Hindawi
Mobile Information Systems
Volume 2022, Article ID 8327451, 12 pages
https://doi.org/10.1155/2022/8327451

Research Article

Artificial Intelligence-Based Framework for Scheduling Distributed Systems Using a Combination of Neural Networks and Genetic Algorithms

Abdolreza Pirhoseinlo, Nafiseh Osati Eraghi, and Javad Akbari Torkestani

Department of Computer Engineering, Arak Branch Islamic Azad University, Arak, Iran

Correspondence should be addressed to Nafiseh Osati Eraghi; n-osati@iau-arak.ac.ir

Received 21 October 2021; Revised 19 January 2022; Accepted 21 January 2022; Published 8 February 2022

Academic Editor: Sang-Bing Tsai

High-performance distributed computation has emerged as a new development in which demand for access to resources over the Internet is being delivered to distributed servers that are dynamically scalable. One of the important research issues that must be considered in order to achieve efficient performance is timing. The goal is to map tasks to the right resources and optimize one or more subgoals so that users get the most out of their resources as quickly as possible. Therefore, in this study, in order to find a suitable solution to the scheduling problem and reduce the total time of task completion, a hybrid approach is proposed by neural networks and genetic algorithms. In this hybrid algorithm, the predictive power of artificial neural networks is used to improve scheduling problem task in order to predict the time of tasks completion on resources and the meta-heuristic of genetic algorithm is used in order to find optimal resources for tasks. Experimental results show that the performance of the proposed method is improved in Makespan compared to other methods available in the journals and can adhere to the principles of service quality.

1. Introduction

Recent advances in hardware, software, and communication technologies, as well as the creation of heavy computing tasks, have stimulated the development of parallel and distributed systems for real-time computing tasks. As such, distributed systems are widely available to users, and users process their computing resources without worry [1]. The distributed system consists of heterogeneous interconnected sources with poor connectivity, and through communication, media communicate only by sending messages and have no shared memory [2]. On the other hand, the popularity of distributed systems due to transparency in user interaction has led to the creation of responsibilities for the proper distribution of user tasks across resources satisfying performance metrics such as runtime, resource efficiency, and throughput and response time [3]. With the development of distributed systems, the number of resource access requests is increasing and is creating a competitive environment. So, resource provision is emerging as a stage to identify sufficient resources in distributed systems for a given amount of work based on resource performance criteria [4, 5].

Currently, applications in distributed systems include web service, scientific computing, and file storage. In general, a program in a distributed system can be divided into a number of tasks and run on different sources. The performance of a program in a distributed system depends on the allocation of program-related tasks over available resources, known as task scheduling and allocation. The performance of a program in a distributed system depends on the allocation of program-related tasks over existing resources, which is known as scheduling and task allocation problems. Distributed systems can, in this case, shift tasks from heavy sources to light-loaded sources to reduce waiting times in resources, which is called load balancing. It is commonly said that workload and load balancing are very important for distributed systems [6].

The purpose of scheduling different tasks in resources is to increase execution speed, reduce execution time, and minimize communication delays, communication costs, and priority problems. In distributed scheduling, the whole task is subdivided into subtasks and assigned to multiple sources, so they perform faster than single sources [7, 8].

Therefore, in this paper, a multilevel approach is used in order to optimally schedule tasks in distributed systems by a combination of predictive neural networks and meta-heuristic and evolutionary features of the genetic algorithm. In this new approach, by neural network prediction, we first estimate the time taken to complete the existing work on computational resources. On the other hand, by considering computational resources in addition to the duration of tasks on resources, the cost of executing tasks is also significant and is a concern of the user that addressed the issue. In this article, given the cost of executing the tasks, we used the clustering method in the second level. After this step, in the third level, to find efficient resources to execute the selected tasks, we use the meta-heuristic of the genetic algorithm to improve population production and combine them and improve the total execution time of the tasks. At this level, the produced chromosomes are the same line of each source and in each cluster, which includes the previous tasks in the line of each new assigned resource and task, and the cutoff point is to jump to the same point of entry of the new task into the queue of a generator. The chromosomes corresponding to the sources in a cluster are compared, and the best combination is selected for the chromosomes. The new tasks are then joined to the queue of the most appropriate combination between the resources in the cluster. In this way, the proposed method can produce an optimal solution for mapping tasks in resources. The contribution of this paper is summarized as follows:

(i) Predicting the completion time of tasks in resources according to the characteristics of resources and tasks using neural networks

(ii) Ranking the tasks based on less completion time and their execution deadline

(iii) Determination of primary chromosomes of genetic algorithm based on the order of execution of tasks

(iv) Using the multicriteria fit function in accordance with service quality goals in the genetic algorithm

(v) Mapping tasks to resources based on finding the optimal solution in a genetic algorithm

The structure of the article is as follows: in Section 2, we will discuss the tasks related to the timing of distributed systems. Section 3 will detail the proposed method. In Section 4, the results of the experiments will be stated. Finally, in Section 5, we will discuss and conclude the article.

2. Related Work

About the problem of distributed scheduling of multiprocessor system resources, Konar et al. have proposed a multipurpose real-time scheduling algorithm for a multi-objective multiprocessor system called Mo-QIGA. This algorithm is implemented in real time to minimize the completion time and overall deadline of each task [9]. Aditi et al. have proposed a genetic algorithm-based scheduling program with three conflicting goals. This algorithm provides an efficient representation of the chromosomes, which is the complete solution to the problem, and the validity of the chromosomes is reliable even after fitting and mutation [10]. Izadkhah has developed a genetic learning algorithm for static scheduling in homogeneous distributed systems by two learning criteria, namely steepest ascent and next ascent learning, which use penalty and reward concepts in learning [11]. Akbari et al. have proposed a genetic-based algorithm as a meta-heuristic to deal with static task scheduling of processors in heterogeneous computing systems. This algorithm is introduced through significant changes in its genetic functions, and the introduction of new operators guarantees population diversity and continuous coverage of the whole space and improves the performance of the genetic algorithm [12]. To solve the problem of resource allocation in heterogeneous distributed systems, *Pan* et al. have proposed a distributed resource scheduling based on a hybrid genetic algorithm [13].

