in cloud manufacturing using an extended biogeography-based optimization algorithm," *Concurrent Engineering*, vol. 27, no. 4, pp. 314–330, 2019.

[44] Y. Zhang, J. Wang, and Y. Liu, "Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact," *Journal of Cleaner Production*, vol. 167, pp. 665–679, 2017.

[45] Y. Zhang, J. Wang, S. Liu, and C. Qian, "Game theory based real-time shop floor scheduling strategy and method for cloud manufacturing," *International Journal of Intelligent Systems*, vol. 32, no. 4, pp. 437–463, 2017.

[46] W. Saber, W. Moussa, A. M. Ghuniem, and R. Rizk, "Hybrid load balance based on genetic algorithm in cloud environment," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 3, p. 2477, 2021.

[47] A. Iranmanesh and H. R. Naji, "DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing," *Cluster Computing*, vol. 24, no. 2, pp. 667–681, 2021.

[48] A. Kishor, R. Niyogi, and B. Veeravalli, "A game-theoretic approach for cost-aware load balancing in distributed systems," *Future Generation Computer Systems*, vol. 109, pp. 29–44, 2020.