About the issue of resource scheduling in a distributed cloud environment, Casas et al. have proposed a GA-ETI scheduling plan to address all the challenges in cloud scheduling by providing some efficient solutions for scientific implementation in the cloud resources [14]. Dasgupta et al. proposed a novel load balancing method by the genetic algorithm. This algorithm is set to balance the load on the cloud infrastructure while trying to minimize the length of the given tasks [15]. Kashanchi et al. have used the benefits of evolutionary genetic algorithms and heuristic approaches to analyze and model behavioral scheduling. In this method, the expected specifications for scheduling are extracted in the form of linear time logic (LTL) formulas, which are used to achieve optimal scheduling of the LTS-labeled transfer system [16]. Amjad Mahmoud et al. have presented a greedy genetic algorithm with an appropriate selection of fit and jump operators (called AGAs) for the assignment and timing of real-time tasks that have precedence constraints on heterogeneous virtual machines [17].

In [18], a graph-based extreme learning machine method (G-ELM) is proposed for imbalanced epileptic EEG signal recognition. In [19], a method aimed to improve the early clinical diagnosis rate of atrophic gastritis (AG) and reduce the risk of disease deterioration or cancerization has been proposed. In [20], a group of dummy query sequences, to cover up the query locations and query attributes of mobile users and thus protect users' privacy in LBS, has been constructed. In [21], a location privacy-preserving system for LBS by constructing "cover-up ranges" to protect the query ranges associated with a location query sequence has been proposed. In [22], a dummy-based approach for text retrieval privacy protection has been proposed.

3. Methodology

As mentioned earlier, in this research, a multilevel approach is proposed to improve the scheduling and reduce the total time of completion in distributed systems by a combination

of neural network prediction that cluster meta-heuristic and evolutionary feature of genetic algorithm. In this new approach, by the predictive neural network, we first estimate the time spent on the available resources. On the other hand, in distributed systems, in addition to the time-consuming tasks on resources, the cost of executing tasks is also significant and is a concern of the user, and this study addresses this case and considers the cost of executing the tasks, and in the second level, we have used a cost-based resource clustering approach. After this step, in the third level, to find efficient resources to execute the selected tasks, we use the meta-heuristic and evolutionary feature of the genetic algorithm to improve population production and their combination, and then, we improve the total execution time of the tasks. At this level, the produced chromosomes are in the same queues of each source as previous tasks, and the cutoff point jumps to the same entry point of the new task on the one-source queue. The corresponding chromosomes in the sources of a cluster are compared pairwise, and the best combination is selected for the chromosomes. In this way, the new tasks are joined to the queue of the most appropriate combination between the two sources. In this way, the proposed method can produce an optimal solution for mapping tasks on resources.

Based on the investigations among the types of distributed scheduling algorithms, the genetic algorithm has been exploited as the main algorithm for discussing the scheduling tasks of distributed systems in terms of speed of execution and its ability to search for possible solutions in distributed scheduling tasks. Genetic algorithm is one of the most common evolutionary optimization algorithms that search the answer to find optimal solutions to an optimization problem with incompatible and conflicting goals. But as mentioned earlier, the problem of the above algorithm is that the initial solution is a random mapping of tasks on resources. Therefore, the use of prediction tools for some of the required parameters will, to some extent, lead us to better performance. It also deliberately performs scheduling of resource clusters tailored to the task demanded by the user, scheduling and distributing loads between the resources of the distributed systems.

3.1. Prerequisites. The first action to be taken by the scheduler in scheduling the tasks of the distributed systems is to identify the appropriate set of resources that perform the tasks. The resources that can be used to execute tasks as well as gain capabilities are identified based on some of the features gained through distributed systems intelligence. Therefore, in order to improve the situation and to reduce the impact of the above problem in the process of allocating tasks on resources to execute, we have used neural network capability in predicting the execution time of tasks and aiming to find a suitable solution based on the prediction of runtime as the key to achieve optimal scheduling in distributed systems. In order to realize the performance of the neural network are considered these parameters as reasons of impact on runtime, being independent of each other in measurement and control, simplicity of estimation or measurement and simplicity of calculation, the characteristics of the resources in the computing environment to predict the execution time in tasks. These parameters are considered as inputs of the artificial neural network, and by determining the weight in the middle layer of the neural network, the output will be the predicted completion time for the computing environment resources of the distributed systems. Quadratic properties of distributed computing environment sources derived from standard CPU-performance-dataset in the standard UCI data repository.

It should be noted, however, that with respect to the properties of computational resources, artificial neural networks can somewhat predict the execution time of tasks. This prediction value from the neural network output as a parameter of similarity, together with the cost required for each source, contributes to cluster the computing environment resources of the distributed systems. The resources within the clusters will have approximately similar features, and the user will be aware of the completion time (waiting time for their task) and the cost of each resource. The new task that comes up is assigned to a resource with the least expected completion time. The genetic algorithm is then implemented to optimize resource allocation tasks and reduce work completion time, and a resource queue with the least job completion time is anticipated for the newly assigned task and a resource queue that assigned for the completion time of the predicted task received the second rank as the primary population. The genetic algorithm, by jumping between these two queues, determines the most appropriate place where a new task can be added. The architecture of the proposed method is shown in Figure 1.

3.2. Proposed Scheduling Algorithm. The proposed method combines the strengths of the two genetic algorithms and the neural network in order to achieve a more optimal algorithm. These methods are combined in such a way that they cover each other's weaknesses so that they result in an integrated and optimized algorithm. Scheduling operations start by the neural network capability to predict the completion time of the tasks in terms of the four mentioned parameters in the previous sections. The structural features of the used neural network are as follows: (a) formation of 12 hidden layers, (b) application of the back-propagation algorithm, and (c) and application of evaluation function. This neural network is used in CPU-performance-dataset standard data of the standard UCI data repository, using MATLAB simulator based on learned resource attribute values and its suitable outputs in the task scheduling process to optimize the task mapping process.

3.2.1. Prediction Completion Time Using Neural Networks. Due to the fact that in the proposed method, various tasks are sent, and there are also heterogeneous resources in distributed systems, so each task and resource will have its own characteristics. One of the features of workflow tasks is the number of instructions in each task. In order to perform a task, each task must execute the instructions in the task text in order to produce output results based on these

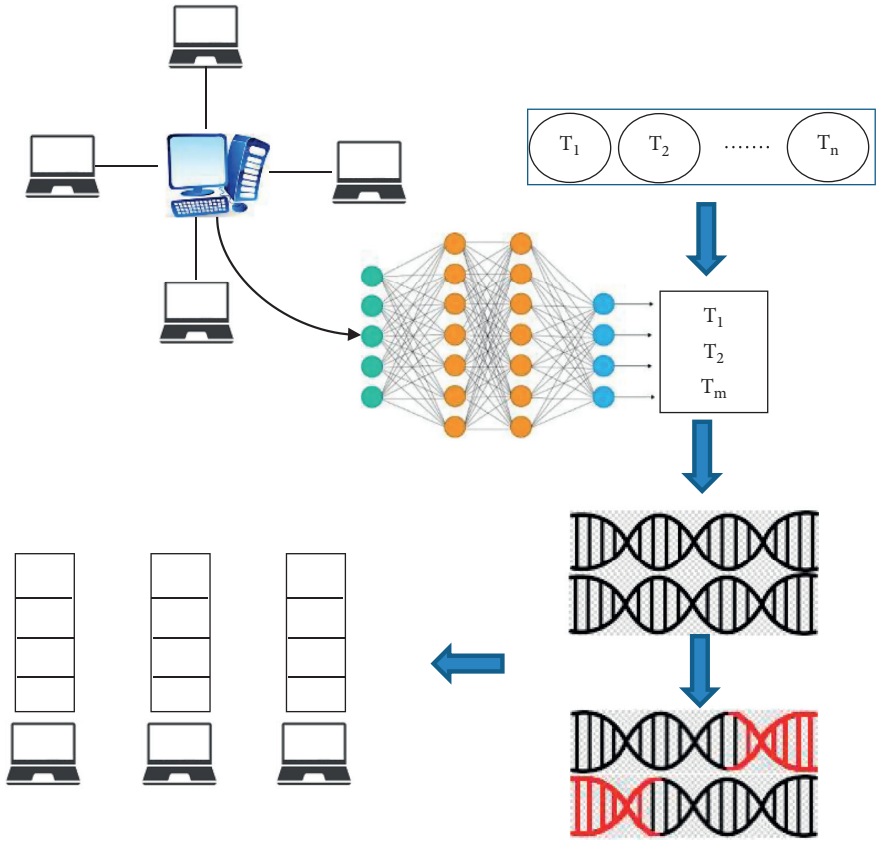

Figure 1: Architecture of the proposed method.

instructions. The unit of measurement is the number of instructions based on MI, which indicates the number of instructions on a million scale. In contrast, for processing instructions in virtual machines, the power of virtual machines is introduced based on MIPS, which is on the scale of millions of instructions per unit time.

The second feature that is considered in the proposed method for tasks is the deadline for performing tasks. In the proposed method, a deadline has been set for each of the tasks, according to which the tasks must be completed by the time they arrive. The deadline for performing tasks starts from the moment of entering each task. Depending on the length of each task, the deadline for performing tasks is naturally different. Therefore, tasks that have a shorter deadline should be performed sooner so as not to disrupt the workflow process.

In the proposed method, resource properties are prepared from the standard data set and include the minimum cost required to execute an instruction in the resource, the maximum cost required to execute an instruction in the resource, the minimum time required to execute an instruction in the resource, the maximum time required to execute an instruction in the resource, the maximum length of the resource queue to store the instruction, the minimum time required to transfer instruction to the resource, the minimum cost required to transmit an instruction to the resource, and the energy constant to execute an instruction in the resource.

This method has been used to predict the completion time of tasks in resources using neural networks. The input of neural networks includes the characteristics of the number of task instructions, the task execution deadline, and the minimum and maximum time required to execute instructions in the resources. Task completion times in prioritization sources in order to find eligible tasks for early execution are predicted. Hence, the existing tasks are trained based on the features mentioned in the neural networks. Hence, the output of neural networks is used as a predictor for the completion of tasks in resources. In the proposed neural network, in order to evaluate the tasks, the proposed fitness function in the following relation has been used.

$$\begin{aligned} &\min f = \sum_{i=1}^{n}\sum_{j=1}^{m} \text{deadline}(i,j) + \sum_{i=1}^{n}\sum_{j=1}^{m} \text{Tlen}(i,j), \\ &\text{s.t.,} \\ &\sum_{i=1}^{n} \varepsilon_j * T_{len}(i,j) \le \text{deadline}(i,j), \\ &i,j > 0, \end{aligned} \tag{1}$$

where n represents the total number of tasks, m represents the total number of resources, i represents the task index, j represents the source, deadline, (i, j) represents the deadline for performing the task in resource allocation, Tlen (i, j) indicates the number of task instructions i that need to be processed in virtual machine j, and εj represents the average

execution time of instruction in virtual machine j. According to (1), as can be seen in this optimization problem, the proposed method is required to complete the tasks before the deadline. According to the proposed evaluation function, each task will have a value of the function that is the less value, the shorter the estimated time to complete the task. In other words, having shorter execution times and shorter instructions lengths can be a good option for early execution, and an estimate of the completion of such tasks in resources will be needed to obtain a ranking.

3.2.2. Scheduling Tasks Based on Genetic Algorithms. As mentioned, in this research, genetic algorithm has been used to schedule the ranked tasks of the distributed system. Genetic algorithm is one of the effective optimization algorithms used in complex problems that are inspired by genetic and evolutionary mechanisms in natural systems. Genetic algorithm as a population-based meta-heuristic approach has different operators that generate optimized chromosomes from a primary random population. Genetic algorithm operators are essential for the convergence of population chromosomes toward optimal points. The genetic algorithm in the basic form has three main genetic operations of selection, crossover, and mutation. Some solutions are selected as chromosomes by the population selector operator as the parent and combine the appropriate parent operator to create offspring. Parents combine to replace several genes on the first chromosome with the second chromosome. The mutant operator also modifies the offspring according to the laws of mutation by changing the amount of one or more genes in the direction of the optimal solution.

Initial Population Creation: the first step in using a genetic algorithm is to encode and generate a random primary population. Coding refers to how genes are determined on chromosomes, where the arrangement of genes together leads to a solution to a scheduling problem. Therefore, each chromosome carries genes that are equal to the number of virtual machines. The index of each gene represents the virtual machine index, and the value inside the gene represents the task index. Figure 2 shows an example of the chromosomes presented in the proposed method. In this type of encoding, tasks are assigned to virtual machines.

In the proposed method, this type of coding is used in which the value of each gene represents the virtual task index. In the proposed method, since the tasks are ranked in the first step, but the distribution of values within the genes is random, the processing power of the virtual machine is also considered as one of the parameters of the fit function of the genetic algorithm. Therefore, in chromosomes where high-ranking tasks are assigned to powerful resources to be processed faster, the value of the fit function will be more optimal.

Crossover and mutation algorithm in the proposed method: given that in the proposed method, input tasks are prioritized, and naturally, there should be a slight change in the fitting and jump operators so that the priority of

G_1	G_2	G_3	G_4	G_5	G_6	…	G_M
5	2	4	7	6	1	…	n

Figure 2: Inscription of initial chromosomes based on tasks.

performing tasks in cloud resources is not disturbed. Given that in optimal chromosomes, high-ranking tasks are allocated to powerful resources, fitting and mutation should be in line with improved scheduling. Therefore, in this method, the fitting point and the adaptive jump will be used. Thus, if one fitting point and mutation cause the value of the fit function to deteriorate, another fitting point and mutation will be selected for the chromosomes. This operation is repeated until the optimal fitting and jumping points are selected. In this case, we will be sure that the generation of chromosomes will evolve and converge towards the optimal points.

Proposed fitness function: as mentioned in the proposed method, we will use the multicriteria fit function. Criteria for use in the fit function will include task completion time, task length, and source processing power. The main purpose of the proportionality function is to reduce the completion time of all tasks in resources, also called Makespan. The most important purpose of the task scheduling is to minimize Makespan. Makespan equals the total required time to complete all input tasks. Each scheduler uses a performance-matching function to perform the scheduling. And by using time and cost parameters, it determines the suitability of different resources to perform a task. Relation (2) shows the fitness function considered for the proposed algorithm.

$$\min f = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ti}(i,j) + \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ci}(i,j) + \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ri}(i,j),$$

$$S.t.,$$

$$\sum_{i=1}^{n} w_{ti}(i,j) \le w_{tseq}(i,j),$$

$$\sum_{j=1}^{n} w_{ci}(i,j) \le \alpha,$$

$$\sum_{j=1}^{n} w_{ri}(i,j) \le \text{deadline}(i,j),$$

$$i, j > 0, \tag{2}$$

where w_{ti} is the time required to complete task i in resource j, w_{ci} is the cost required to complete task i in resource j, w_{ri} is the estimated time based on the processing power of source j to perform task i, and α is the cost threshold for performing tasks. In this function, the weight of the completion time and the performing cost of the task (wt and wc) are taken into account by the user. Time and cost are not of the same scale, and they are not in the same range; to solve this problem, we have to normalize them and mapped them in the range of zero to one. It should be noted that the weighting coefficients are specified by the user in introducing the task. W_t and W_c give more freedom to user in presenting his work to

distributed systems. The purpose of the scheduler is to find a source j for each task i so that EF (t_i, r_j) be minimum.

The above objective function determines the source fitting of j to execute task i. As can be seen in the above function, the normalization of time and cost is multiplied by their weight. In the above function, t_{max} and t_{min} are the maximum and minimum completion times of tasks on the appropriate resources (in the previous sections are described the appropriate resource indicators). c_{max} and c_{min} are the maximum and minimum cost of executing tasks on appropriate resources. $t_{i,j}$ and $c_{i,j}$ are the time and cost of executing i task on j source. The execution time of tasks assigned to a resource is obtained by the relation as follows:

$$w_{ti}(i,j) = \sum_{i=1}^{n}\sum_{j=1}^{m} T_{len_i} * \frac{\text{JobCycle}_j}{\text{Processorspeed}_j} + w_{wi}(i,j), \quad (3)$$

$$w_{wi}(i,j) = \sum_{i=1}^{k}\sum_{j=1}^{m} Q_{T_j} * T_{len_i} * \frac{\text{JobCycle}_i}{\text{Processorspeed}_j}, \quad (4)$$

where the T_{len_i} parameter represents the length (number of instructions) of task i, $\text{JobCycle}_j/\text{Processorspeed}_j$ determines the speed of resource processor j, and w_{wi} is the waiting time to access the resource Q_{T_j} and number of tasks within the source queue. Running time is considered to be a $1 * M$ matrix where M is the number of resources. The value of the j-element of this matrix indicates the execution time of the tasks assigned to the source of the number j. Figure 3 shows an example of this matrix, in which the execution time of the assigned tasks to the second source is 250.

In each iteration, the updated population evaluated the produced solutions. This evaluation is based on the fitness function. To evaluate a member of the population (a solution to the problem), each of the independent input tasks is allocated to resources based on the strategy of the scheduling algorithm. After allocating tasks to resources, to determine what tasks are assigned to each resource and for each resource j, a $1 * N$ matrix is called Job-CPUj (N is the number of independent input tasks). The values within the Job-CPUj matrix are the number of tasks assigned to the source j. Figure 4 shows an example of this matrix in which the first, second, and seventh tasks are assigned to source j.

The total execution time of tasks assigned to source j (tasks within the Job-CPUj matrix) is calculated by relation (3), and finally, the execution time of the tasks is calculated in each resource and placed in the running time matrix. Then, the maximum runtime in the running time matrix is considered as Makespan. The proposed method algorithm is shown in Table 1.

4. Performance Evaluation

Since the proposed method uses a combination of two artificial neural network algorithms and a genetic algorithm, in order to evaluate the performance of the proposed method, the performance accuracy of both algorithms is estimated by the neural network in ready tools and genetic algorithm in the Matlab toolbox.

R_1	R_2	R_3	R_4	…	R_M
100	250	270	165	…	300

Figure 3: An example of a running time matrix.

T_1	T_2	T_3	T_4	…	T_n
1	5	3	7	…	N

Figure 4: An example of a Job-CPUj matrix.

4.1. Performance Evaluation of Neural Networks. This paper assumes that a task has been handed over to users of distributed systems or other applications. The timing of these tasks is considered at random. A number of resident tasks are also randomly distributed in the queue of processors in the distributed systems where new tasks will be assigned to the processor upon completion of these tasks (waiting time of the processor to be assigned to the new task or the time that tasks are completed on the processor). As mentioned, the neural network has been used to predict the time of tasks completed in the queue of a processor consisting of three layers: input, midst (middle), and output. The input data contain CPU-related features to apply in the neural network input layer as shown in Figure 4. The neural network in the middle layer performs the process necessary to train the model on the received input data from the input layer and presents the results to the output layer. As mentioned in the previous chapter, the applied neural network structure uses the evaluation function, which is a function of error backpropagation to correct weights. Figure 5 shows the accuracy of the evaluation function.

As shown in Figure 5, the performance of the neural network using the evaluation function after 40 iterations is better both in model training and validation and in the test stage of trained samples (lines shown by dot).

In this study, another criterion used to evaluate the accuracy of the proposed model is the correct prediction rate of the samples in the model training stages: validation and experimental data of the model replication stages. For this purpose, a graph is drawn as confusion, which specifies the number of correctly classified data against the incorrect data classified in the training, validation, and testing stages of the data. Figure 6 shows the confusion diagram of the proposed model in steps.

As shown in Figure 6, the proposed neural network model in this study has correctly predicted 100% of training samples in the training phase, 87.5% of training samples in the validation stage, and 75% of experimental samples in the testing phase. Therefore, the average accuracy of the whole proposed neural network model to predict the completion time of tasks residing on the CPU is 94%.

4.2. Implementation of the Clustering Method. As mentioned, the K-means clustering method is one of the most popular unsupervised learning algorithms that solve the problem of well-structured clustering. This method looks for a simple and easy way to classify data according to a set of

Table 1: Proposed method algorithm.

Proposed method algorithm

Input data:
F: similarity criteria features of processors;
C: similarity criteria features of clusters' center;
N: number of processors;
T: the task with different process time;
Begin
For $i = 1$ to N
P = predict the end time of task in each processor's queue using *neural network*
$D = \sqrt{(C_j - F_i)^2}$;//Clustering processors in different clusters, $j = 1, 2, 3$//End
For $i = 1$ to size of max cluster members ($\approx N/3$).
$G = \mathrm{GA}(P,T)$
Makespan = sum *(G,P)*
End

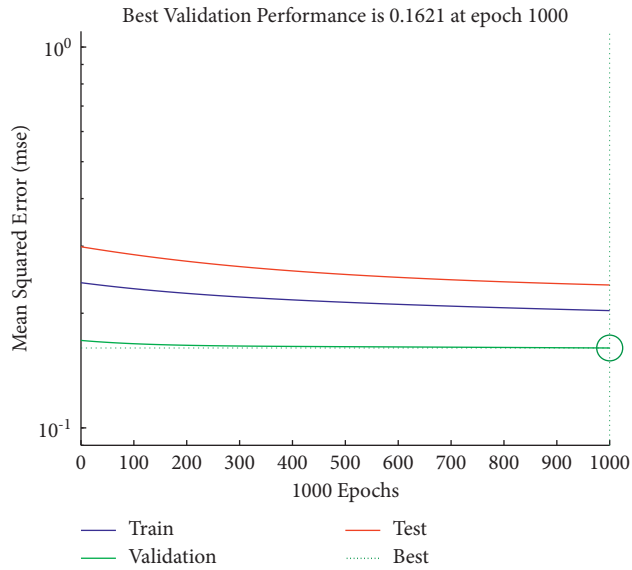

Figure 5: Evaluation function accuracy.

predetermined number of clusters (K cluster). The basic idea behind this method is to define the k as the central focal point, one for each cluster. This focal point should be chosen wisely because different situations give different results. Therefore, it can be concluded that the best choice is to keep them as far apart as possible. In this study, K is equal to 3, at which the focal point of the first cluster has the lowest value, the focal point of the second cluster has the midst data, and the focal point of the third cluster has the highest value. The clusters are divided into three clusters: cheap, medium, and economically expensive, based on the amount of time and cost of completing their tasks. The next step is to capture each point in the dataset and connect it to the nearest central point. When no point is waiting, the first stage is over, and the initial groups are formed. At this point, it is necessary to calculate the points at K of the new central point as the centerline of clusters delivered from the previous step. After obtaining this new K focal point, a new connection must be established between the same set of data points and the nearest new focal point. Hence, a loop is generated, and as a result of this loop, we may find that the central points of K in each time shift their position until a step is completed without any change. In other words, the focal point has no further movement. Table 2 shows resources allocated to clusters based on resource characteristics.

As shown in Table 2, the processors belong to a cluster that is less distant than the focal point of that cluster. Processors within identical clusters are very similar. If the CPUs in different clusters are more cost-effective and time-consuming, tasks delivered to the distributed system environment are driven to one of these clusters that are more in line with their preference, depending on user or application concerns about the cost and deadline of the tasks.

4.3. Performance Evaluation of Genetic Algorithm. As mentioned, the genetic algorithm is used to map the task in the appropriate processor to reduce the total execution time of a task. Therefore, it is expected that the total waiting time and task processing time before applying the genetic

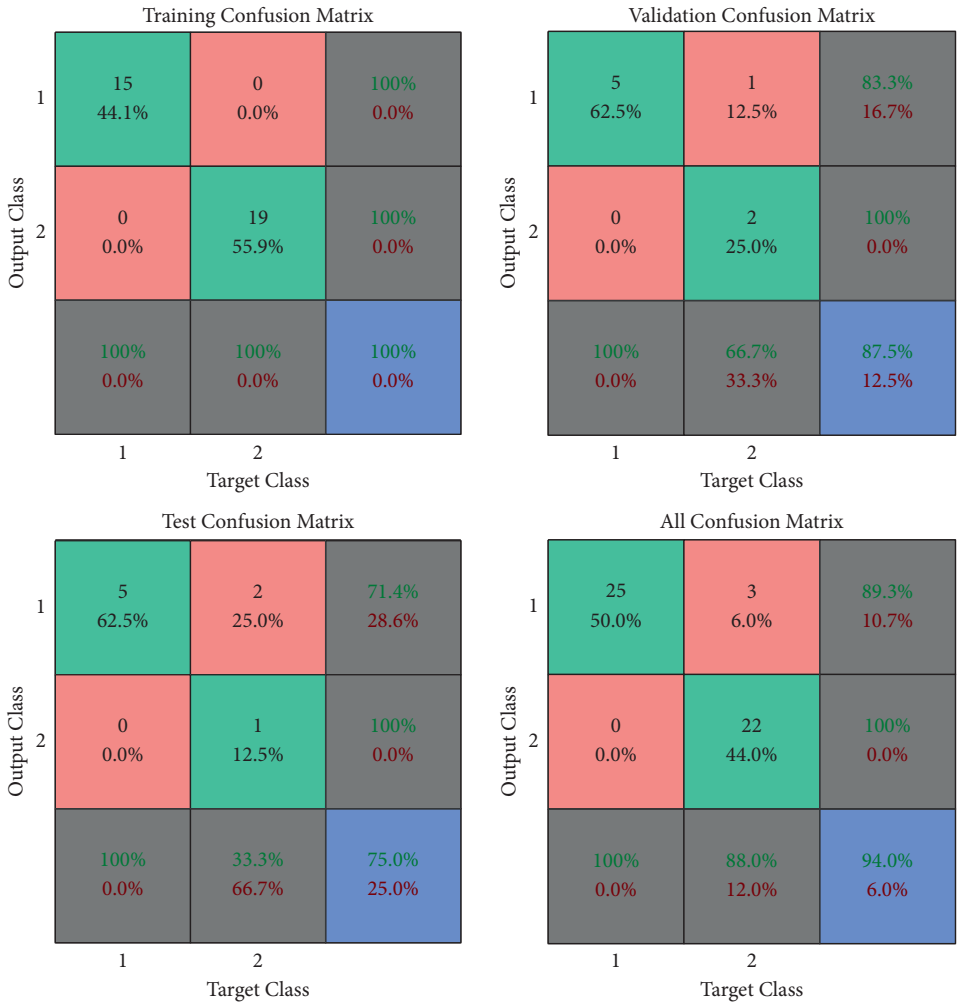

Figure 6: Confusion diagram of the proposed model.

Table 2: Assigned resources to clusters based on processing power.

Source#	Cluster#	Source#	Cluster#	Source#	Cluster#	Source#	Cluster#	Source#	Cluster#
1	3	11	2	21	1	31	2	41	2
2	2	12	1	22	1	32	2	42	1
3	3	13	2	23	1	33	1	43	1
4	2	14	3	24	2	34	1	44	2
5	1	15	1	25	1	35	1	45	1
6	1	16	3	26	3	36	2	46	1
7	1	17	2	27	2	37	1	47	3
8	2	18	2	28	2	38	2	48	2
9	2	19	1	29	1	39	3	49	1
10	2	20	1	30	2	40	1	50	2

algorithm will be different compared to the same amount after applying the genetic algorithm and allocating tasks on the most appropriate processor to reduce runtime. Due to the evolution of solutions, the genetic algorithm converges to the optimal solution during the iteration steps. Figure 7 shows the convergence of the genetic algorithm towards the optimal solution.

As shown in Figure 7, the genetic algorithm achieves the optimal solution after the iteration steps. Based on the optimal solution in Gantt's proposed genetic algorithm, the assignment of tasks to resources is shown in Figure 8.

As shown in Figure 8, the computational tasks are assigned to the appropriate virtual machines based on the proposed method. Then, in Figure 9, the completion time of the tasks is compared based on the proposed method and the neural networks without using a genetic algorithm.

As shown in Figure 9, the time to complete tasks in each resource in the proposed method is less than the neural

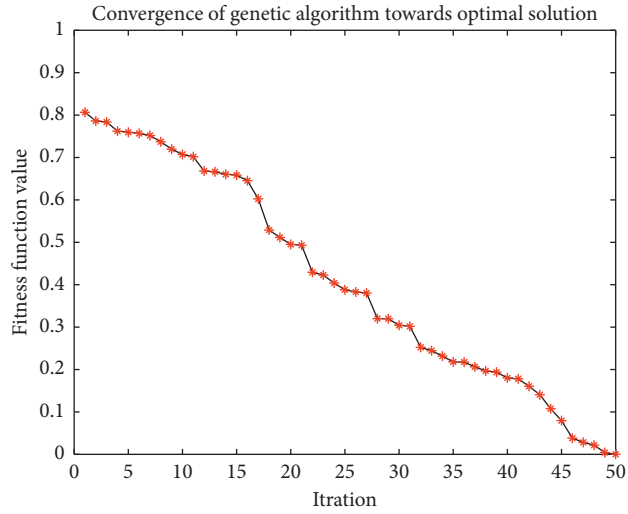

Figure 7: Convergence of genetic algorithm towards the optimal solution.

Tasks	1, 68	2, 95	3, 81, 116	4, 63, 147	5, 92	6, 57, 109, 140	7, 117	8, 59, 64, 75, 83, 127	9, 5, 110	10, 53, 56, 73, 93
Source	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
Tasks	11, 96	12, 106	13, 94, 142	14, 97	15, 123	16, 79	17, 98	18, 76, 148	19, 118	20, 51, 54, 80, 119, 143
Source	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}
Tasks	21, 124, 134	22, 84, 111	23, 69, 149	24, 85, 133, 144	25, 65, 107, 135	26, 85, 136	27, 89, 137	28, 90, 146, 150	29, 99	30, 70
Source	S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{26}	S_{27}	S_{28}	S_{29}	S_{30}
Tasks	31, 66, 138	32, 71, 77, 125, 128	33, 72, 100	34, 120	35, 101	36, 67, 121	37, 88, 102, 112	38, 103, 129	39, 108, 113	40, 122
Source	S_{31}	S_{32}	S_{33}	S_{34}	S_{35}	S_{36}	S_{37}	S_{38}	S_{39}	S_{40}
Tasks	41, 86, 130	42, 55, 74, 82,104	43, 60, 126	44, 114	45, 78	46, 91, 131	47, 105, 139	48, 61, 132	49, 87, 115	50, 62, 145
Source	S_{41}	S_{42}	S_{43}	S_{44}	S_{45}	S_{46}	S_{47}	S_{48}	S_{49}	S_{50}

Figure 8: Assigned tasks to appropriate resources.

network without genetic algorithm for finding optimal resources. Therefore, resource selection is not optimal on the basis of resource processing power and the provision of server services and estimated time to complete tasks in resources, and the combination of time and cost in task allocation can make this approach more optimal. Figure 10 also shows the total execution time of all tasks under all sources in the neural network-based method and the proposed method.

As shown in Figure 10, the total execution time of all tasks under all sources plus the waiting time to release all resources in the neural-network-based approach without the use of genetic algorithm is greater than the sum of the same time in the proposed method. The graph of the proposed method is linearly increasing with a slope lower than the previous graph, with respect to the tasks in the queue and the allocation of tasks to the most appropriate resource using the genetic algorithm proportional function. As shown in Figure 9, the neural network combinatorial method and genetic algorithm improve the Makespan criterion better than the neural-network-based scheduling method. In other words, the total execution time of the assigned tasks in the proposed method is about 24% less than the neural-network-based scheduling method.

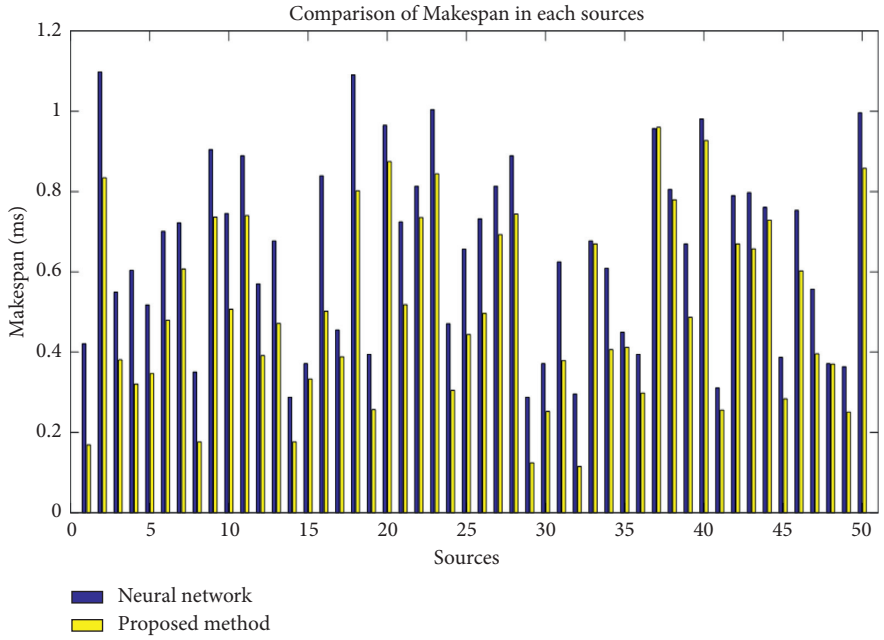

Figure 9: Comparison of the Makespan criterion for each source in the proposed method before and after applying the genetic algorithm.

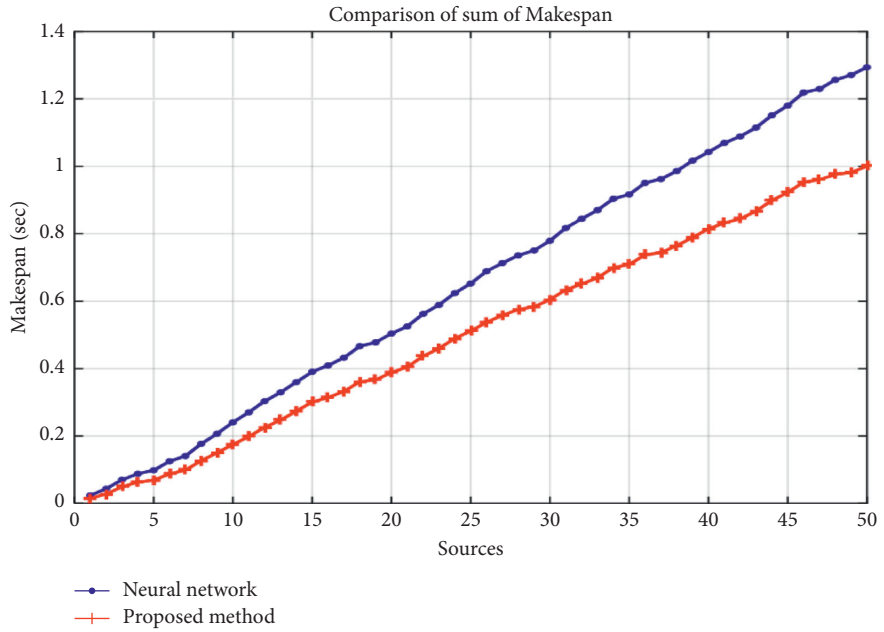

Figure 10: Comparison of the total execution time of all tasks under all sources in the neural network method and the proposed method.

We also compare it with previous methods to evaluate the validity of the proposed method. As shown in Figure 9, the accuracy of the proposed method in distributed systems is optimized by allocating tasks to resources and combining the parameters in the form of a proportional function. The proposed method can now be compared with previous methods based on the same criteria with equal conditions. Therefore, Figure 11 shows the comparison of previous methods with the proposed method in this study based on the Makespan criterion.

As shown in Figure 11, the performance of the proposed method was better than other methods available in the journals in allocating tasks to resources and reducing the time required to perform the whole task. Therefore, the

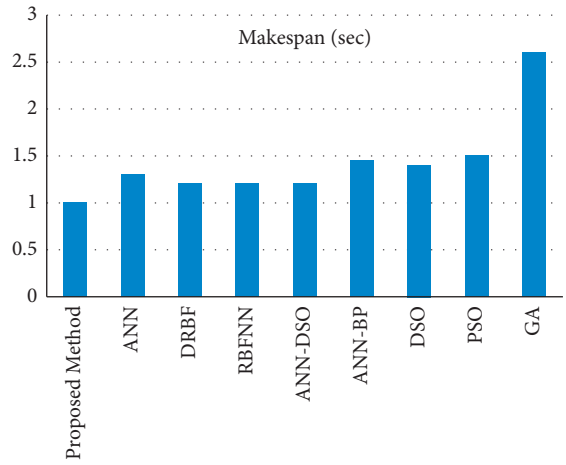

Figure 11: Comparison of the proposed method with previous methods.

proposed method has a lower value for the Makespan criterion than previous methods and can adhere to the principles of service quality.

5. Conclusion

In this study, a scheduling algorithm combining artificial neural networks and genetic algorithms that presented as a method for scheduling tasks in cloud computation aims to improve the task allocation conditions by reducing Makespan without violating service quality constraints. Predicting the time consumption of tasks in resources is the key to obtain optimal scheduling in the cloud. Application-level schedulers can use prediction to achieve better performance. The genetic algorithm also searches for the allocation of tasks to optimal resources by combining time and cost. By predicting the execution time of tasks in the cloud environment and using fast resources, task scheduling can be optimized. The purpose of this research is to minimize the total execution time of tasks assigned to each resource. In the proposed method, the parameters of service quality are considered that have a time constraint. An implementation result shows that the total execution time of the assigned tasks in the proposed method is about 24% less than the neural-network-based scheduling method. The performance of the proposed method is also better than other methods available in the journals in allocating tasks to resources and reducing the time required to complete the whole work. Therefore, the proposed method has a lower value for the Makespan criterion than previous methods and can adhere to the principles of service quality.

Data Availability

The data used to support the findings of this study are simulated in a random scenario.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Tyagi and S. K. Gupta, "A survey on scheduling algorithms for parallel and distributed systems," in *Silicon Photonics & High Performance Computing*Springer, Berlin, Germany, 2018.

[2] E. Ilavarasan and P. Thambidurai, "Genetic algorithm for task scheduling on distributed heterogeneous computing system," *Engineering Applications*, vol. 3, 2015.

[3] 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, no. 1, pp. 1–31, 2021.

[4] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: issues and challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, 2016.

[5] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 2019.

[6] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 585–599, 2015.

[7] A. Asghari, M. K. Sohrabi, and F. Yaghmaee, "Task scheduling, resource provisioning, and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2800–2828, 2021.

[8] J. C. Loftus, A. A. Perez, and A. Sih, "Task syndromes: linking personality and task allocation in social animal groups," *Behavioral Ecology*, vol. 32, no. 1, pp. 1–17, 2021.

[9] D. Konar, K. Sharma, V. Sarogi, and S. Bhattacharyya, "A multi-objective quantum-inspired genetic algorithm (MoQIGA) for real-time tasks scheduling in multiprocessor environment," *Procedia Computer Science*, vol. 131, pp. 591–599, 2018.

[10] A. Bose, T. Biswas, and P. Kuila, "A novel genetic algorithm based scheduling for multi-core systems," in *Smart Innovations in Communication and Computational Sciences*, pp. 45–54, Springer, Berlin, Germany, 2019.

[11] H. Izadkhah, "Learning based genetic algorithm for task graph scheduling," *Applied Computational Intelligence and Soft Computing*, vol. 2019, Article ID 6543957, 15 pages, 2019.

[12] M. Akbari, H. Rashidi, and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 35–46, 2017.

[13] S. Pan, J. Qiao, J. Jiang, J. Huang, and L. Zhang, "Distributed resource scheduling algorithm based on hybrid genetic algorithm," in *Proceedings of the International Conference on Computing Intelligence and Information System (CIIS)*, April 2017.

[14] I. Casasab, 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.

[15] G. Mandal and S. Dam, K. Dasgupta and P. Dutta, Load balancing strategy in cloud computing using simulated annealing," in *Proceedings of the International Conference on Computational Intelligence, Communications, and Business Analytics*, July 2018.

[16] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1–21, 2017.

[17] A. Mahmood, S. A. Khan, and R. A. Bahlool, "Hard real-time task scheduling in cloud computing using an adaptive genetic algorithm," *Computers*, vol. 6, no. 2, 2017.

[18] J. Zhou, X. Zhang, and Z. Jiang, "Recognition of imbalanced epileptic EEG signals by a graph-based extreme learning machine," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 5871684, 12 pages, 2021.

[19] J. Zhang, J. Yu, S. Fu, and X. Tian, "Adoption value of deep learning and serological indicators in the screening of atrophic gastritis based on artificial intelligence," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 1–20, 2021.

[20] Z. Wu, G. Li, S. Shen, X. Lian, E. Chen, and G. Xu, "Constructing dummy query sequences to protect location privacy and query privacy in location-based services," *World Wide Web*, vol. 24, no. 1, pp. 25–49, 2021.

[21] Z. Wu, R. Wang, X. Lian et al., "A location privacy-preserving system based on Query range cover-up or location-based services," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5244–5254, 2020.

[22] Z. Wu, S. Shen, X. Lian, X. Su, and E. Chen, "A dummy-based user privacy protection approach for text information retrieval," *Knowledge-Based Systems*, vol. 195, Article ID 105679, 2020.